ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUMBCM009

FRONT-END MONITOR AND CONTROL WEB APPLICATION FOR LARGE TELESCOPE INFRASTRUCTURES: A COMPARATIVE ANALYSIS

Stefano di Frischia*, INAF-OAAb, Teramo, Italy
Matteo Canzari, INAF-OAAb, Teramo, Italy
Valentina Alberti, INAF-OAT, Trieste, Italy
Athos Georgiou, CGI Scotland, Edinburgh, UK
Hélder Ribeiro, Universidade do Porto, Porto, Portugal

Abstract

A robust monitor and control front-end application is a crucial feature for large and scalable radio telescope infrastructures such LOFAR and SKA, whereas the control system is required to manage numerous attribute values at a high update rate, and thus the operators must rely on an affordable user-interface platform which covers the whole range of operations. In this paper two state-of-the-art web applications such Grafana and Taranta are taken into account, developing a comparative analysis between the two software suites. Such a choice is motivated mostly because of their widespread use together with the TANGO Controls Framework, and the necessity to offer a ground of comparison for large projects dealing with the development of a monitor and control GUI which interfaces to TANGO. We explain at first the general architecture of both systems, and then we create a typical use-case where an interactive dashboard is built to monitor and control a hardware device. Then, we set up some comparable metrics to evaluate the pros and cons of both platforms, regarding the technical and operational requirements, fault tolerances, developers and operators efforts, and so on. In conclusion, the comparative analysis and its results are summarized with the aim to offer the stakeholders a basis for future choices.

INTRODUCTION

Nowadays, the complexity of the most important scientific projects related to radio-astronomy require the design of large infrastructures involving both hardware and software side. A prominent software architecture challenge for the correct operational behaviour of such extensive infrastructure is not only the acquisition and processing of a huge amount of data, but also the management of an affordable monitor and control system. This system must be capable of inspecting a large number of variegated attributes and values of the station, and therefore allowing an automated or user-controlled action if a certain event occur during the operational time of the station.

A crucial part of the monitor and control infrastructure is played by the front-end application which usually allows the operators to manage the whole station configuration and its range of operations, and let them be able to act in real time if any action is needed. Since the control system is required to manage numerous attribute values at a high update rate, the front end application is required to possess significant ad strict requirements such affordability, consistency, security, fault-tolerance, user-friendly interface, and many other features which are covered in the present paper.

The two infrastructures being analysed in this paper are among the largest radio-astronomy facilities in the world, one fully operational and one under development: respectively LOFAR [1] and SKAO. LOFAR (Low-Frequency Array) is structured as an antenna network located mainly in the Netherlands, and spreading across 7 other European countries. Originally designed and built by ASTRON, it makes observations in the 10 MHz to 240 MHz frequency range with two types of antennas: Low Band Antenna (LBA) and High Band Antenna (HBA), optimized for 10-80 MHz and 120-240 MHz respectively [2]. The SKA (Square Kilometer Array) Observatory is an intergovernmental and international radio telescope project being built in Australia (low-frequency) and South Africa (mid-frequency) [3]. It is designed to reach a continuous frequency coverage from 50 MHz to 14 GHz. The frequency range from 50 MHz to 14 GHz requires more than one design of antenna, and so the SKA will comprise separate sub-arrays of different types of antenna elements that will make up the SKA-low, SKA-mid and survey arrays.

Regarding the front-end control interface, a comparative analysis between two different architectural and software choice of both the aforementioned observatories is covered in this paper. On LOFAR side, the software tool GRAFANA, a multi-platform open source analytics and interactive visualization web application, will be examined, since it is the LOFAR adopted choice at the present day. On the other hand, on SKAO side, the Taranta suite, a tool for creating dashboards and interacting with the devices within a TANGO Control System, will be examined as the SKAO preferred choice at the present day.

