Satisfying wishes for SKA engineers: how Taranta Suite meets users' needs

Matteo Canzari[a], Valentina Alberti[b], Hélder Ribeiro[c], Ajaykumar Dubey[d], and Vincent Hardion[e]

[a]INAF, Osservatorio Astronomico d'Abruzzo, Teramo, Italy
[b]INAF - OATs, Trieste, Italy
[c]FCUP - Centro de Investigação em Ciências Geo-Espaciais (CICGE), Portugal
[d]Persistem System, Pune, India
[e]MaxIV Institute, Lund, Sweden

ABSTRACT

SKA Construction phase we are now in a transition phase that hopefully will prepare us for the next challenge: start building the SKA. One of the targets of this period is to evaluate the suitability of the Taranta (proper Webjive) Suite for creating engineering User Interfaces (UIs). The Taranta suite is a framework that allows the fast creation of web UIs that directly communicate with TANGO devices. What we need to address are the answers to questions such as: What kind of interface are you targeting? What are the performance constraints that you foresee? What are the current limitations of the tool that would make you choose a different one instead? What will the context of use be? What kind of features would you like the tool to have?

Keywords: SKA, Taranta, web, Tango, LeanUX

1. INTRODUCTION

The Square Kilometre Array (SKA)[1]as project is an international effort aimed to build the world's largest radio telescope. 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.

The project is currently approaching the Construction Phase, planned to start in the second quarter of 2021. In order to mitigate the risks raised in the Critical Design Review and to be prepared for the start building of the telescope, an early prototype development has been started. In this phase, a user interface toolset that can be issued to development teams to build engineering screens is required. Uniformity of user experience, linking systems developed by multiple consortia, reduction of up-front development costs, are quality attributes to take into account.

Taranta has been selected[1] a web-based User Interface Generator software to create engineering User Interfaces and is currently used by the teams working on SKA.

2. TARANTA

2.1 Taranta Architecture

Taranta (proper WebJive) is a MaxIV Institute[2] and SKA effort, supported by the TANGO Collaboration, to build a web application that allows a user to create a **graphical user interface** to interact with Tango devices. Tango Controls[3] is the control framework adopted by SKA for the whole telescope. It provides tools to view a tree of Tango devices, view and modify device properties, view and execute device commands, create dashboards for interacting with Tango devices.

Software and Cyberinfrastructure for Astronomy VI, edited by Juan C. Guzman, Jorge Ibsen, Proc. of SPIE
Vol. 11452, 114523D · © 2020 SPIE · CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2562585

