SP-2359 Summary of the performed investigations

Cream Team, May 2022

Introduction

The purpose of this document is to summarise the process and results of the study performed by the team in the context of SP-2359 with the goal of delivering the value as described in SPO-1675.

To gather the information and validate the created mockups the team followed the process described in the following.

Process: we run 2 cycles of interviews with expert users who worked at or are currently working at other radio telescopes. The first interview was aimed to gather the information needed to create mock-ups of the various widgets and to get the context of use while the second one aimed to validate the presented mock-ups. The UX techniques followed in these two stages are respectively a semi-structured interview and informal user testing.
The team interviewed 6 people in total, three of them have been involved in the first interview while the other 3 performed the validation of the mock-ups.

Tabular info

Document link: Google Drive

Widget description and context of use:

This widget is used to arrange the information in rows and columns. The major strength of a tabular representation of data is the ability to show large amounts of data in a compact way, thereby simplifying the process of connecting and interpreting (i.e. when a colour code is

associated with it) the data.

The data shown could include key state indicators from tango devices (See for example Figure 2 - Figure 6) or health information of subsystems as part of the same component or relevant metadata to describe the activities related to job cards[1]. For example, while reporting on maintenance activities a table could show for each maintenance job card the status of the work, who performed it and other relevant info. Another concrete example is the following one: each row could represent a dish while subcomponents or subsystems, such as the frequency band receiver etc., are represented as columns. Each cell could then give information about the overall health state of the subsystem. Depending on the type of data in the table, the ability to select a certain range and display the data on a plot or pie chart or histogram has also been mentioned as useful.

To reduce the amount of information displayed and avoid overwhelming the user, filters should be considered (i.e. the user may be interested in visualising only the rows with faulty elements).

Another important feature would be the ability to click on a specific cell and be redirected to a different dashboard or panel where more details are given or to a jira ticket linked to a job card or to the ticketing system to report a new problem (for faulty conditions). In this last example, some metadata could be added to the ticket automatically, i.e. the subsystem that is failing.

In order to allow for more space in complex dashboards, the ability to expand and collapse tables has been explored. It is deemed useful by the user. A dedicated investigation has been carried out on the general behaviour and is explained in the “Active Widget” section of this document.

The mockups in Figure 2-6 also show the download button. The user found it useful to be allowed to download and save data locally in order to review the information later on.

Other mechanisms like sorting and pagination are not deemed important at this stage. The only mentioned application is to sort rows in alphabetical order but it doesn’t seem to be a priority.

This widget would be used very often.

Widget mockup:

[1] A job card is a ticket created to track maintenance activities in the EMS (Engineering Management system)

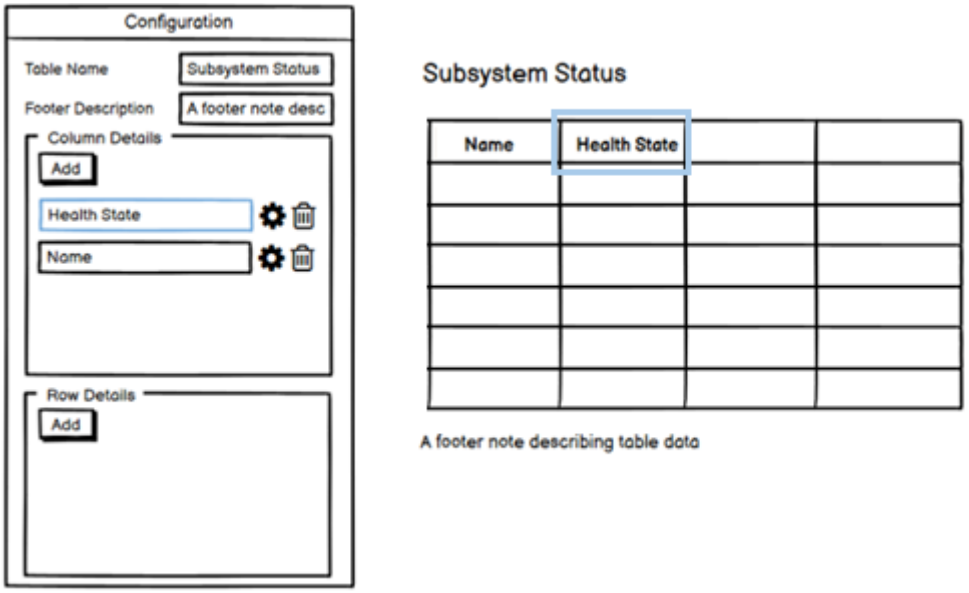


Figure 1: Taranta column setup: By clicking on the "Add" button users can add columns to the table, column name being an attribute of the device. The link to the specific device happens while configuring rows.

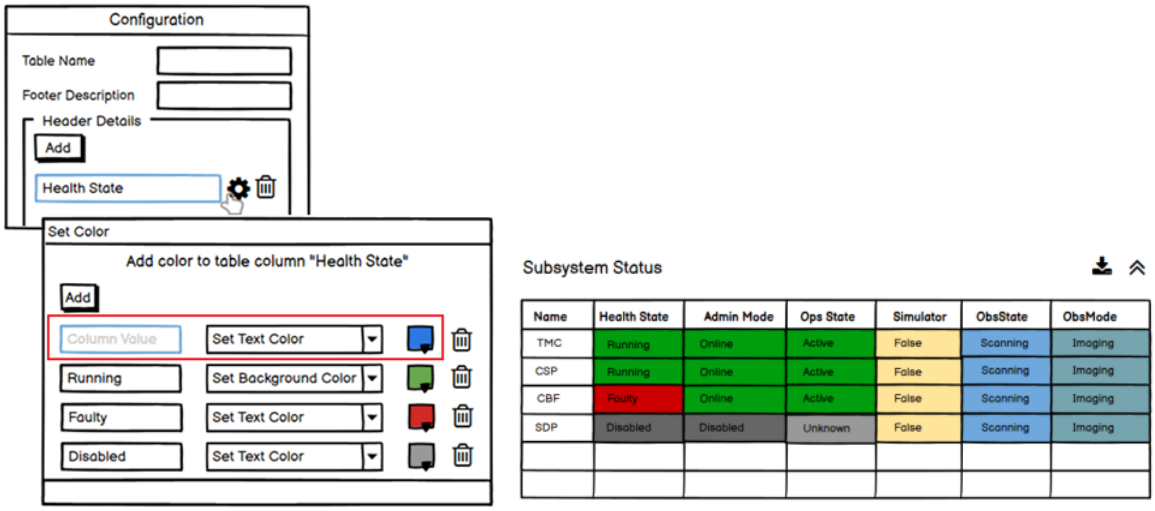


Figure 2: *(Left)* Setting colours for a specific condition. The user can configure a background colour when a specific condition (i.e. a value of the state attribute) is verified. This will help to highlight important information and improve usability of the UI by speeding up the identification process. *(Right)* The final result after configuration. Icons showing the download and collapse/expand functionalities are visible as well.

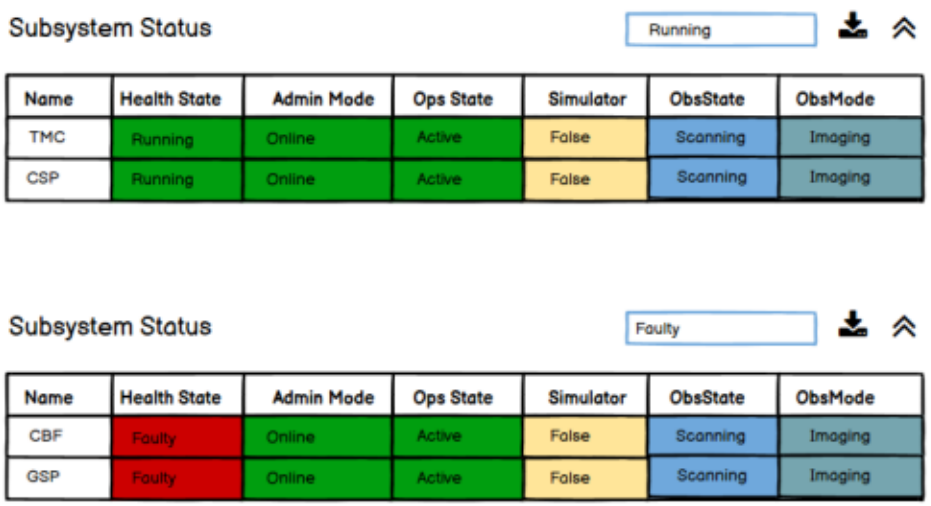


Figure 3: Table with search filter: Search filter applies on all columns in the table, in figure above only rows that contain “Fault” states are shown.

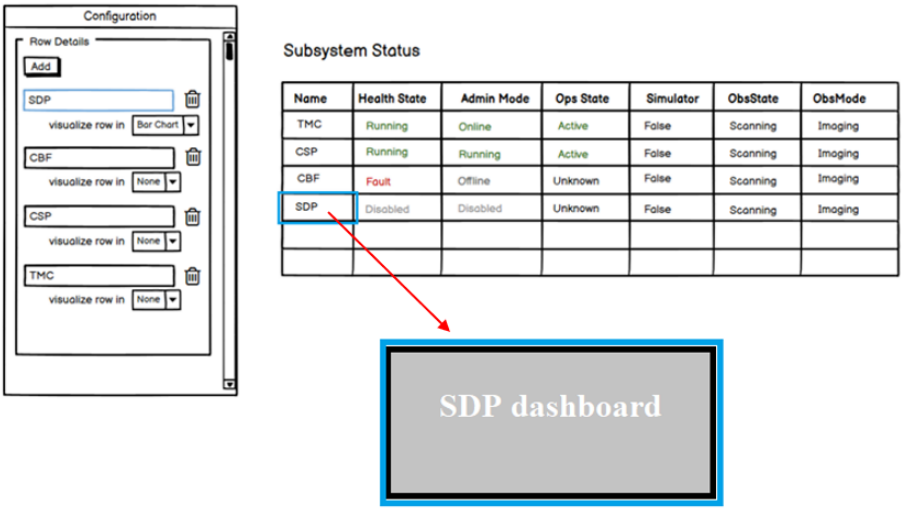


Figure 4: The user can click on a specific cell to retrieve more details.