FRONT-END CONTROL APPLICATIONS

In the present section a concise explanation of the chosen front-end frameworks from the point of view of the software architecture is carried out. In particular, the description will be focused on the context, the purpose, the features, the roles and several other characteristics related to the scope of this paper, emphasizing thus their use in radio-astronomy facilities.

The comparative analysis has been chosen to examine the front-end web interfaces, due to their high impact to the

* stefano.difrischia@inaf.it

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUMBCM009

whole development of the observation facility. The use of a certain front-end framework impacts not only at a software developer level, but also (if not mainly) at the station operator level, and in the end at the stakeholder level. Front-end is the first and main interface with which the users and stakeholders are supposed to deal with, to inspect, to evaluate and to test the behaviour and the result data of the observation station.

As mentioned above, the comparative analysis will be focused on the monitor and control context of the station. Other main aspects which may be of primary importance in the development of the front-end framework such as observations dynamic scheduling, data correlation, reduction and analysis, are out of the scope of this paper.

In order to carry on the comparative analysis, a series of key features will be examined in the present study such Data Sources, Data Collection, TANGO Controls compatibility, visualization appeal, dashboards options, scalability, and many others.

Grafana

Grafana® is a multi-platform web application developed by Grafana Labs. It is designed to support analytics and interactive visualization as well as provide charts, graphs, and alerts when connected to supported data sources (for a general overview, see Fig. 1).

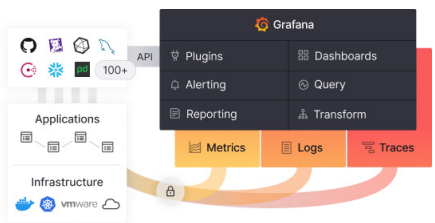

Figure 1: Grafana System Overview [4].

Grafana is commonly used in the fields of system monitoring, IoT, and network monitoring. Grafana framework is based on the definition of an expandable plug-in system. They can be essentially of two types:

- Built-in plugins (official release for DB services, mostly integrating some of the most used data infrastructure).
- Custom plugins (implementation of custom units based on the available resources or desired features).

The main features of Grafana Framework are its modularity and re-usability. The end users have the possibility to access a set of capabilities to customize dashboards layouts (visualization panels, interactive query builders, etc.). It is an Open Source project (Apache License v2). It consists of two main parts: a front-end layer and a back-end layer. The Front-end layer is written in TypeScript, JavaScript. It includes the plugins system that is used to visualize data and setup the data source connection and integration. The Back-end layer is written in Go. It includes all the function needed to run the Grafana service and plugin management (framework layout, etc.).

One of the main reason because Grafana has been adopted in several large infrastructure facilities is due to its capability to monitor and analyse in real time the vast amount of data that large-scale and distributed systems generate.

TARANTA

Taranta Suite [5] is a collection of web applications jointly developed by MAX IV Laboratory [6] and the SKA Observatory [7] for the Tango Control System [8]. Some contributions come also from the Tango community.

Its main purpose is to build a web application that allows a user to create a graphical user interface to interact with Tango devices. Its use is bound, therefore, to the adoption of Tango Controls as control framework of the infrastructure. Taranta provides tools which represent the underlying Tango devices as a tree, and thus viewing and modifing device properties, viewing and executing device commands, creating dashboards for interacting with Tango devices.

A Taranta application is composed of two main elements: the backend and the frontend application. The backend provides a GraphQL API [9] to a Tango control system. The client is a web application that provides a generic tango device view. Figure 2 displays how Taranta components are linked together. It is composed of several applications, or microservices, this way each component is developed and tested independently.

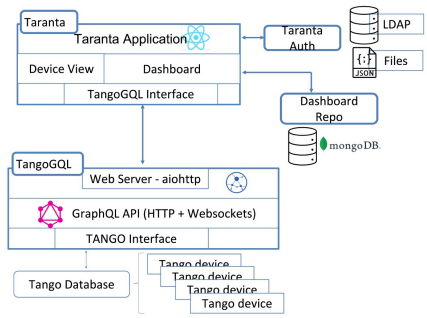

Figure 2: Structure of Taranta [8].