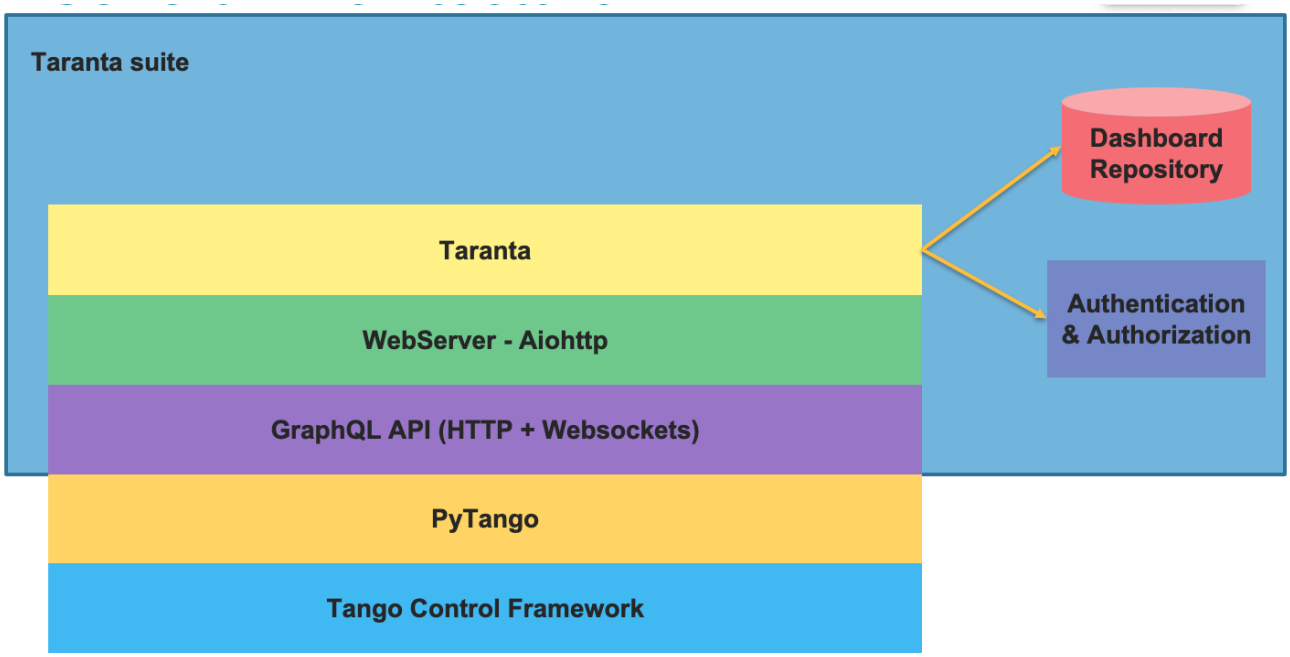

Figure 1. Overview of Taranta Architecture

Taranta's high-level software architecture is described in the following diagram.

Taranta suite is composed of **four software components** that permit the interaction with the Tango Control Framework through PyTango.

- Taranta: the frontend layer
- GraphQL API + Webserver: implement the backend layer which communicates with Tango
- Dashboard Repository: a software component which provides API to store and share user dashboard
- Authentication and Authorization: a software component that implements the login and logout functionalities

2.2 Taranta Suite frontend

Taranta has been developed using **modern programming patterns, languages, and technologies**. React[8] is an open-source javascript framework for building user interfaces or UI components. React is only concerned with rendering data to the DOM. The management of the states is entrusted to Redux,[10] which is a small library with a simple, limited API designed to be a predictable container for applications state. The overall application has been developed using Typescript, which extends Javascript and provides static type definitions.

Taranta[4] 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). Provides functionality to automatically detect inputs and has a search bar for devices. Furthermore, it arranges a generic implementation for all the devices.

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 and connect them to devices and

Further author information: (Send correspondence to A.A.A.)
A.A.A.: E-mail: matteo.canzari@inaf.it

