ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL005

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

A TECHNOLOGY DOWNSELECTION FOR SKA USER INTERFACE GENERATOR

M. Canzari*, INAF - Osservatorio Astronomico d'Abruzzo, Teramo, Italy
V. Alberti, INAF - Osservatorio Astronomico di Trieste, Trieste, Italy
P. Klaassen, M. Nicol, S. Williams, UK Research and Innovation, Edimburgh, UK
H. Ribeiro, FCUP (CICGE) Centro de Investigação em Ciências Geo-Espaciais, Porto, Portugal
S. Valame, Persistent Systems, Pune, India
V. Hardion, F. Bolmsten, H. Petri, Max-IV Institute, Lund, Sweden

Abstract

The Square Kilometre Array (SKA) project is an international collaboration aimed to design and build the world's largest radio telescope, composed of thousands of antennae and related support systems, with over a square kilometre of collecting area. In order to ensure the proper and uninterrupted operation of SKA, the role of the operator at the control room is crucial and the User Interface is the main tool that the operator uses to control and monitor the telescope. During the current bridging phase, a user interface generator has been prototyping. It aims to provide a tool for UI developer to create an own engineeristic user interface compliant with SKA User Interface Design Principle and operator and stakeholder needs. A technology downselection has been made in order to evaluate different web-solution based on TANGO.

INTRODUCTION

Square Kilometre Array (SKA) [1] is a project which aspires to build the biggest radio telescope in the world. It is composed of two arrays of radio-telescope: SKA1-LOW, consisting of about 200,000 dipole antennas that operate in the frequency range ranging from 50 to 350 MHz in Australia; SKA1-MID, composed of 197 dishes, covering frequencies from 350 MHz to 14.7 GHz, in South Africa. SKA General Headquarter is located in the UK. In order to ensure proper and uninterrupted operation, the experience and the skills of a human operator play a central role in controlling and monitoring a complex facility like SKA. Graphical User Interface (GUI) is the main tool the permits to the user to carry out such system control operations. Currently, the SKA project is in the bridging phase, the period between the Design Phase and the Construction, scheduled for the last quarter of 2020. The aim of this phase is to mitigate the risks raised during the Critical Design Phase and to develop early prototypes to adopt during the Construction phase. Considering the importance of the user's role and user interfaces, a downselection of the GUI technology has been necessary for this period.

USER INTERFACE

A large radio telescope like SKA is composed of several like electronic, mechanic, computer hardware and software components, which work in coordination with each other to carry out a high number of complex operations. The role of the operator is crucial during the monitor and control of the facility, in order to maximize the observing time, minimize the overhead and operational cost. GUI is the only tool available to the operator in performing such monitor and control operation and in collecting information useful for decision making. Considering the importance of the operator, SKA adopted the User-Centered Design (UCD) [2] Approach during the Design Phase. UCD is a process in designing and developing User Interfaces focused on the users, that has the aim to develop UIs that satisfy consumer's skills and expectations. The UCD process is summarized in the following diagram (Fig. 1).

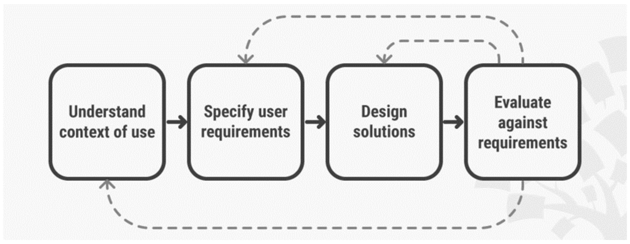
Understand context of use
Specify user requirements
Design solutions
Evaluate against requirements

Figure 1: User Centered Design process diagram.

The outcome of the analysis, the result of a series of interviews with SKA-precursor control room operators, can be summed up in a list of requirements [3] to take into account during the technological downselection:

- scalability
- integrated tools
- extendability
- completeness

In addition, during the Bridging Phase, SKA stakeholders required other functional and non-functional requirements. In particular:

- full compatibility with Tango Controls [4], the control framework adopted by SKA for the whole telescope
- adoption of open-source software, in order to share with the Tango community problems and solutions
- web-based software, in order to exploit the accessibility and deployability features of a web application

Starting from the requirements defined during the Design and Bridging phase, a SAFe [5] team has been formed in order to choose and develop a user interface generator to

* matteo.canzari@inaf.it

create engineering user interfaces that operators can use during the first period of the Construction of SKA. This prototype consists of a tool that engineers can use to generate and customize simple user interfaces. It needs to be intuitive to its users, without requiring a software background, or explicit knowledge of the underlying UI framework. It runs in the browser (to make it accessible) and it presents a rich and flexible interface.