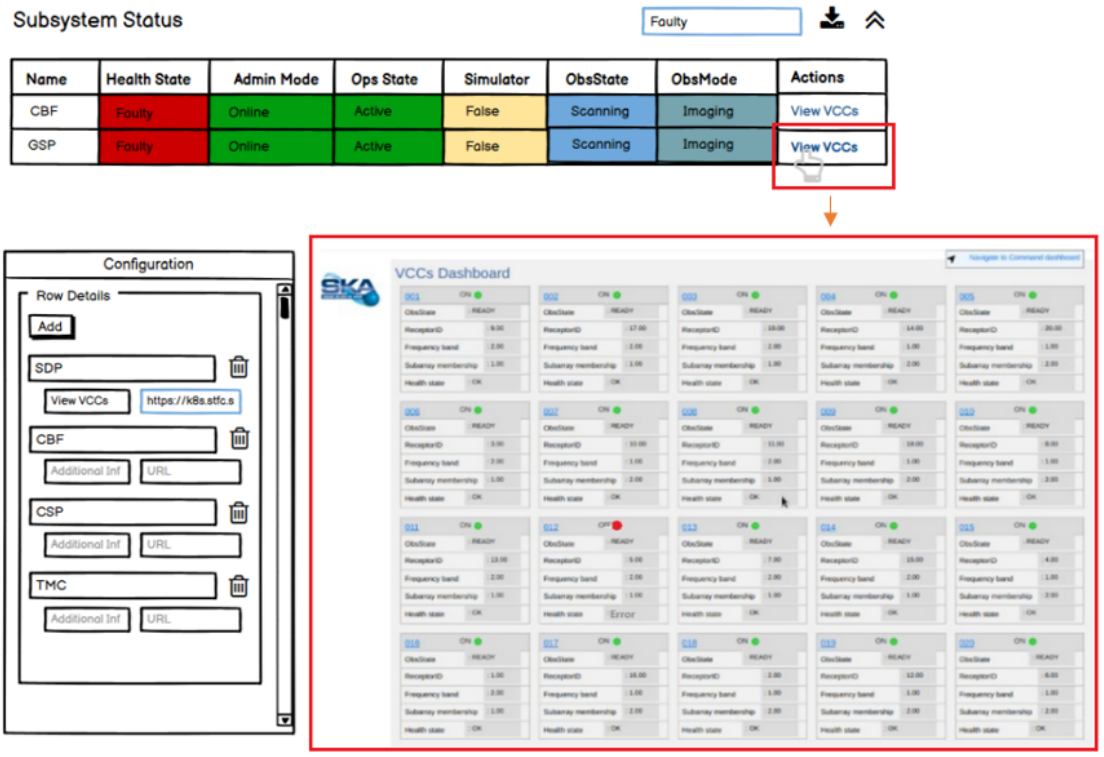


Figure 5: In case additional information is required, the user can configure an action column that allows them to link i.e. a dashboard for a specific component (VCCs in the image). The top panel shows the configured table with the "action" column, the bottom part shows the configuration panel (next) and the linked dashboard (right, VCCs dashboard). The action (opening the dashboard) is triggered by clicking the link.

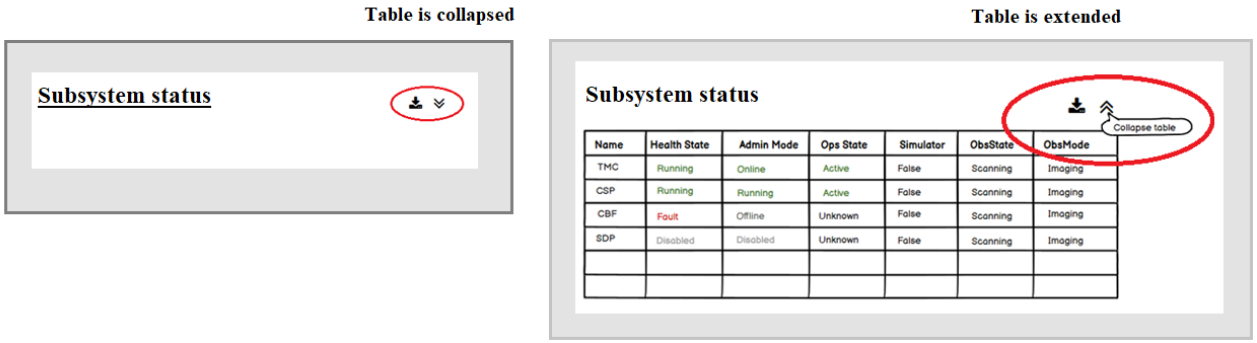


Figure 6: Table with tooltip, download and collapse option. *(Left)* Collapsed table with download and expand icons and functionalities. *(Right)* Clicking on the double arrow-down icon the user can expand the table. The option of showing a tooltip (figure below) has also been explored but is not deemed mandatory.

Time estimation: approximately **4/5 FP**.

Tabbed dashboard

Document link Google Drive

Widget description and context of use:

This widget is used to display data through a hierarchy system which is provided by tabs. Each tab is identified by the tab title and gives access to specific information by clicking the title area. The title should make the context of the data in it explicit.

The major strength of the tabbed dashboard is to allow the user to organise a large quantity of information without cluttering the screen and saving display space. It also simplifies the navigation because it doesn't require the user to open a separate dashboard but only needs to click on the desired tab name.

Such a way of data presentation simplifies the process of understanding the data structure and the inner data connections.

Tabbed dashboards won't be used when summary displays are needed, for example, to show the overall status of the system at a glance. In this case the usefulness of tabs is low and clever ways to condense a lot of information through meaningful parameters and effective representations should rather be considered.

A concrete example that applies the hierarchy in Figure 7 is a dashboard that allows monitoring and control of the CSP from the main tab and its subsystems (CBF, PSS and PST) from the relevant tabs.

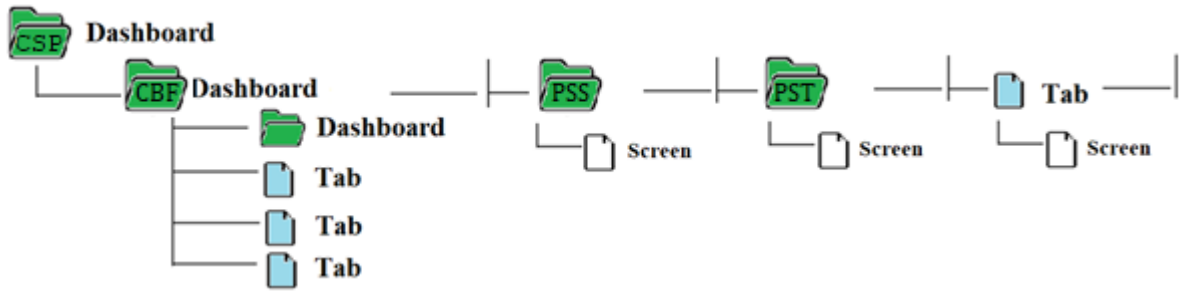


Figure 7: Example of the tabbed dashboard showing detailed hierarchy of the data and based on data linking.

The following example shows the hierarchy of the data based on their linking.

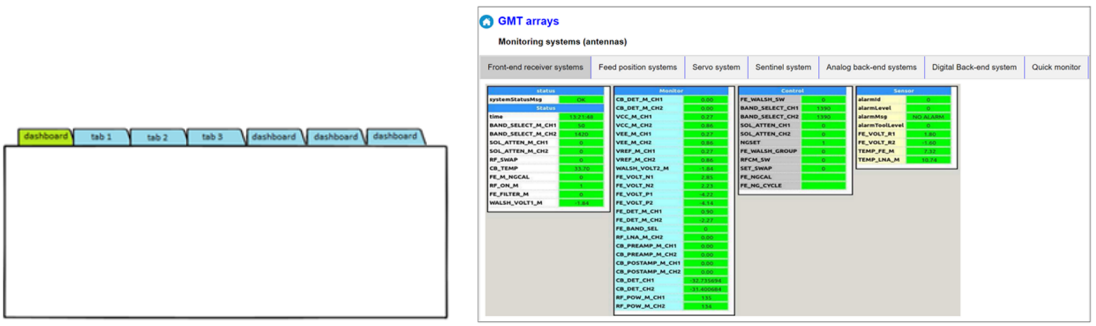


Figure 7.1 (*Left*) - full tabs` dashboard. (*Right*) - full tabs dashboard with real data. Each tab represents a subsystem and shows information on its status, monitoring parameters, and

data from its sensors.

The data shown could include monitoring and controlling complex systems of devices (See **Figure 7.1**, image on the right). Can present general data, such as SKA system`s overview (for example, for Telescope overview data).

It is considered a must-have feature by users.

An alternative way of displaying data in tabbed dashboard format, is to display it as a separate container of tabs. Such a dashboard can be called a tabbed *widget.* It can take only about 50-70% of the screen and lets locate (next to the tabbed dashboard) a separate set of the relevant data , like data visualisation, for example (see Figure 8).

Such layout could be used, for example, for CSP, Subarray's overview dashboards.

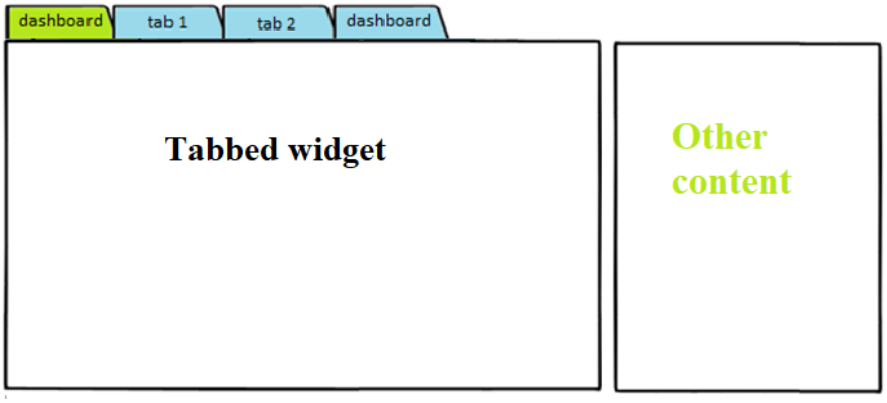


Figure 8: Tabs` widget as a separate (stand alone) container of tabs (template).