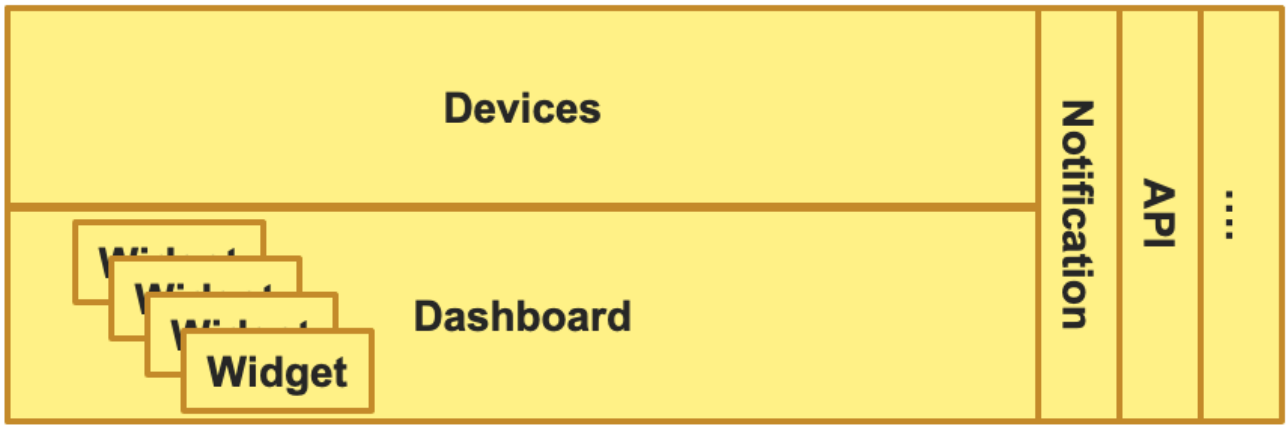

Figure 2. Taranta Frontend

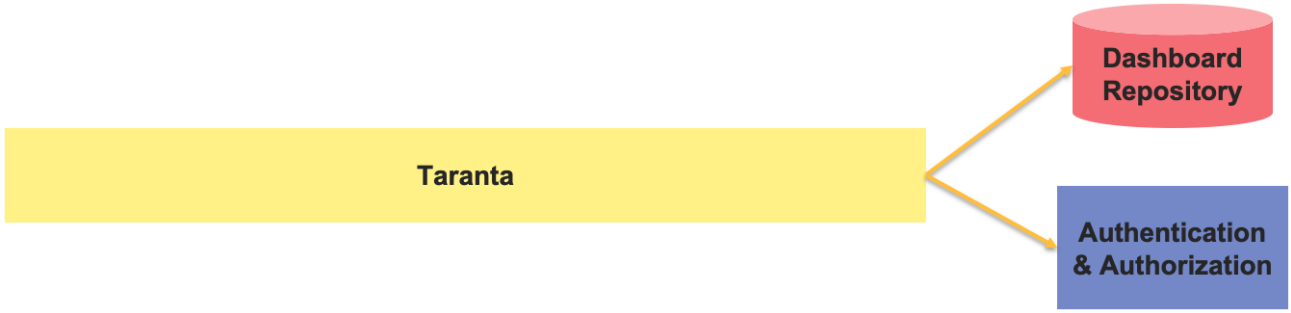

Figure 3. Taranta Services

own command or attribute. Once created, dashboards can be **run**, **exported** in a JSON-formatted file, in order to share screens with collaborators or the community. Each widget, which is a React component, has been developed using a Lean UI Approach described in the next section. In the current 1.0.5 version, there are 21 available widgets. A dashboard creation requires minimal knowledge of web technologies and no programming skills.

2.3 Taranta Suite Services

Across the two views, a series of libraries implement **common functionalities**. **Notifications** are addressed to a specif library that provides a unique way to push notifications and arranges a common space to show and manage the alerts occurred during operations, independent of the view where have been generated. Also, access to the data and **subscribing/unsubscribing** attributes libraries have been standardized using a common API that interacts with the backend layer. Furthermore, other libraries have been developed to improve the user experience, such as the **log** library.

Dashboards created with the Dashboard view are managed to a software component called Dashboard Repository.[5] In particular, it provides API to **store** and **share** dashboards with users and groups. The software component has been developed on MongoDB,[12] which is a cross-platform document-oriented NoSQL database. MongoDB uses JSON-like documents with optional schemas.

Finally, **login and logout** functionalities are ensured by the Authentication and Authorization[7] module, a service that permits retrieving user information from a third-part directory service (such as Active Directory) and persistent the session using JSON Web Token (JWT),[13] which is an Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims.

2.4 Taranta Suite Backend

Communication between Taranta, the frontend, and Tango is performed using a backend layer, called TangoGQL[?]