The front-end of Taranta is a React [10] application that is used both to browse, inspect and control Tango devices and to create and run dashboards, each composed of widgets. A widget is a dashboard component to serve the function of interacting with Tango devices. There are different types of widgets. A command can be sent to tango device through command related widgets. An attribute can be read/write through specific attribute widgets. Using widgets, users are able to monitor attribute changes and take remote control operations.

The existing widget can be grouped into several categories based on the element they are supposed to deal with [11]: *labels*, *attributes*, *commands*, *spectrum*, *grouping of widgets*, *dashboard*, and many others. The number of available widgets are increasing regularly thanks to requests from the user community.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUMBCM009

Taranta implements two main views and some common libraries.

- *Devices* is a view the user can use to control all the devices in the control system using a tree-like hierarchy (attributes, commands, and properties).
- *Dashboard* view provides functionality to build customizable, intuitive, and shareable views. It affords a collection of default widgets that the user can choose using drag-and-drop.

The Dashboard view provides a configurable user interface that enables users to create their UI starting from a blank page and dragging and dropping widgets into it. Each widget comes with its configuration, depending on the purpose of the widget, and provides a Tango device (or attribute) selection. In Taranta, a dashboard creation requires minimal knowledge of web technologies and no programming skills [12].

USE CASE DEFINITION

In order to perform an effective comparative analysis, a common use case between the two front-end frameworks must be set. Since we are dealing with monitor and control web applications in the domain of large telescope infrastructures, we decided to consider the following use case which can be representative of a real use case in the aforementioned projects like LOFAR or SKA: monitor a dashboard that shows the image of a Printed Circuit Board (PCB) which represents a real hardware device of the observing station. The main attributes of the device (and therefore their corresponding values) are placed upon the image with the possibility by the operator to read and/or modify them.

In details, the use case can be defined by the following steps:

- create a new dashboard inside the chosen front-end framework;
- place an image of the selected PCB as background image of the dashboard;
- retrieve from the data source the list of the attributes which belong to the chosen device;
- place each pair label-values upon the image, selecting the right format, graphic, unity, etc.;
- check if the dashboard is updated correctly after the device values updating;
- modify a read/write attribute if possible directly from the GUI.

Grafana Dashboard

First of all, Grafana requires a data source, among a list of supported ones, from which the device attributes will be retrieved. As data source for this use case Prometheus [13] DB has been chosen. Prometheus is an open source monitoring system and time-series database for which Grafana provides out-of-the-box support [14]. The final aim is to display on the dashboard the Tango device attributes which will be retrieved as system metrics from the Tango DB server monitored by Prometheus. So, the preliminary steps to configure this data source are: download and install Prometheus and Prometheus node_exporter (a tool that exposes system metrics), establish a connection between Prometheus and TangoDB server, and finally configure Prometheus for Grafana in the *Data Source* menu.

Then, the next step is to create a new dashboard, which will be composed by a single panel. This panel is the environment where a PCB image and the Tango device attributes will be placed. In the *Edit Panel* window all the operations to create, setup and modify the panel elements are present. In the *Data Source* tab of this window, our connection with Prometheus can be selected, along with the device metrics that references the device attributes that we want to show on the dashboard.

In order to place a PCB image as a background for the panel, the *ePict Panel* must be chosen among the *Visualization* panel list. Then the developer must provide an image URL and other optional graphic options like auto-scale, size, etc.

If the connection between Prometheus and TangoDB has been successful, the definition of a metrics that groups all the desired attributes should be enough simple like, for example, the following query:

```
device_attribute{
  host="stat_host",
  device=~"station/pcb/1"
}
```

At this point, the positioning of label-value pairs is ready to start, with the action of adding a *Box* in the *ePict Panel*. In the box window the developer can select the appropriate metrics, that is shown as the attribute value, the name that corresponds to the attribute label, and other optional accessory settings, like font size, color, etc. The position of the box, and therefore the label-value pair can be defined with fixed numbers, or can be simpler dragged and dropped on the desired point of the background image.