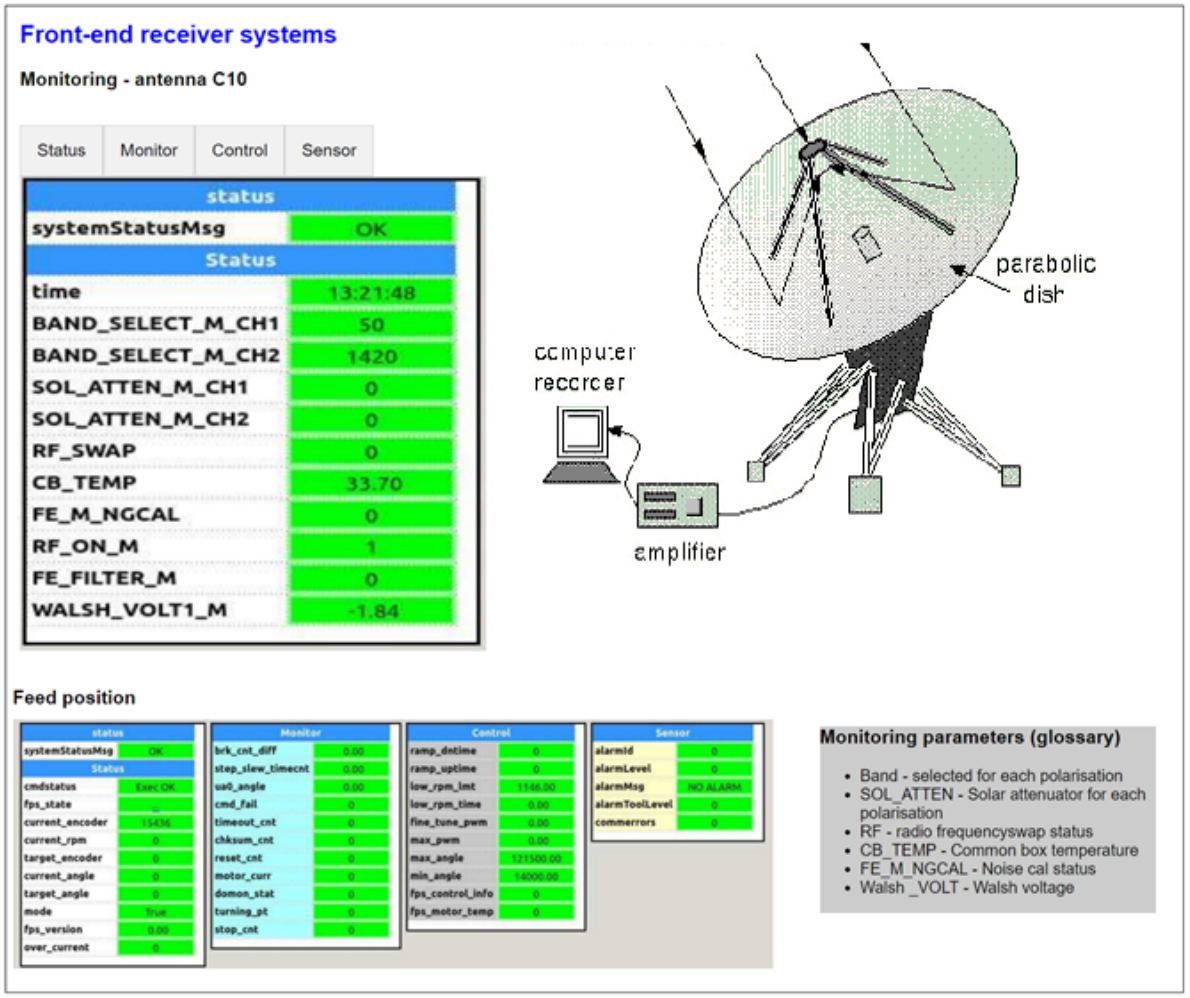


Figure 9: ***(Top-Left)*** Tabbed widget, based on some examples from other facilities as provided by the interviewee. The layout in the image shows the tabbed widget on the left, providing relevant monitoring information, and additional info like graphics displayed on the right and bottom side of the screen.

Each tab can contain another dashboard to let users navigate between related screens using the same browser window (see Figure 8). The surrounding content will stay the same in the basic implementation but could change to provide tuned contextual information if required and helpful to the user. In this second case, the displayed data will be relevant to the content in the left part of the screen. This second option needs further investigation.

The major strength of tabbed widget is to give the user the ability to retain contextual information while looking at the system from a different perspective**.**

The tabbed widget can be configured with the following properties:

- number of tabs
- tabs` names
- relevant sub-dashboards.

The navigation between tabs could support the following methods:

- on mouse-click
- by keyboard` keys one tab (tab, shift-tab).

Another optional useful layout of the tabbed dashboard is a vertically tabbed dashboard where some tabs are brought outside of the main container. This will reduce the vertical space needed.

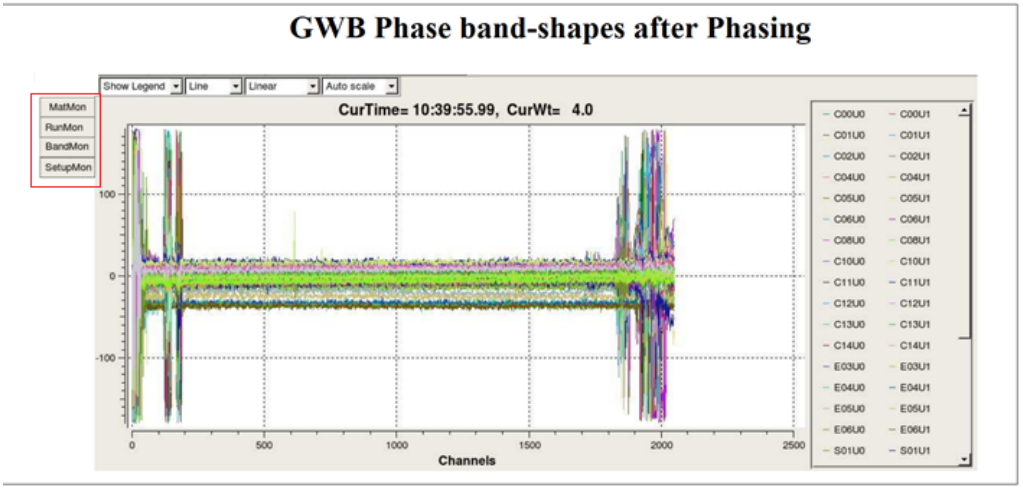


Figure 10: Tabbed dashboard with vertical stacked tabs outside main frame as template based on the real data.

Overall, tabbed dashboards will be needed in the future but aren't crucial at the current stage because the system isn't too complex yet.

Time estimation:
It is a summary of the efforts, given to build an active dashboard (depends on complexity of data to be loaded).
Approximately: **3 FP.**

Active widget

Document link: Google Drive

Widget description and context of use:
Active widget is also sometimes called collapsible widget. This widget paires a header to a content, and allows the user to cycle through collapsed or expanded in a header click. Unlike the accordion widget which is initialised for a set of title/contents pairs, the collapsible widget is initialised for one title/content pair. Once it is expanded, the content can be updated, visualised and interacted with normally. The collapsed/expanded state can be saved into local storage, or cookies if the browser does not support local storage.

Using the feedback from interviews it is clear that active widget is a must have widget that will be used all across SKA operators, dashboard engineers and maintainers.

Interviewees reported that this functionality should be present on every widget, being very useful to save dashboard space, and to better focus user attention on specific widgets, basic functionality schema can be seen in Figure 11. When new data is available on a collapsed widget, the icon should blink every X seconds to let the user know Figure 12.

Interviewees asked for the option to warn that widget collapsed content could be in warning state, and users should see a warning icon to request their attention Figure 13. Once the user acknowledges this and clicks either the expand icon or the warning icon, warning should disappear Figure 14.

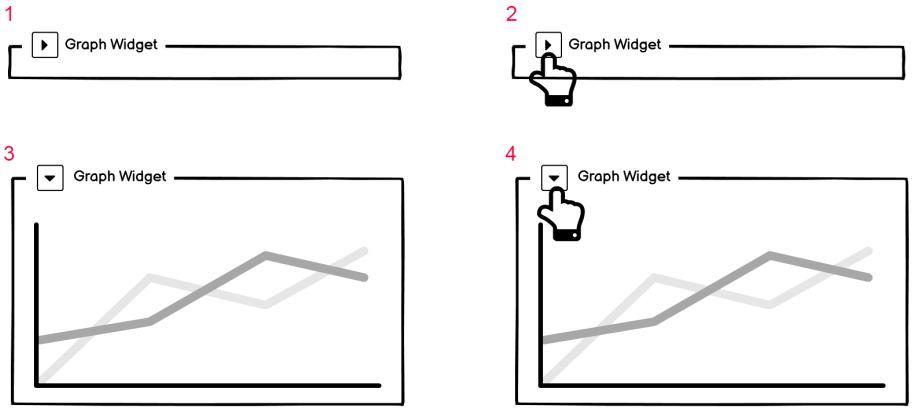


Figure 11 : Active widget basic functionality of expanding and collapsing its content.

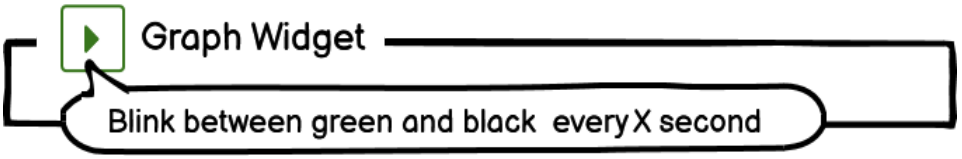


Figure 12 : Active widget with new data available, button should blink every X seconds to get the user's attention.

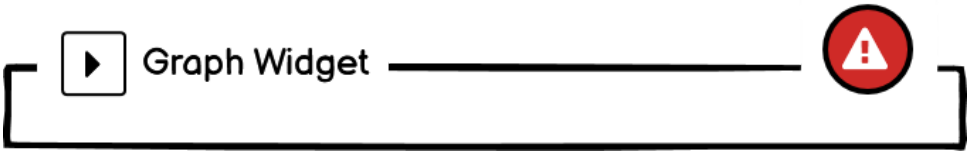


Figure 13 : Active widget with alarm state, warns the user that widget content is in alarm state.