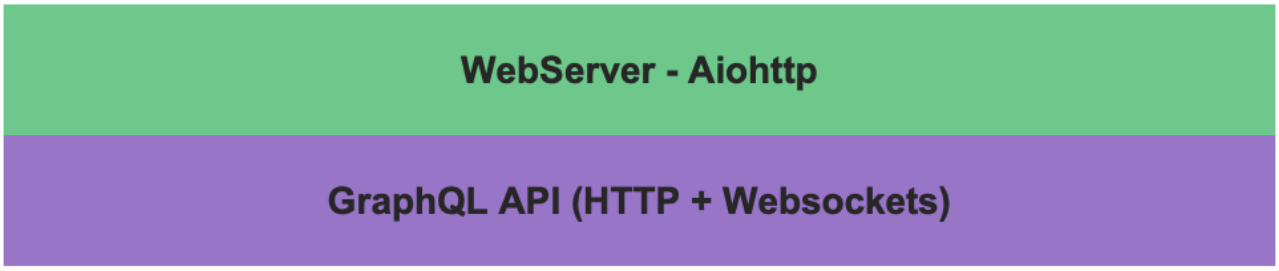

Figure 4. Taranta Backend

TangoGQL? is a library to access TANGO using GraphQL. GraphQL[9] is a new API standard that provides a more efficient, powerful, and flexible alternative to REST. It is a **query language** for APIs and a server-side runtime for executing queries by using a type system defined in the data. In general, REST is a fixed schema instead, GQL is a flexible schema. Furthermore, REST is based on multiple end-points, GQL on a single endpoint. To implement the backend layers modeling GraphQL application, the technologies involved are:

- graphene:[15] a library for building GraphQL APIs in Python
- aiohttp:[11] an asynchronous HTTP Client/Server
- Taurus:[14] an internal implementation using TaurusAttributes
- PyTango:[16] a python module that exposes to Python the complete Tango C++ API

real-time connection with the server, events from tango through **WebSockets**.

3. THE PROCESS TO COLLECT FEEDBACK FROM THE COMMUNITY

Taranta is a collaboration between MaxIV, SKA, and also the TANGO Collaboration. Even though cooperation across different Institutions increases the effort dedicated to the project, it is necessary to figure out **priorities** shared between the different stakeholders in order to focus the development following a common **roadmap**. The Taranta development process aims to take into account designing and developing User Interfaces focused on the users, that satisfy operators' skills and expectations. To achieve these goals and collect continuous feedback from users, a Lean UX approach has been adopted.

3.1 LeanUX approach

Lean UX[17] is focused on the **experience** under design and is less focused on deliverables than traditional UX. It is necessary to reach a good level of collaboration with the entire team to produce changes that improve the product in the here and now.

A crucial phase is retrieving **feedback** from the users. Different inputs can come from SKA engineers, MaxIV operators, or TANGO Community. During this phase, it takes place some discussion between Taranta stakeholders (such as the product manager, chief software from different Institutes), and the developer team to prioritize the request and plan the roadmap. The roadmap is composed of three plans: a **short-term plan**, tasks that should be developed in the next three months; a **medium-term plan**, tasks scheduled in the next six months, and the **long-term plan**, the expected tasks released in the next year. Furthermore, the roadmap is shared and discussed with the community during the **Tango status update meetings**, which usually occur every six months. In addition to the request from the community, further tasks are taken into account in the roadmap, such as improving the **test coverage**, reducing the **technical debt**, improving the **documentation**. Once created the roadmap, suggested by the whole Taranta Community and agreed with the Taranta stakeholders and the developer team, high priority features have been selected and analyzed.

A tool that permits the analysis of features with the LeanUX approach is the Lean UX Canvas.