After saving the last panel updates and setting the desired interval, if everything has been set properly, the attribute's values are supposed to change according to the correspondent changing in the Tango DB. So, a simple monitor dashboard for a device has been built in few steps and relatively simple operations. A snapshot of a work-in-progress panel can be seen in Fig. 3.

The only downside of this architectural approach is the unavailability of modifying directly a read/write attribute from the relative dashboard box itself. In fact, Prometheus data source can act only as a monitor interface, while queries that modify the state of the Tango DB are not allowed. An implementation that overcomes this missing requirement is possible through dedicated Tango tools, but not yet exhaustively implemented neither in LOFAR nor in SKAO.

Taranta Dashboard

After the installation of Taranta suite (typically through Makefile and Docker-Compose), there should be no need to

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUMBCM009

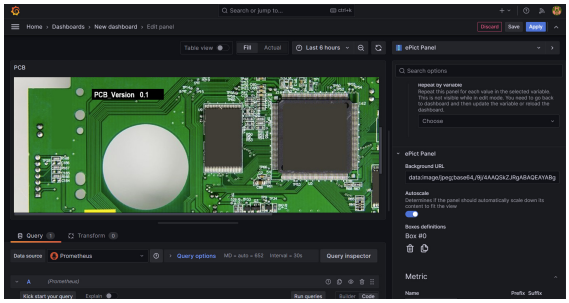

Figure 3: Grafana Edit Panel window.

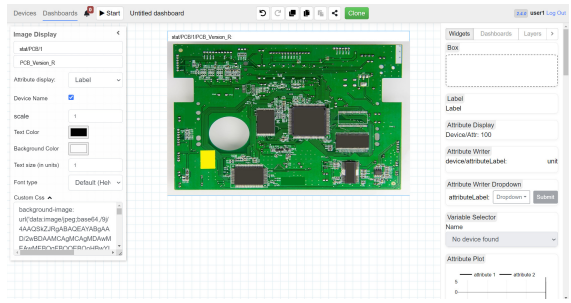

Figure 4: Taranta Dashboards view window.

configure any data sources, since the Tango DB host address is expected to be automatically detected, so the device and attribute list of the Tango running instance are loaded at the boot of Taranta. If everything has worked properly, the device list should appear in the left vertical bar of the *Devices* view. If the user clicks on a certain device, it is shown a new window in the right panel where attributes, properties, commands and logs of the selected device are listed.

Then, the aim is again to create a dashboard where a PCB image (which represents a real hardware device) is set on the background, and relative positions of its attributes are placed upon the image. It is necessary therefore to switch to the other view, the *Dashboard* window, in order to interact with the dashboard and widget creation environment.

After dragged and dropped a new *Box* widget on the main window, the operation to setting the background image is simply performed through adding a custom CSS code in the relative option among the ones in the Box configuration bar. For example, a possible entry could be the following:

```
background-image: url("<your-image-path>");
```

Once that the PCB image has been added and graphically adjusted, the attribute label-value pair can be placed. The action can be performed with dragging and dropping one of the available widget from the widgets bar. The *Attribute Display* widget has been designed to show the label-value pair, while, for example, the *Attribute Writer* or the *Attribute Writer Dropdown* widget have been designed not only to track the selected monitor point, but also to modify the value for a read/write attribute, in order to allow a direct action on the Tango controls framework by the end user.

After graphically adjusting the attribute monitor and control widgets, if everything has been set properly, the attribute's values are supposed to change according to the correspondent changing in the Tango DB and/or to our direct action. Note that here there is no refreshing interval, since the dashboard updating is handled by a web socket through an asynchronous event management. At this point, a simple monitor and control dashboard has been built in few steps, relatively simple operations, with possible CSS adjustments. A snapshot of a work-in-progress Taranta dashboard can be seen in Fig. 4.