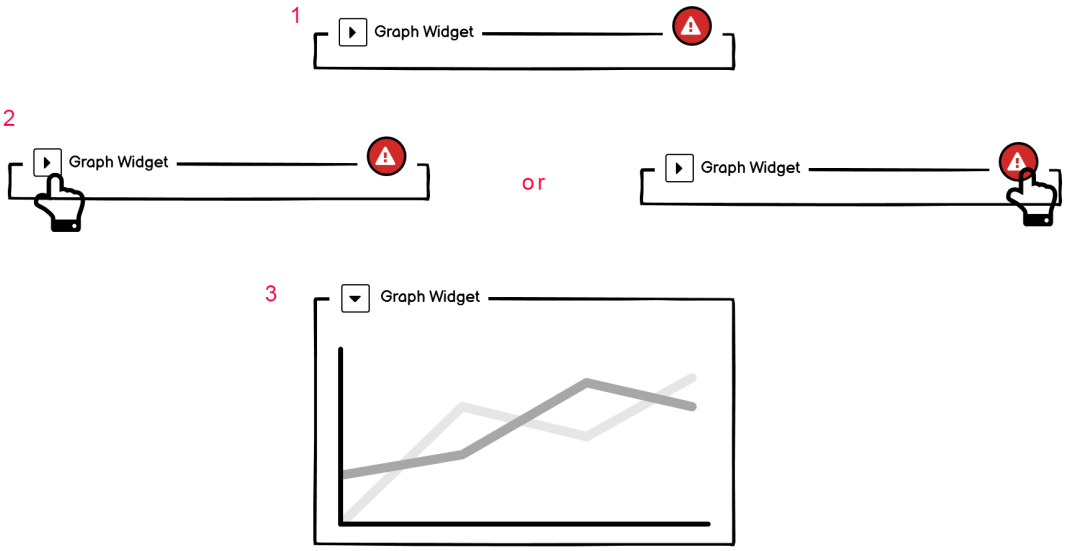


Figure 14 : Active widget alarm state inspection diagram.

Time estimation:

- As a **single widget feature**, something we apply for example to graph widget, and give the ability to collapse and expand - in that case each widget can take up to **0.5 FP** to fully implement all functionality.
- As a library to apply to **all widgets at once -** in that case **5 FP** should be needed.

Bar chart widget

Document link: Google Drive

Widget description and context of use:

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally. A vertical bar chart is sometimes called a column chart. A bar graph shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value. Some bar graphs present bars clustered in groups of more than one, showing the values of more than one measured variable.

Using the feedback from interviews it is clear that bar chart widget is a must have widget that will be used all across SKA operators, engineers and maintainers.

The main data examples that were given to use on this widget were failures and warnings related data. This type of widget will allow SKA users to better visualise failures for examples on sub arrays during weeks or months, aggregating all this data in a single bar chart, giving a more visually appealing feedback Figure 15.

One concrete example was given where the user can visualise the number of failures related to multiple sub arrays at once, and can even give the user the ability to filter between some sub arrays Figure 16.

Another example was given when the data has a lot of entries to the axis, for example visualising all the warnings (day by day) that occurred during April on sub array 1 Figure 17. As a final suggestion users said if the data to display is too big a vertical bar chart should be used like we can see on Figure 18.

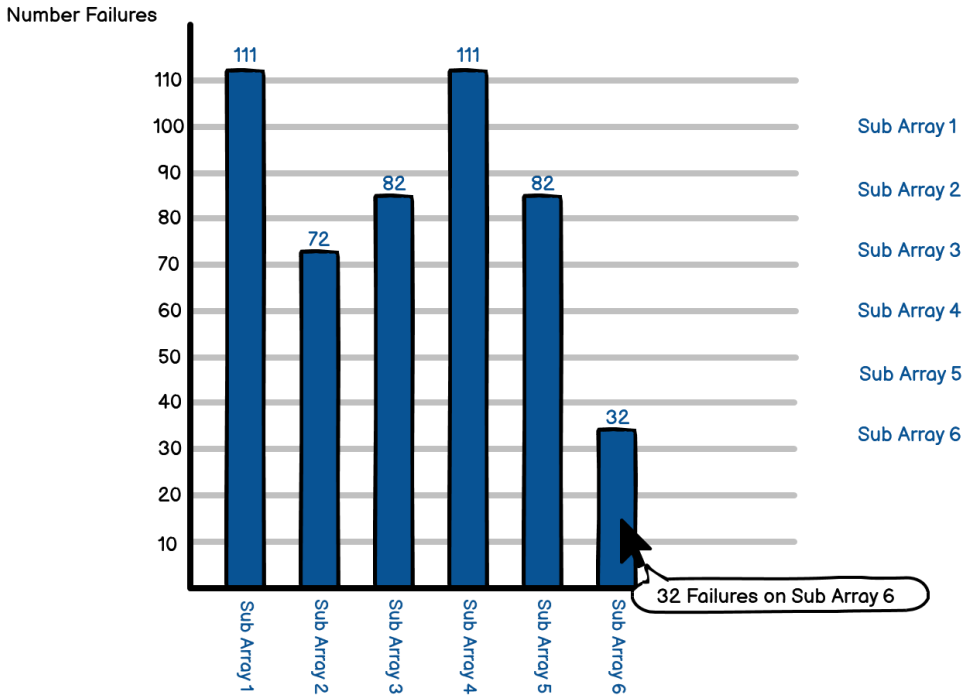


Figure 15 : Bar Chart

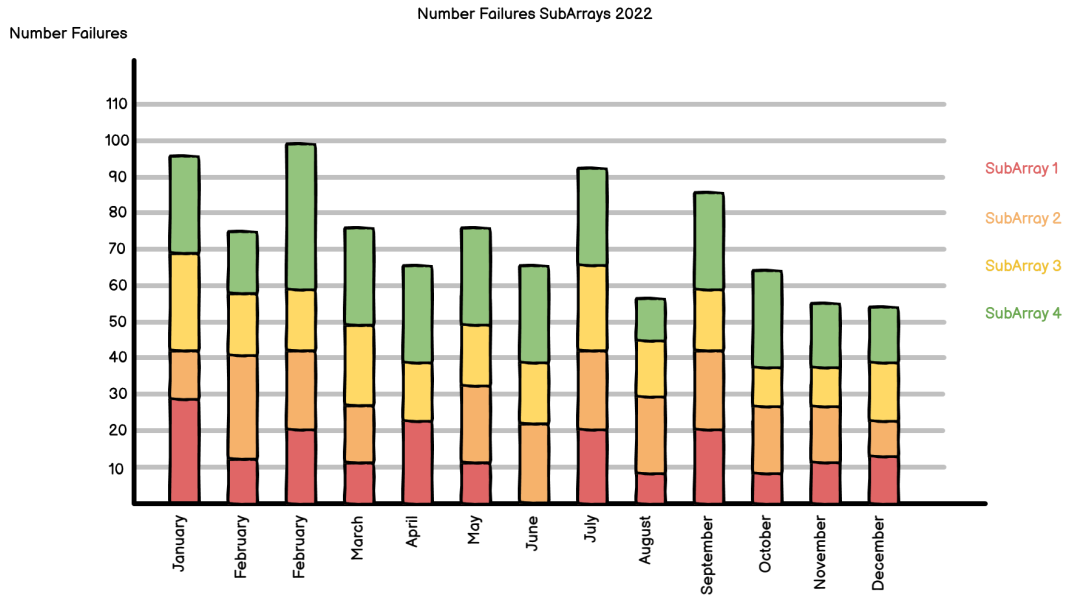


Figure 16 : Stacked bar chart showing multiple sub arrays 1 to 6 failures from Jan to Dec.

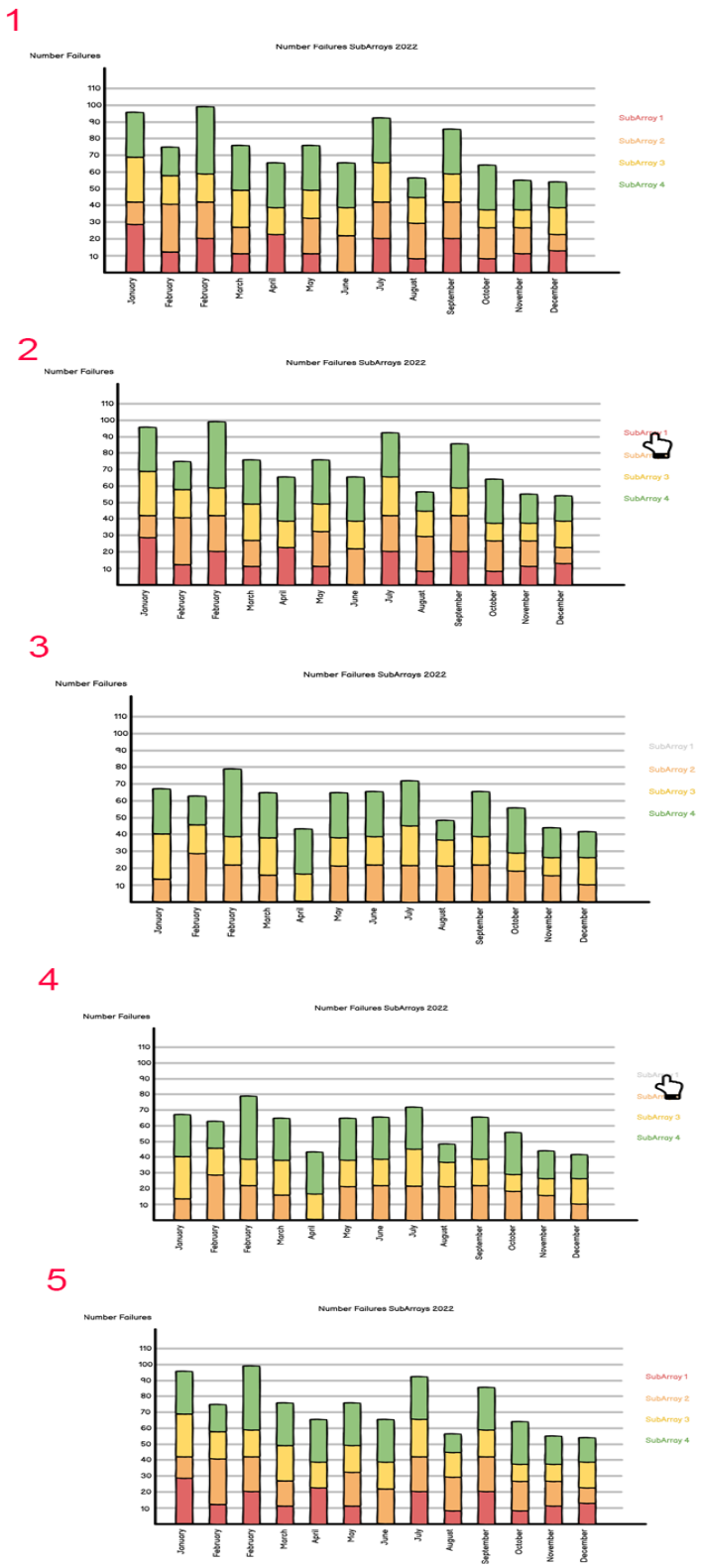


Figure 17 : Stacked bar chart showing how a user can filter the displayed sub arrays data.