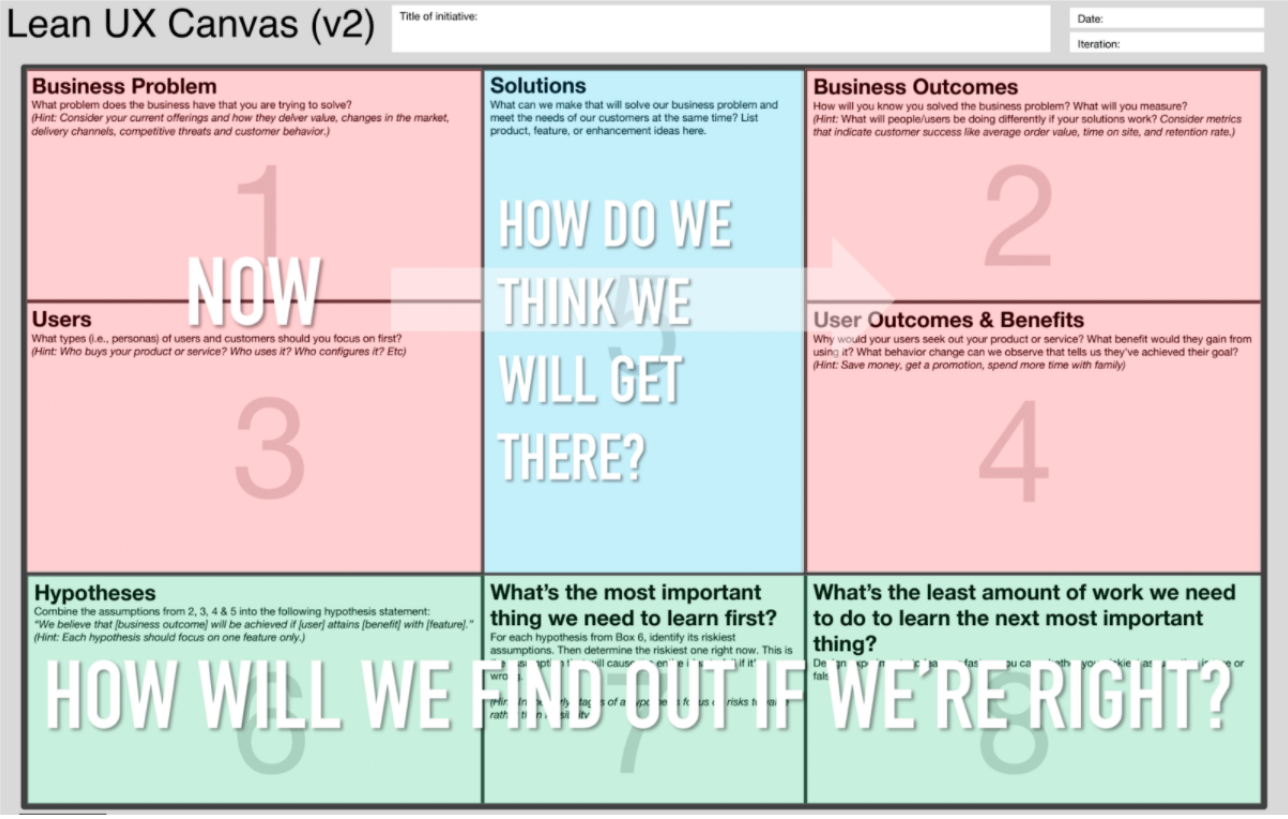

Figure 5. LeanUX Canvas

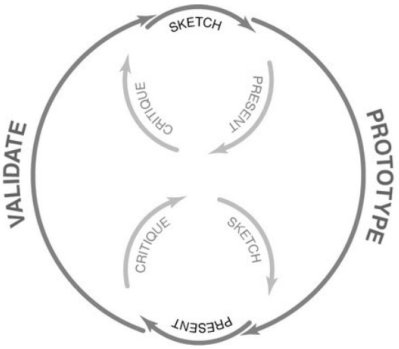

Figure 6. LeanUX Process

Lean UX Canvas is a tool created by UX consultant Jeff Gothelf,[18] that acts as a practical guide to both understand the current scenario of business and user needs. It allows teams to **focus** on business problems, business outcomes, user personas, and their benefits and convert these assumptions into the hypothesis. And finally, it guides to **design** research to test the most critical hypotheses.