One of the downsides of this architectural approach is that everything in Taranta is tailored on a monitor and control system which runs Tango Controls Framework. This makes the front end application less extendable if different data sources are needed in a single dashboard.

COMPARATIVE ANALYSIS

In the present section, a summary comparative analysis between the two front end frameworks is exposed, enlightening the pros and cons of both choices with the aim to offer the stakeholder a basis for future decisions.

For the sake of clarity we must emphasize that we are comparing two open-source applications which rely on a different scale community of developers. Grafana encompasses a community of about 2100 developers, while Taranta counts about 40 developers. This implies of course an imbalance of the available features and peculiarities, but nevertheless, they are contending nowadays the primacy as preferred front end frameworks on large telescope facilities.

From the point of view of the platforms and operative systems (OS) supported, Taranta lacks of a Windows installer, while Grafana supports its installation on all the common used OS. On the other hand, for our aforementioned use case, Taranta did not need any accessory applications outside the ones deployed after its installation, while Grafana needed a Prometheus DB and a Prometheus Node Exporter to work properly with Tango.

About the support and compatibility with the Tango Controls Framework, it is indubitable that both applications suit perfectly as a front end with Tango as middleware, but Taranta has proven to possess much features that enable the use of Tango commands and Tango read/write attributes, allowing the end user to directly act on Tango devices from the view. On the other hand, Grafana supports a multitude of data sources that make it a preferrable choice in case of multi-source monitor and control system.

Regarding the GUI and UX/UI aspects, the visual appeal of Grafana overcomes the essential interface of Taranta, but both of them result in a simple and straightforward experience from the point of view of the end user, which is required to possess basic software skills. From the developer perspective instead, Grafana could require to adjust the dashboard configuration with data sources queries in SQL or in Prometheus Query Language, while Taranta requires a minimum knowledge of CSS to tweak the dashboard options.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUMBCM009

Both of them support a system of Alerting, and provide Authentication/Authorization tools, while the scalability in a large scale can be achieved by several tools that have already been successfully tested: for example there is a dedicated Grafana Cloud platform, suitable for different requirements, while Taranta encourages the use of Minikube and Docker as preferred deployment tools. The results of comparative analysis are summarized in Fig. 5.

	GRAFANA®	TARANTA
Repository	https://github.com/grafana/grafana	https://gitlab.com/tango-controls/web/taranta-suite
Version	10.1.0	2.4.0
License	AGPL-3.0-only	LGPL 3.0
Target platforms	All	Windows not supported
Main frontend programming language	Typescript	React
Contributors	Around 2100	Around 40
Installation	Pretty straightforward	It may require to tweak some parameters in Docker Compose file
Other software needed for present use case	Prometheus, Prometheus-node Exporter	TangoGQL, MongoDB and other accessory tools are automatically installed
TANGO Controls support	Through Prometheus, or directly inject TangoDB as data source (f.e. MariaDB). Needs proper configuration.	Tailored on TangoDB, it needs only its host address to connect
Data Sources	Supports a multitude of data sources like Prometheus, Loki, Elasticsearch, InfluxDB, Postgres and many more.	Tailored on TangoDB
Developer preparation for dashboard creation	It may require to modify JSON files as well as write queries in the data source format	Minimum
End user preparation for dashboard interaction	Minimum	Minimum
On-line Support	Slack channels, community forums and dedicated Grafana Labs contact support	Developer community contacts and slack channesl
Documentation	Official docs along with tutorials, webinars, videos and blogs	Repository and Tango community documentation
Tool Customization	Lots of extensions and plugins developed by the community	No plugins outside the official suite
GUI usability	Highest. UX/UI dedicated development	High. Simple and straightforward
Mobile responsiveness	High. Dedicated development	Basic
Dashboard visual appeal	High. Many customizable graphic options.	Medium. It focuses on readability rather than graphic embellishment
Data visualization tools	A broad choice of widgets that should meet all data format needs. Less customizable from the point of view of the Tango device monitor and control.	A minor number of widgets but each one of them tailored on a specific Tango Controls feature
Dashboard customization	High, but it required more effort to adapt it to the present use case	High, it required less effort since it met easily the present use case
Alerting	Grafana Alerting System	Taranta Alerting features
Authentication and Authorization	Grafana Auth or other auth providers	Taranta Auth package
TANGO command support	Not present. Not allowed through Prometheus	Present. Dedicated widget
Dashboard minimum refresh rate	Between 1s and 5s	Handled by websocket through an asynchronous event management
Suitability for business companies	Suitable both for small and large companies	Suitable for large companies which adopt Tango Controls framework
Scalability	Dedicated Grafana Cloud platform suitable for different requirements	Minikube and Docker as preferred deployment tools