In the Tango ecosystem, there are two software to evaluate in order to select the one that best fits SKA operators' needs and expectations: Waltz (proper TangoWebApp) and Wijive

WALTZ

Waltz (proper TangoWebApp) [6, 7] is a framework currently supported by the Tango collaboration. It emerged from a collaboration between several partners in the TANGO Community to implement a standard TANGO web platform designed to interact with the TANGO REST API [8]. It consists of a single page web application hence it uses standard technologies stack and a Model-View-Control pattern.

In the diagram below (Fig. 2), an overview of Waltz architecture has been shown.

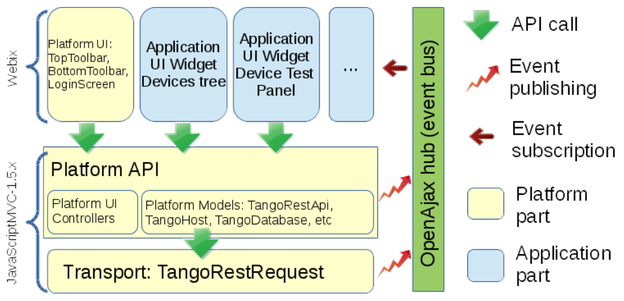
Webix
Platform UI: TopToolbar, BottomToolbar, LoginScreen
Application UI Widget Devices tree
Application UI Widget Device Test Panel
...
JavaScriptMVC-1.5.x
Platform API
Platform UI Controllers
Platform Models: TangoRestApi, TangoHost, TangoDatabase, etc
Transport: TangoRestRequest
OpenAjax hub (event bus)
API call
Event publishing
Event subscription
Platform part
Application part

Figure 2: Waltz architecture diagram.

Waltz implements a layered architecture. On the bottom, there is a layer with primitives that provide API to *Tango REST* i.e. transport layer. On top of the transport layer resides a Platform API layer i.e. non-UI primitives (models, controllers) that provide high-level API to Tango REST. On top of the Platform API layer, Waltz has a bunch of smart components.

In the diagram below (Fig. 3), the technologies involved in Waltz have been shown.

On top of *jmvc* [9] *webix* [10] is used for UI widgets and application graphical layout. Webix is one of the best JavaScript UI libraries alternative to JQueryUI nowadays. Finally, *plotly.js* [11] is used for plotting all kinds of graphs within the application. All the libraries are open source.

WEBJIVE

WebJive [12] is an experimental framework developed by the MAX-IV institute. It consists of a React-based [13] client, which communicates with a Max-IV developed GraphQL [14] server, named TangoGQL.

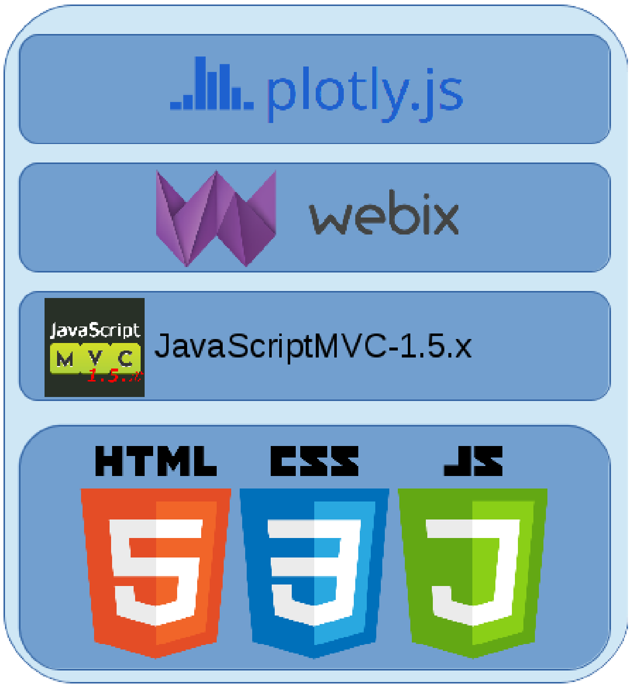

Figure 3: Waltz Technologies stack.

In the diagram below (Fig. 4), an overview of WebJive architecture has been shown.