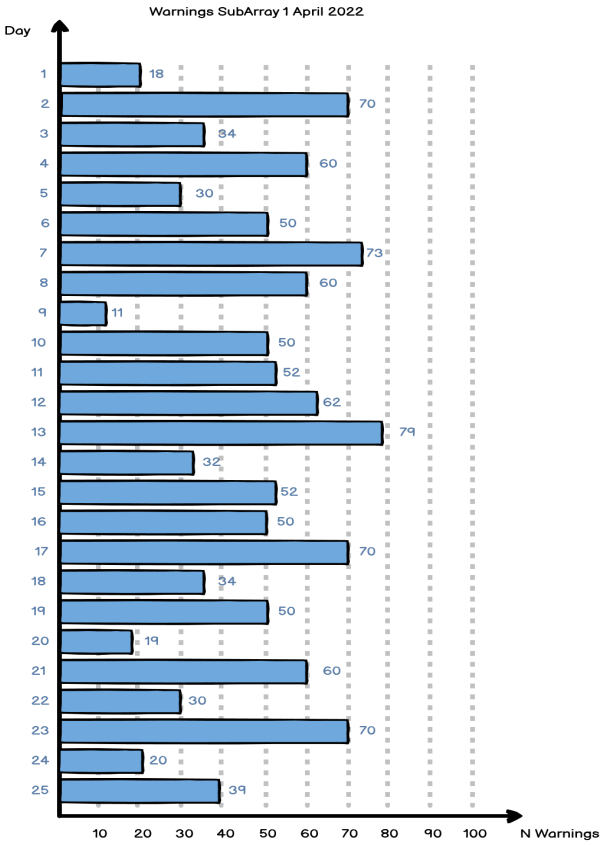


Figure 18 : Horizontal bar chart to display a bigger set of data on the axis.

Time estimation:

Taking into account we got a feedback without any special requests, we can fore see the use of a well know library already used in taranta (Plotly). So our esteem 2 FP should be enough for this widget. There are a lot of configurations possibilities already present that could enable the user to configure the widget in all the ways shown in the mockups.

Starburst diagram

Document link: Google Drive

Widget description

A starburst diagram is used to visualise hierarchical data, depicted by concentric circles. The circle in the centre represents the root node, with the hierarchy moving outward from the centre.

A starburst chart without any hierarchical data (one level of categories), looks similar to a doughnut chart. However, a starburst chart with multiple levels of categories shows how the outer rings relate to the inner rings.

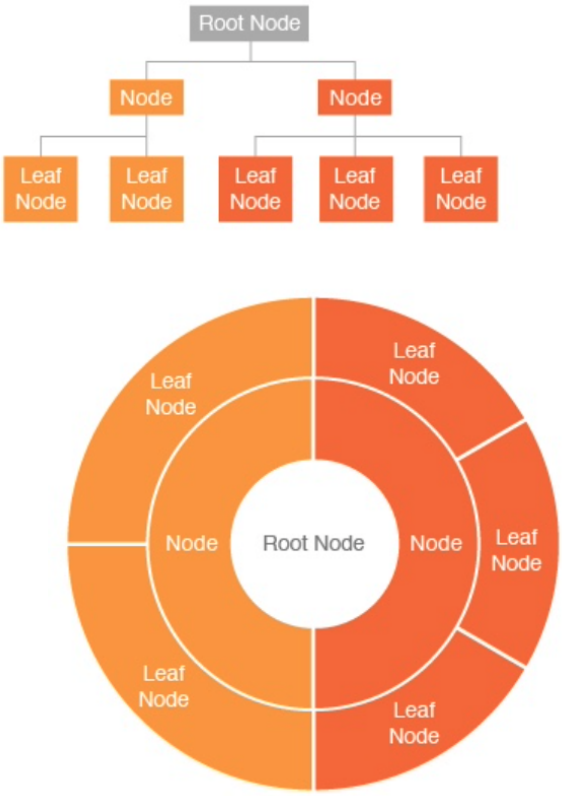


Figure 19 : Starburst diagram data structure

The usage of a starburst diagram is common in data representation activities. Following the suggestion based on telescope operator experience during the interview, the difference between a common usage chart and a view in a telescope control room is the structure of data. Usually, this kind of diagram represents an homogeneous structure of data, for example the population, where the root node represents the total amount of the population and leaf nodes represent the population of a specific region. In the SKA use case, it should be useful that each circle represents a level of device to monitor. For example, in the monitoring of an antenna health status activity, it is suggested that the inner circle represents the status of the subsystems that compose the antenna, and the outer circles represent the value of the components that belong and affect the subsystem.

The diagram is useful mainly for telescope operators, but also for Science operation, Engineering operation, Maintainers. In general, operators can use it 24 hours a day. Other users monthly or weekly.

Operator can use the starburst diagram for a health status view. Maintainers, Engineers and so on to investigate failures.

The priority of the widget is want-to-have.

Other functionalities requested are:

- Circular sectors can be clicked (node becomes the centre of the diagram and new circular layers will be drawn with node children)

● Activating clickable circular sectors will allow drill-down and drill-up navigation on the tree

● Drill-out function can be performed with a mouse/ keyboard`s key.

Widget mockup

Configuration

To configure the diagram, it is necessary to create a tree structure. The mockups in Figure 20, 21, 22 represent creating a root-node-child connection. It is important to note that the value of a single node comes from the Engineer Data Archive or from the control system. The user can customise every node using a colour or modifying the colour based on the condition defined.