After an analysis with the Lean UX Canvas, the outcome is a Taranta feature **well-defined** and ready to be developed by the developer team. During the development, the interaction between developers and users is maintained in order to constantly retrieve feedback. This phase is supposed to the Agile process the developer team has adopted. The **iterative review** process is also based on the Lean UX process, which requires an interaction between the developers and the user as in the figure listed below.

This kind of interaction guarantees that the final product satisfies the users' needs and requirements.

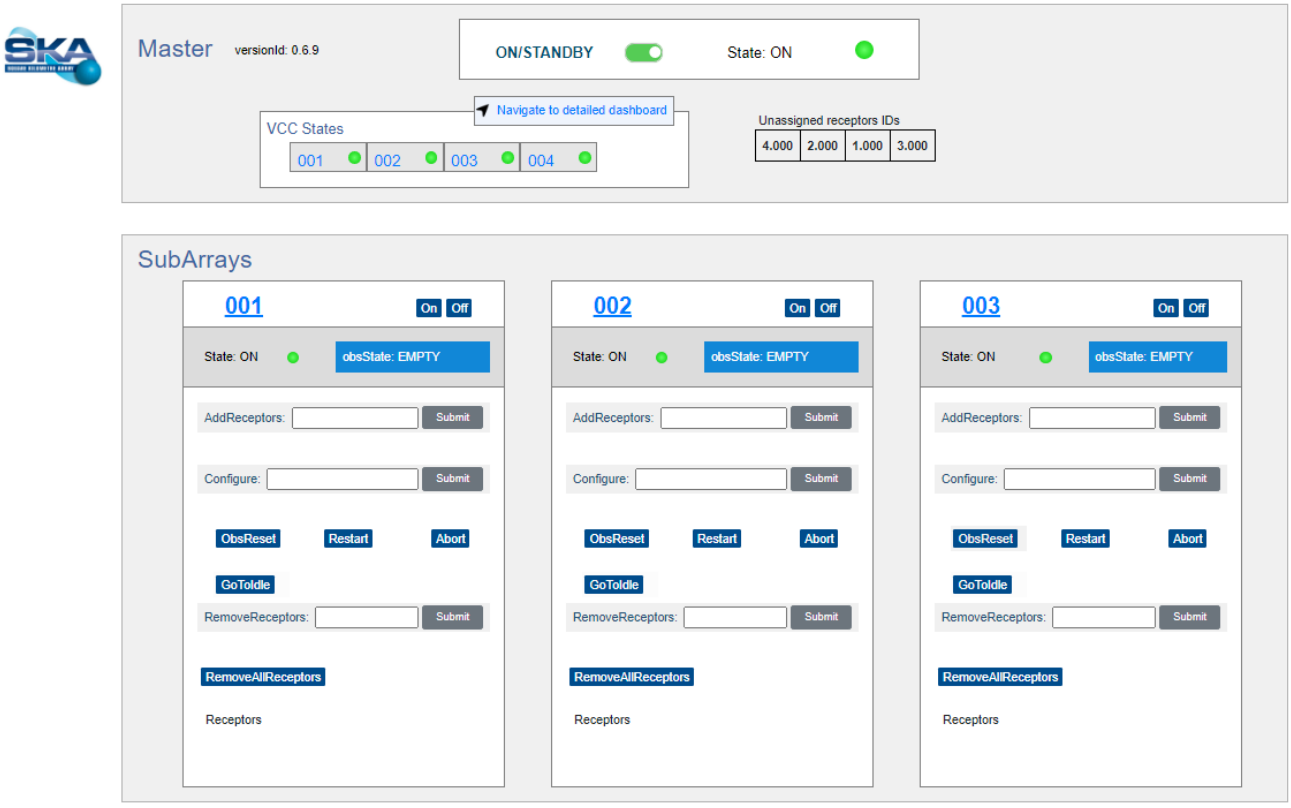

Figure 7. Taranta dashboard

4. TARANTA CURRENT STATE

In the current *1.0.5* Taranta version, there are 21 widgets available to the users in the dashboard view. Most of the widgets are **compliant** with the users' expectations from different Institutes. One widget, in particular the Command Switch, released in beta test in the *1.0.4* version, received negative feedback from MaxIV institute. As explained by the users, the switch between two commands doesn't reflect the switch between two states. On the other hand, SKA users agreed on the behavior of the widget. Further investigations are needed and a fix of the problem in the medium-term plan has been agreed upon. The *1.0.5* Taranta version has been tested by the SKA users, who build dashboards that support an engineer in monitoring and debugging real devices that the engineer is working on. A dashboard example is shown in the figure below.

Time spent creating this dashboard from an engineer is about 1 hour. From the feedback collected from the users, even though Taranta available widgets are enough to model a real TANGO device, some expected functionalities are still missing to have complete control of the control framework. These widgets have been analyzed and planned to be developed in the short term. The Taranta limits reported by users concern **performance** and **different behaviors** in different views. In particular, performance limitations come from the Dashboard view when specific widgets are running. The problem is caused by synchronous calls that slow the refresh of the dashboard. A problem fix has been figured out and planned to implement in the medium-long term plan. The problem related to the different widget behaviors is caused by a different implementation of some React components between the Dashboard and Devices view. This problem will be fixed in the short-term plan. Future feedback from the community has been expected in the months.