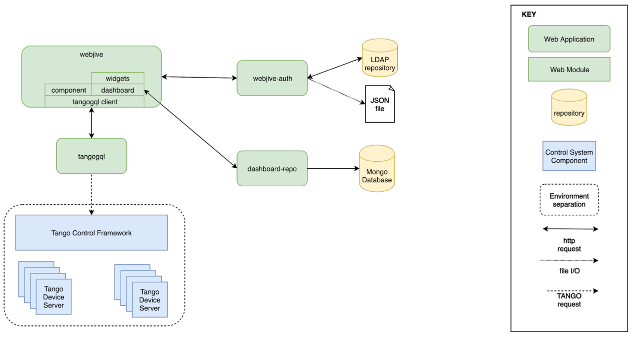

Figure 4: Webjive architecture diagram.

Webjive permits to explore Tango devices and to create custom dashboards, using a collection of widgets. it accesses to the Tango Controls Framework through tangogql. The communication between webjive and tangogql is managed by a library into webjive. tangogql is a GraphQL interface for Tango. In order to send command and save dashboards, it is necessary to login to the system. webjive uses webjive-auth to manage users. webjive-auth accesses to LDAP repository or JSON file to retrieve user information. The dashboards created by the user logged into the system, are stored in a MongoDB database through the dashboard-repo application.

In the diagram below (Fig. 5), the technologies involved in Webjive have been shown.

React is a JavaScript library for building user interfaces that guarantee to be fast, scalable, and simple, maintained by Facebook. *Redux* [15] is a predictable state container for JavaScript apps that permits to easily manage the state

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

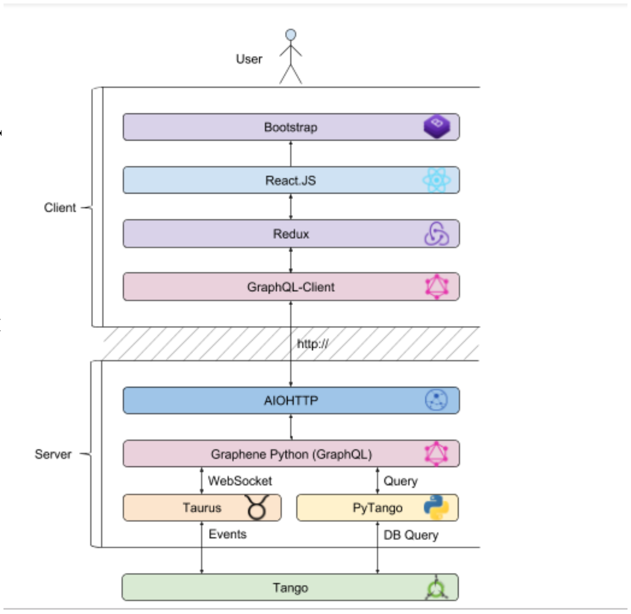

Figure 5: Webjive architecture diagram.

of on application. *aiohttp* [16] is an asynchronous HTTP Client/Server for asyncio and Python. GraphQL is a query language for APIs, that gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

TECHNOLOGY DOWNSELECTION

Once identified the technologies involved based on requirements, a list of valuation criteria has been defined. In particular, different types of scenarios that typically emerged when working with any technology has been considered. From scenarios, a set of neutral measures for what would count as a reasonable expectation for a platform or technology have been derived, and what the result would make using that technology unworkable, or at least make it significantly difficult for the developers to be able to deliver. The result of the analysis, divided by scenarios, is summarized in the following paragraphs.

Stages of Maturity

The two frameworks have different aims and are at different stages of maturity.

Waltz is a mature and established framework that makes it easy for developers, or those with software experience to create interfaces to TANGO systems. It is essentially a framework for developing fixed GUIs. It relies on a core set of widgets taken from other JavaScript libraries. The developer defines the layout of the GUI as part of the code in a way that would require either manual coding or an external tool, or further coding within the prototype to allow users to manipulate the layout of the widgets.

WebJive is a less mature framework, but one that is closely aligned to the type of 'Taurus [17] like' dragging and dropping of widget components into a view. Its equivalent widgets are React components.

Extending Waltz would require developing the code to allow users to 'drag and drop' widgets into their interface or screen. Extending WebJive would mean extending the framework to expose any missing interfaces. The ability to create scripts of commands for execution on a TANGO device, for instance, will require unpicking the Max-IV specific authentication layer to allow direct commands to be entered and passed on to the device.