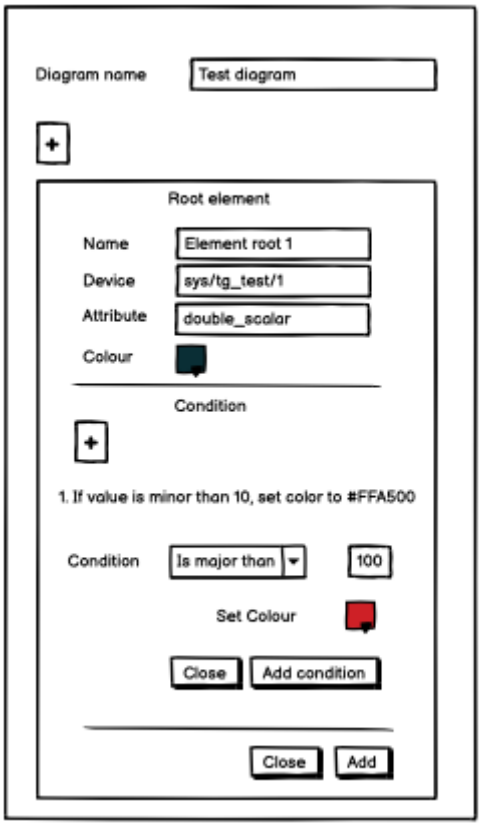


Figure 20 : Creation of a root node with 2 condition associated

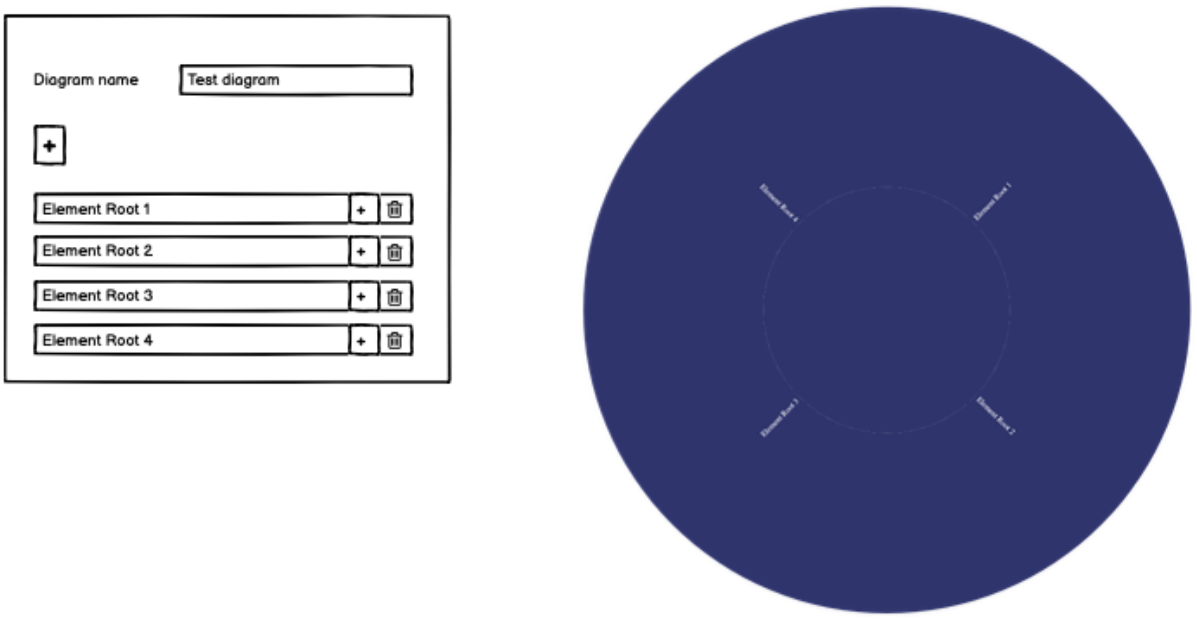


Figure 21 A diagram with 4 root elements. It is basically a doughnut chart

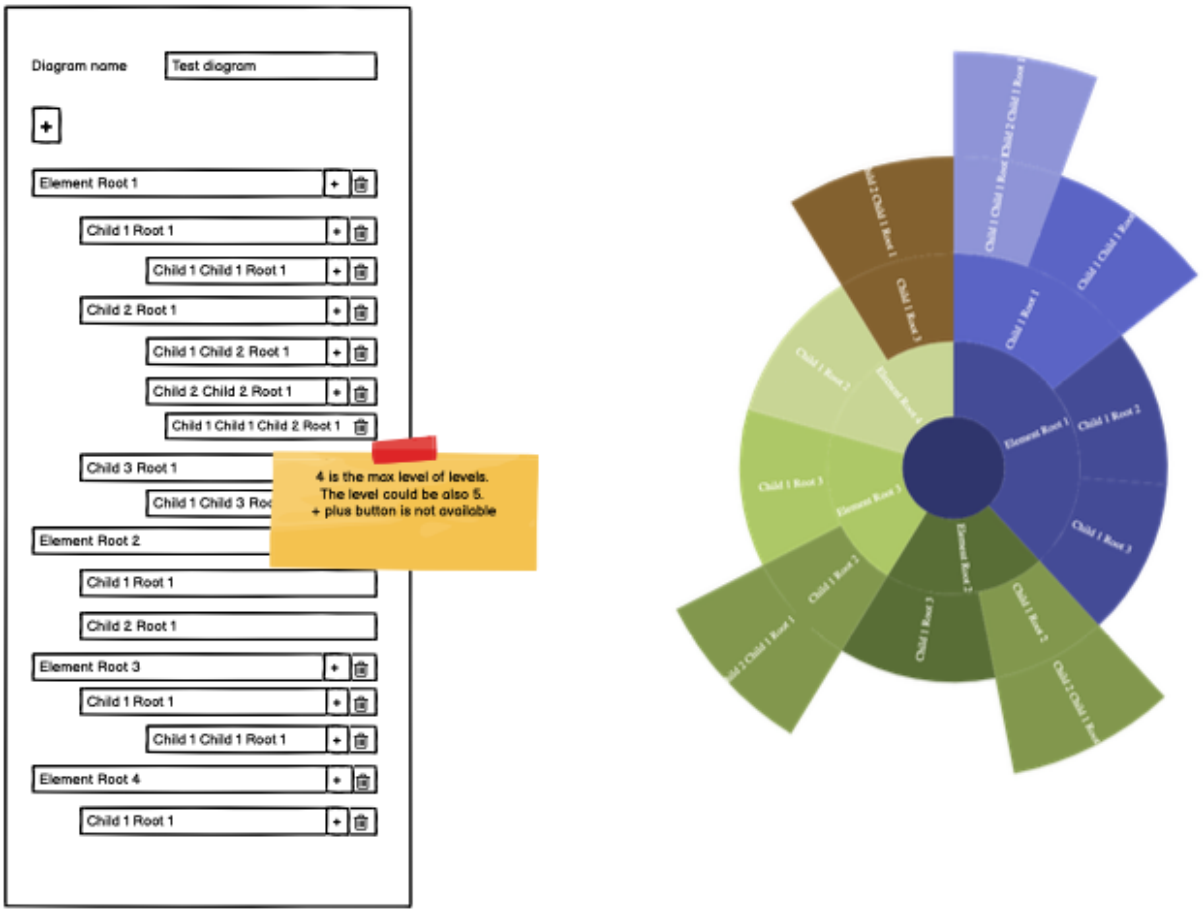


Figure 22 A diagram with 4 root elements and the childs associated

Real use case

The mockup in figure 23 represents a “realistic” use case of a monitoring antenna health stats. In the chart, the inner circles represent the overall status of the subsystem. The outer circles, or leaf nodes, represent the values that affect the status of the subsystem. In this

example, we can see how the value matches the condition. The chart changes the colours of the different elements. Also the background changes.

Figure 23 a “realistic” use case of a monitoring antenna health stats

Scalability

The solution proposed in figure 24 is based on the aggregation of different diagrams. Each diagram is represented by a status that changes the colour based on condition. The operator explores the diagram(s) aggregated clicking on a single “led”.

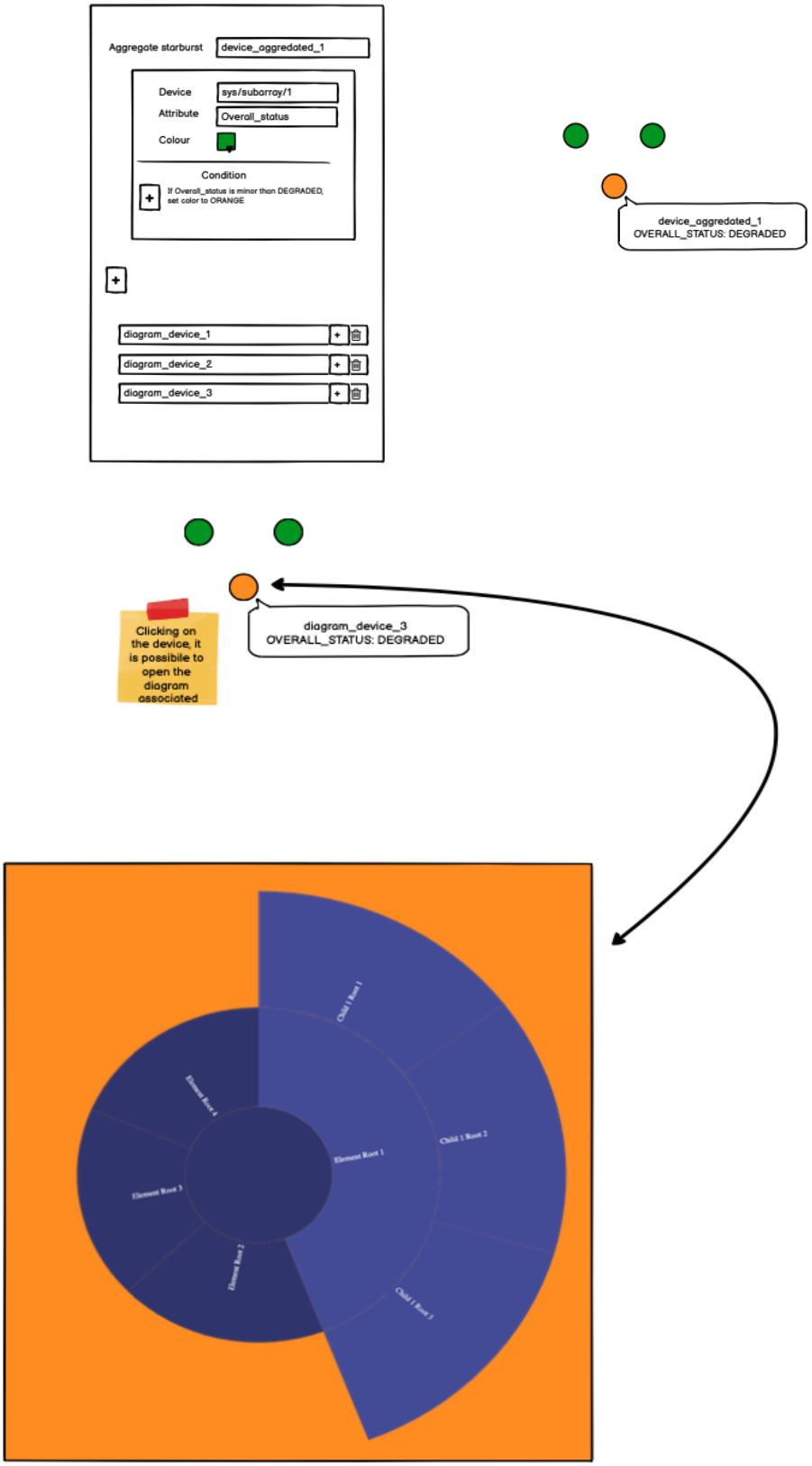


Figure 24 aggregation of different diagrams

It is also possible to aggregate recursively a set of aggregated diagrams, as represented in Figure 25.

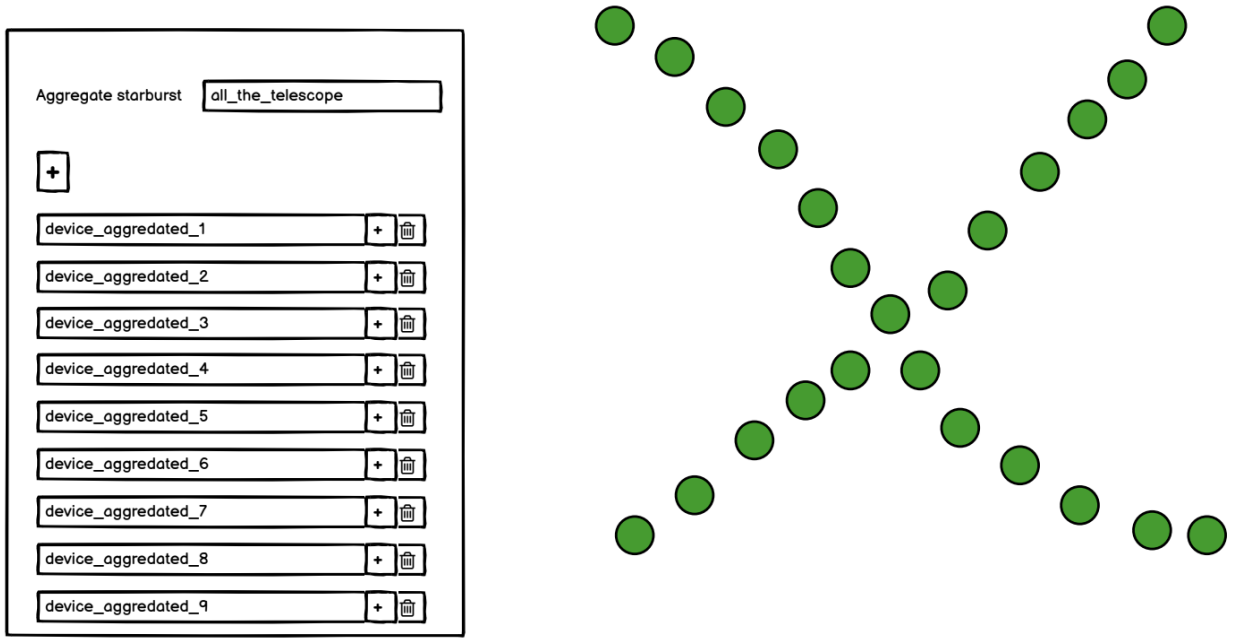


Figure 25 *An example of a complex configuration with more plot aggregated*

Time estimation
The correct estimation of the widget requires a deeper understanding. In general, we can split the development in two parts:

- If plotly library permits to render different set of data in a single diagram, ~~permits~~ to customise colours and add colours based on condition, it will be enough the creation of a generic plot using existing Plotly library: **1 FP**
- In case of Plotly library is not suitable, it is needed to study and analyse a new library and implement the basic functionality with it: **2-3 FP**
- Configuration. **3-4 FP**
- Scalability solution has not been evaluated because it needs further discussion.