Figure 5: Comparative analysis table.

As a final result, it can be assert that both applications meet the needs of a complex monitor and control front end framework, suitable for large infrastructures, and in particular for those who rely on the Tango Controls framework as control system architecture. Grafana is perfectly suitable for any kind of systems and organization, especially if a on-the-fly setup is needed or if the system relies on several different data sources. On the other hand, Taranta, even if it is a younger competitor and can rely on a small community of developer, makes its integration and compatibility with Tango its strength point, allowing direct control features not present in Grafana.

CONCLUSION

In the present paper, a comparative analysis of two popular front-end applications for monitor and control large telescope infrastructures such LOFAR or SKAO has been carried on, elaborating a real use case. Both systems have proven to be suitable for this task, each one of them with its peculiarities and features. Grafana has proven to be perfectly suitable for any kind of systems and organization, while Taranta, even if it is a younger competitor, makes its integration and compatibility with Tango Controls framework its primary strength point.

ACKNOWLEDGEMENTS

Many thanks to the INAF staff, the ASTRON staff, the SKAO developers community and the TANGO community for their support, great work and ideas.

REFERENCES

[1] LOFAR, https://www.astron.nl/telescopes/lofar

[2] M. P. van Haarlem *et al.*, "LOFAR: The LOw-Frequency ARray", *Astron. Astrophys.*, vol. 556, p. A2, 2013. doi:10.1051/0004-6361/201220873

[3] P. Dewdney *et al.*, "The Square Kilometre Array", in *Proc. IEEE*, vol. 97, no. 8, 2009, pp. 1482–1496. doi:10.1109/JPROC.2009.2021005

[4] Grafana, https://grafana.com/

[5] Taranta Suite, https://gitlab.com/tango-controls/web/taranta-suite

[6] Max-IV Institute, https://www.maxiv.lu.se

[7] Square Kilometre Array (SKA), https://www.skao.int/

[8] M. Eguiraun *et al.*, "Taranta, the No-Code Web Dashboard in Production", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 1017–1022. doi:10.18429/JACoW-ICALEPCS2021-FRAR01

[9] GraphQL,Aquery language for your API, https://graphql.org/

[10] React, A JavaScript library for building user interfaces, https://reactjs.org/

[11] M. Canzari *et al.*, "How Taranta provides tools to build user interfaces for TANGO devices in the SKA integration environment without writing a line of code", in *Software Cyberinfrastructure Astron. VII*, vol. 12189, 2022. doi:10.1117/12.2630141

[12] M. Canzari *et al.*, "Satisfying wishes for SKA engineers: how Taranta suite meets users' needs", in *Software Cyberinfrastructure for Astron. VI.*, vol. 11452, 2020. doi:10.1117/12.2562585

[13] Prometheus, monitoring system and time-series database, https://prometheus.io/

[14] Get started with Grafana and Prometheus, https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-prometheus/