Support from Community

Support from the community is an important consideration and can be considered as a predictor both of how easy it would be to get support if tackling any issues or limitations in the framework now. It also emerged as a long-term predictor of the longevity of the solution. The more people supporting a framework, the more likely it is to grow and evolve. Essentially the chance that if a maintainer loses interest that someone else from the community will fulfil that role.

For both Waltz and WebJive, there are a small number of core contributors. It is however clear looking at the GitHub stats for the underlying technologies that there is a far larger and more active community currently supporting the building blocks of WebJive than those of Waltz.

Patterns and Technologies used

Software is always in a constant state of evolution. For this reason, it is important to analyze if the technologies involved and the design pattern adopted in the framework guarantee the maintainability and make possible the evolution of the software.

While both Waltz and WebJive make use of a solid set of software patterns and technology choices, some of the choices made for Waltz could become increasingly harder to maintain in the future. The version of JavascriptMVC used appears to be a fork of the main project taken in 2014. It also relies on a Java-based REST API component. Long-term support for the Java CORBA libraries that underpin the use of Java components in the TANGO infrastructure is far from clear. In contrast, WebJive uses a combination of JavaScript libraries (such as React) and Python libraries (such as Graphene) that are well established in other web applications. It combines these with TANGO components (such as PyTango and Taurus) that are widely used across a number of TANGO sites and projects.

Quality and Availability of Documentation

For a programmer reliable documentation is always a must. The presence of documentation helps keep track of all aspects of an application and it improves on the quality of a software product. Its main focuses are development, maintenance and knowledge transfer to other developers.

The Waltz framework provides a comprehensive set of user and developer documentation. WebJive provides a lot less. However, the usefulness of the Waltz documentation

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL005

is reduced by broken links and missing sections, which in the team ranking criteria were considered important, both in terms of learning, and in the discussions as an indicator of how ‘trustworthy’ the code was. Where WebJive scores highly is that because it is based on React and GraphQL, there are a wide range of external training and support materials available.

CONCLUSIONS

In the paper, the importance of the Graphical User Interface in a complex facility like SKA has been reported. Starting from the requirements derived during the Design Phases and Bridging Phase, two framework have been identified. Following the feedback from the SKA stakeholders during the decision process and defining a list of evaluation criteria, Webjive has been chosen as framework for SKA User Interface Generator. Following the User-Centered Design process, a selection a software between the two very valid candidates has been possible. Also if Waltz has a higher level of maturity than Webjive, thanks with the interaction with the SKA stakeholders during the decision making, the last one has been selected because has proven to adapt better the operators’ expectations and necessities.

Currently, the team is developing new features and making improvements in Webjive in order to reach a good level of maturity. Next step will be to analyze if, with the modification of the code, the framework can be definitively adopted as a possible candidate as the Graphical User Interface tool for the whole SKA.

REFERENCES

[1] Square Kilometre Array (SKA), https://www.skatelescope.org

[2] D. A. Norman and S. Draper, User-Centered System Design: New Perspectives on Human-Computer Interaction, CRC Press, 1986.

[3] V. Alberti *et al.*, “Usability recommendations for the SKA Control Room obtained by a User-Centred Design approach”, in *Proc. ICALEPCS 2017*, ES, Barcelona. doi:10.18429/JACoW-ICALEPCS2017-THAPL0

[4] Tango Controls, https://www.tango-controls.org

[5] Scaled Agile Framework (SAFe), https://www.scaledagileframework.com

[6] Waltz, https://waltz-docs.readthedocs.io/en/latest

[7] M. Canzari *et al.*, “A GUI propotype for SKA1 TM Services: compliance with user-centered design approach”, in *Proc. SPIE 2019 Software and Cyberinfrastructure for Astronomy* V, TX, Austin. doi:10.1117/12.2313276

[8] TANGO REST API, https://tango-rest-api.readthedocs.io/en/latest

[9] JavascriptMVC, https://github.com/jmvc-15x/javascriptmvc-1.5.x

[10] WebIX, http://webix.com

[11] ploty.js, https://plot.ly/javascript

[12] Webjive, https://webjive.readthedocs.io/en/latest

[13] React, https://reactjs.org

[14] GraphQL, https://graphql.org

[15] Redux, https://redux.js.org

[16] AIOhttp, https://aiohttp.readthedocs.io/en/stable

[17] Taurus, https://taurus-scada.org

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2019). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.