Active dashboard

Document link: Google Drive

Widget description and context of use:
The active dashboard feature is helpful when the user is working with a tabbed dashboard. It allows to configure different tabs and to make them active/visible based on some condition or event[2]. The predefined mechanism to move between tabs is a manual click on the one of interest. More sophisticated mechanisms, such as cyclic dashboards, can also be introduced. For example, users could configure each of these dashboards to be active for a specific time interval or they could include them in a loop that switches to the next tab/dashboard on a predefined time interval.

[2] ***Condition***: Any arithmetic expression.
Event/Signal: Comparison of device attribute to a discrete value or even to another device attribute value.

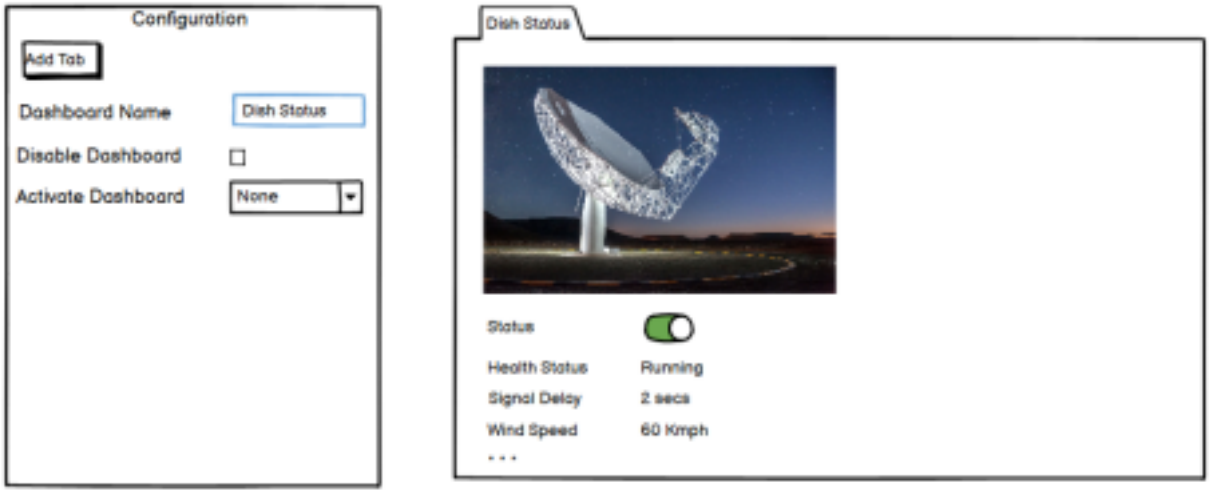
Active dashboards may be useful to signal changes in the state of some devices present in one of the tabs (for example a receiver that goes from Ok to degraded) or if an alarm is raised by the system. In this case, the dashboard would become active in the sense that it will show a notification to the user. The notification mechanisms can be of various types: visual blinking or sound effects are examples. An icon on the tab of the dashboard that contains the devices is another possible solution (see mockups in the figures below).

Operators can configure activation conditions based events (see Footnote 2) with some sound effects to immediately grab user's attention when wind speed exceeds a certain limit or when a device becomes overheated etc. The active dashboard feature will help in identifying where to look at in order to find additional information on the event.

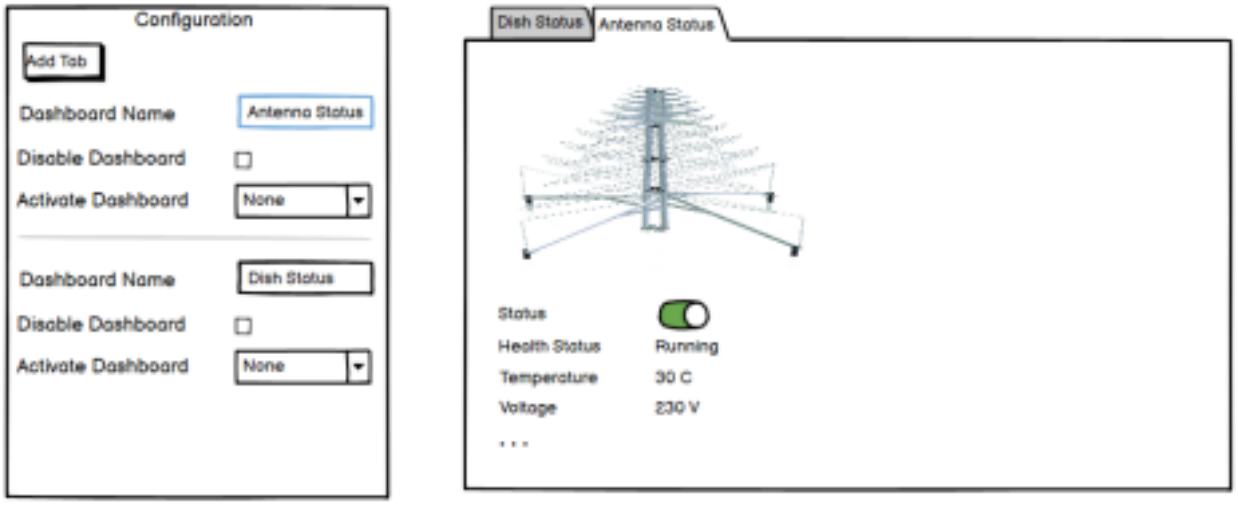
The user should also be allowed to disable some tabs when they are not needed or if they are temporarily archived. Disabled tabs are not visible in run mode till they are reactivated again in edit mode.

Widget mockup:

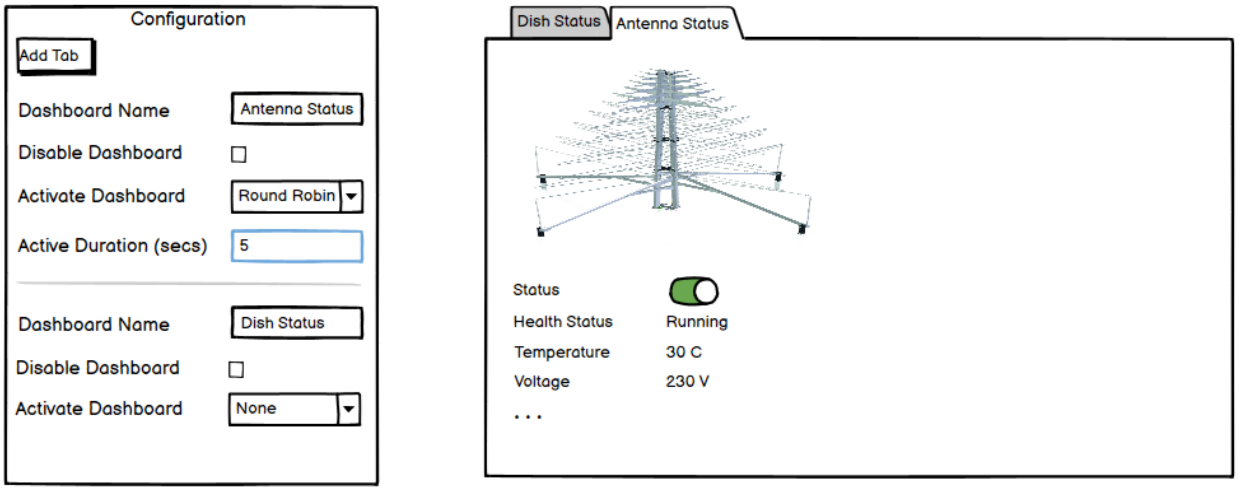
Figure 26: set of mockups describing different aspects of active dashboards



a. Adding new tab

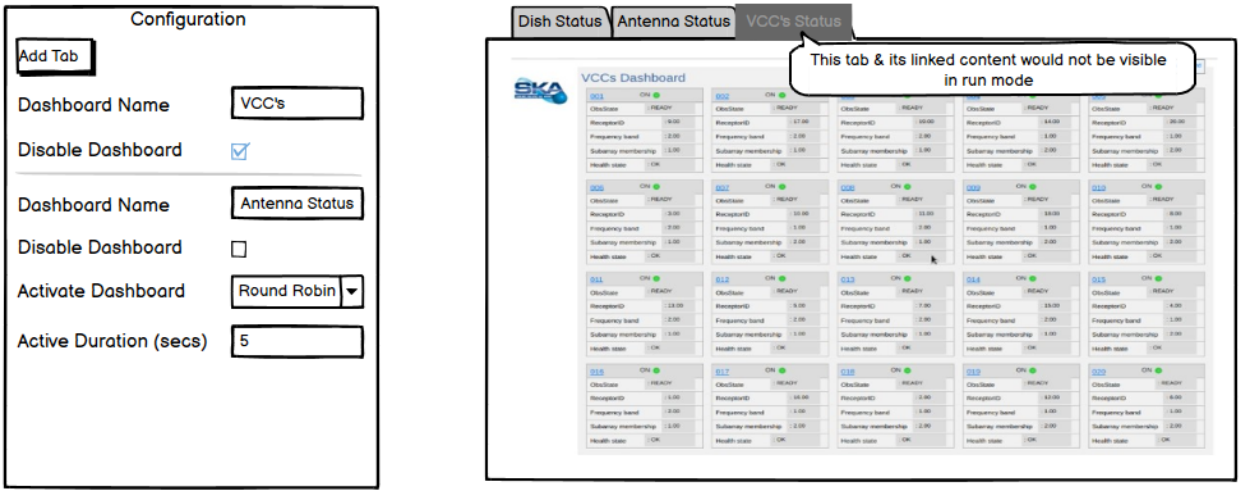


b. Configure tab

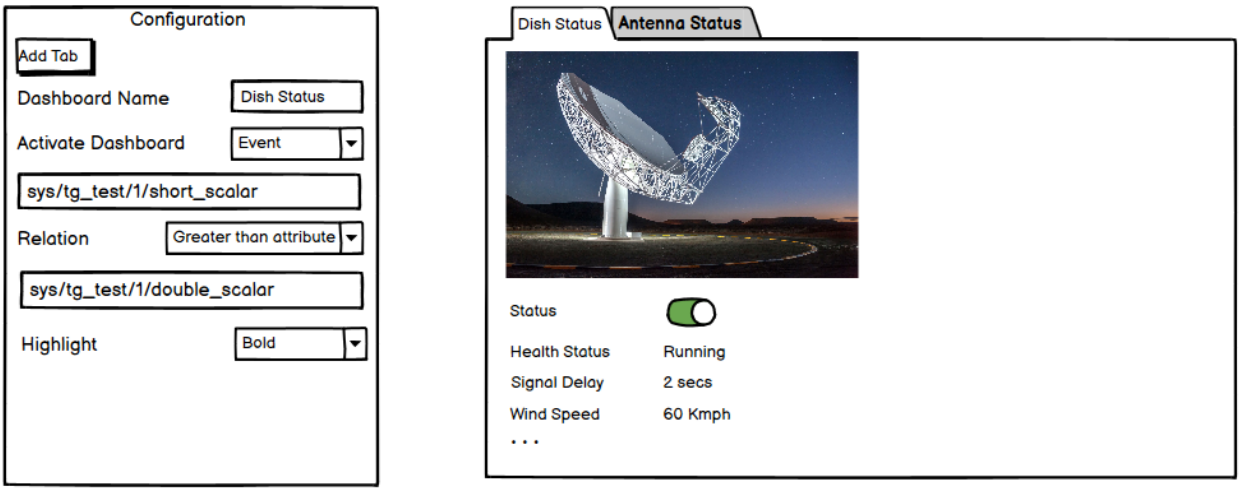


c. [Cyclic dashboard:]