5. CONCLUSIONS

In this paper, the importance of user feedback has been discussed in order to achieve software which satisfies users' need and expectations and improve and facilitate the work of the engineer. A clear and shared feedback process

and roadmap are needed to involve the community in the decision process. Taranta permits to easily change and adapt the software component with what is requested from the users in an iterative manner. It is possible thanks to Taranta architecture based on microservice, that permits the creation of independent components, which changes don't affect the whole system. Furthermore, TangoGQL support changes thanks to its flexible way to retrieve data.

ACKNOWLEDGMENTS

This work has been made possible thanks to the financial support by the Italian Government (MEF - Ministero dell'Economia e delle Finanze, MIUR - Ministero dell'Istruzione, dell'Università e della Ricerca).

REFERENCES

[1] Canzari, M., Carlo, M. D., Khokhriakov, I., Poppi, S., Dolci, M., and Smareglia, R., "A GUI prototype for SKA1 TM services: compliance with user-centered design approach," in [*Software and Cyberinfrastructure for Astronomy V*], *Proceedings of the SPIE, Volume 10707, id. 107072P 11 pp. (2018).* **10707**, 107072P (June 2018).

REFERENCES

[1] Square Kilometre Array (SKA), https://www.skatelescope.org
[2] MaxIV Institute, https://www.maxiv.lu.se
[3] Tango Controls, https://www.tango-controls.org/
[4] Taranta Gitlab Repository, https://gitlab.com/MaxIV/webjive
[5] Taranta Dashboard Repository, https://gitlab.com/MaxIV/dashboard-repo
[6] Taranta Authentication Repository, https://gitlab.com/MaxIV/webjive-auth
[7] TangoGQL Repository, https://gitlab.com/MaxIV/web-maxiv-tangogql
[8] React, https://reactjs.org
[9] GraphQL, https://graphql.org
[10] Redux, https://redux.js.org
[11] AIOhttp, https://aiohttp.readthedocs.io/en/stable
[12] MongoDB, https://www.mongodb.com/
[13] JSON Web Token (JWT), https://jwt.io/introduction/
[14] Taurus, https://taurus-scada.org
[15] Graphene, https://graphene-python.org/
[16] PyTango, https://pytango.readthedocs.io/
[17] Lean UX Approach, https://www.interaction-design.org/literature/article/a-simple-introduction-to-lean-ux
[18] LeanUX Canvas, https://jeffgothelf.com/blog/leanuxcanvas-v2/