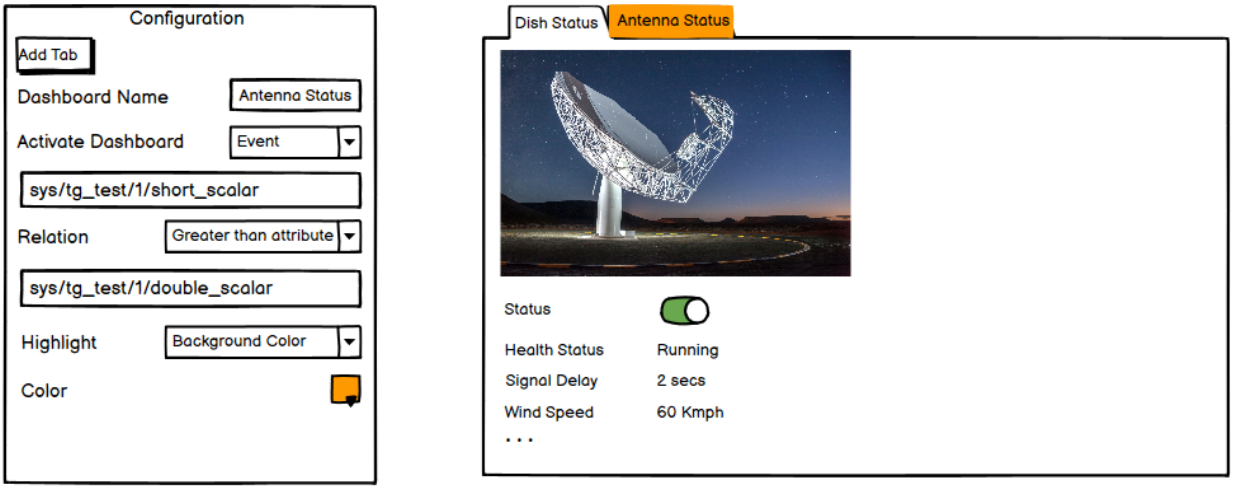
By setting “Activate dashboard” to “Round Robin” with some duration



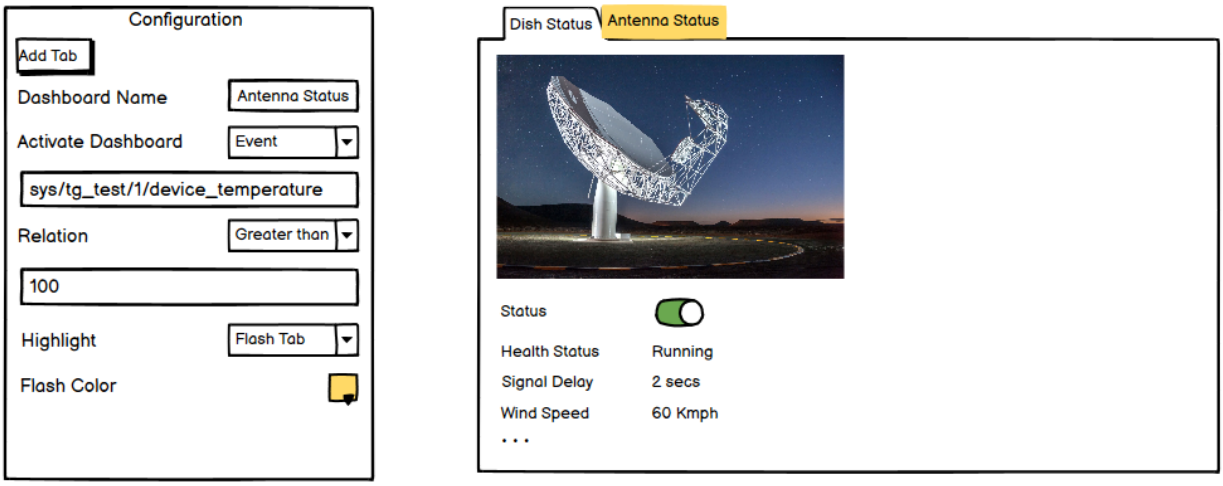
d. Disabled tab



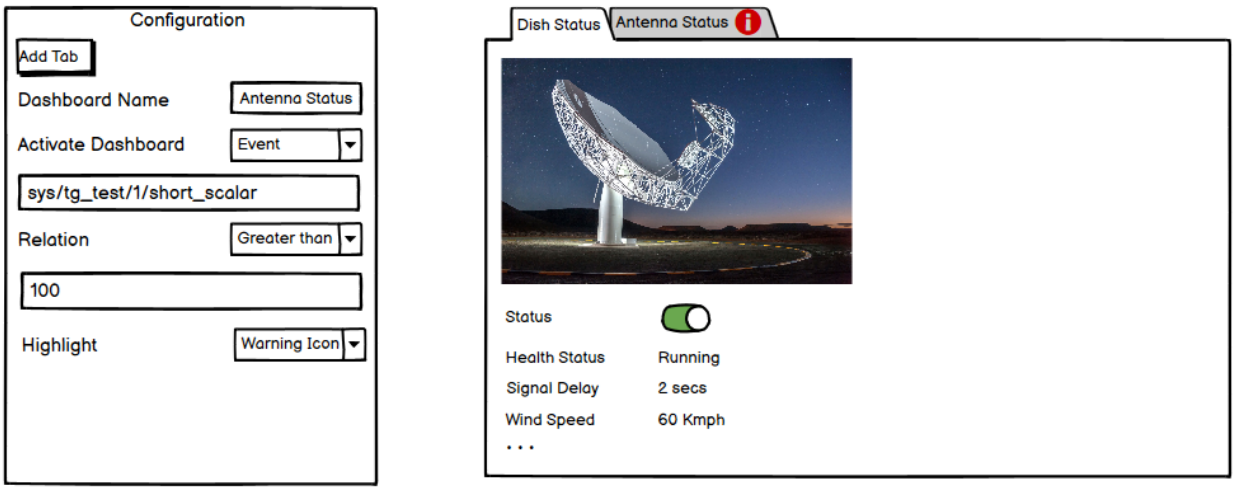
e. Setting up event on tab



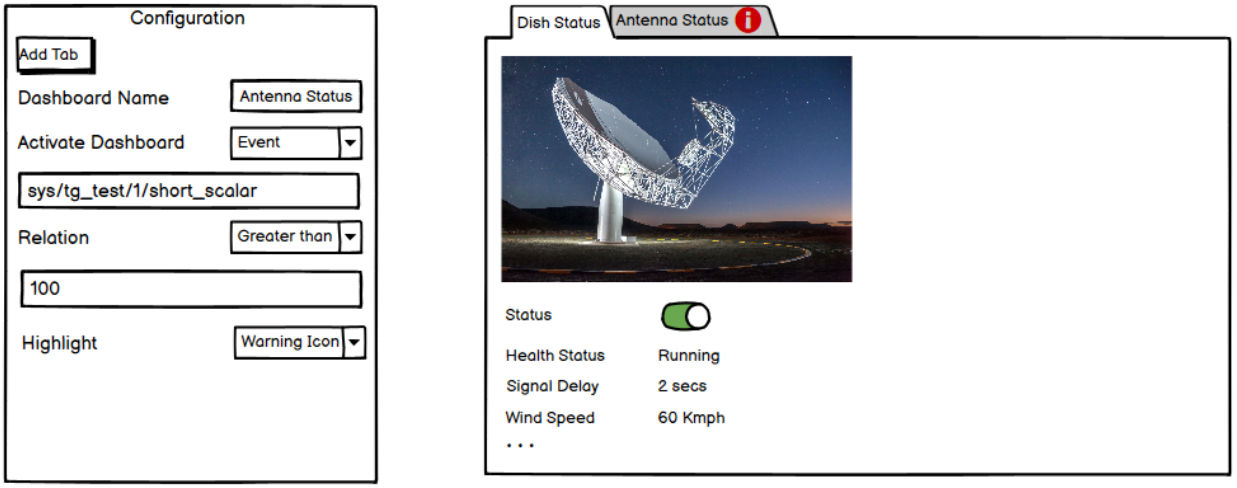
f. Highlight tab with background color



g. Highlight tab with flash background color



h. Highlight tab with warning icon



i. Highlight tab with sound effect

Time estimation: This widget is closely related to the tabbed dashboard widget. The configuration of the active dashboard is done when creating tabs of tabbed dashboard. Hence the mockup also shows the setup of the tabbed dashboard. It should be feasible to implement this feature in Taranta with **3 FP** excluding the effort of the tabbed dashboard.

Priority estimation

We believe the following list might represent a possible prioritisation of the widgets` implementation.

- **Tabular widget**: Users are already creating tables without having a proper widget to represent them. The initial implementation may be specialised to display device state;
- **Bar charts:** as soon as system performance and availability data will be gathered this widget will become crucial to get an overview of the system ;

- **Active widget:** initially it will collect information only from single devices. Feature development may take into account a wider system alarm system;
- **Tabbed dashboards and active dashboard:** browser tabs can be used to access multiple dashboards in the meanwhile. Tabbed dashboards will have to be in place in order to be able to implement active dashboards.
- **More general tabular widget:** table that can display different type of data**;**
- **Starburst**: will be very useful when the system will be operational, we are unsure about the short term usage.