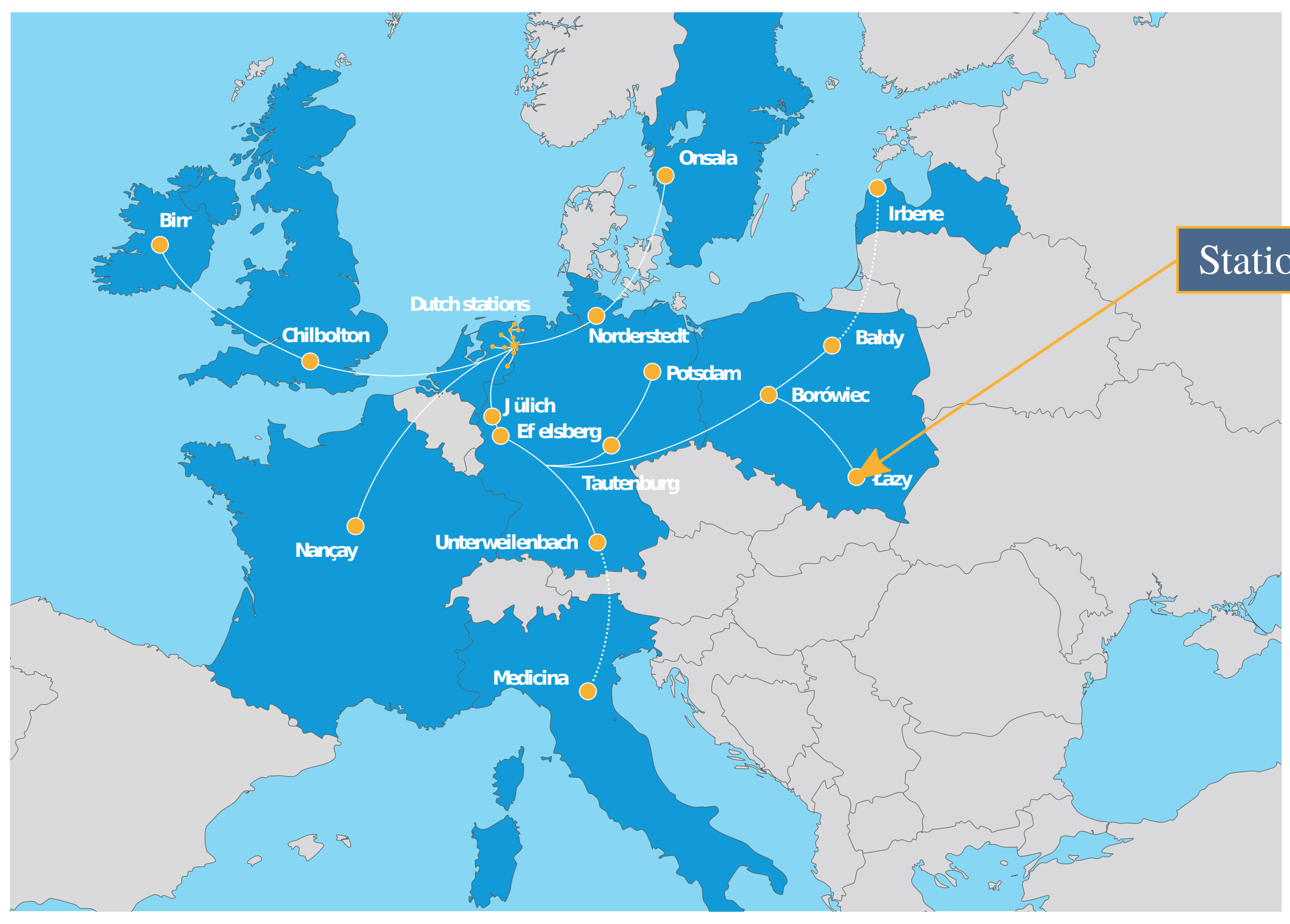
MMIS: StationMonitor

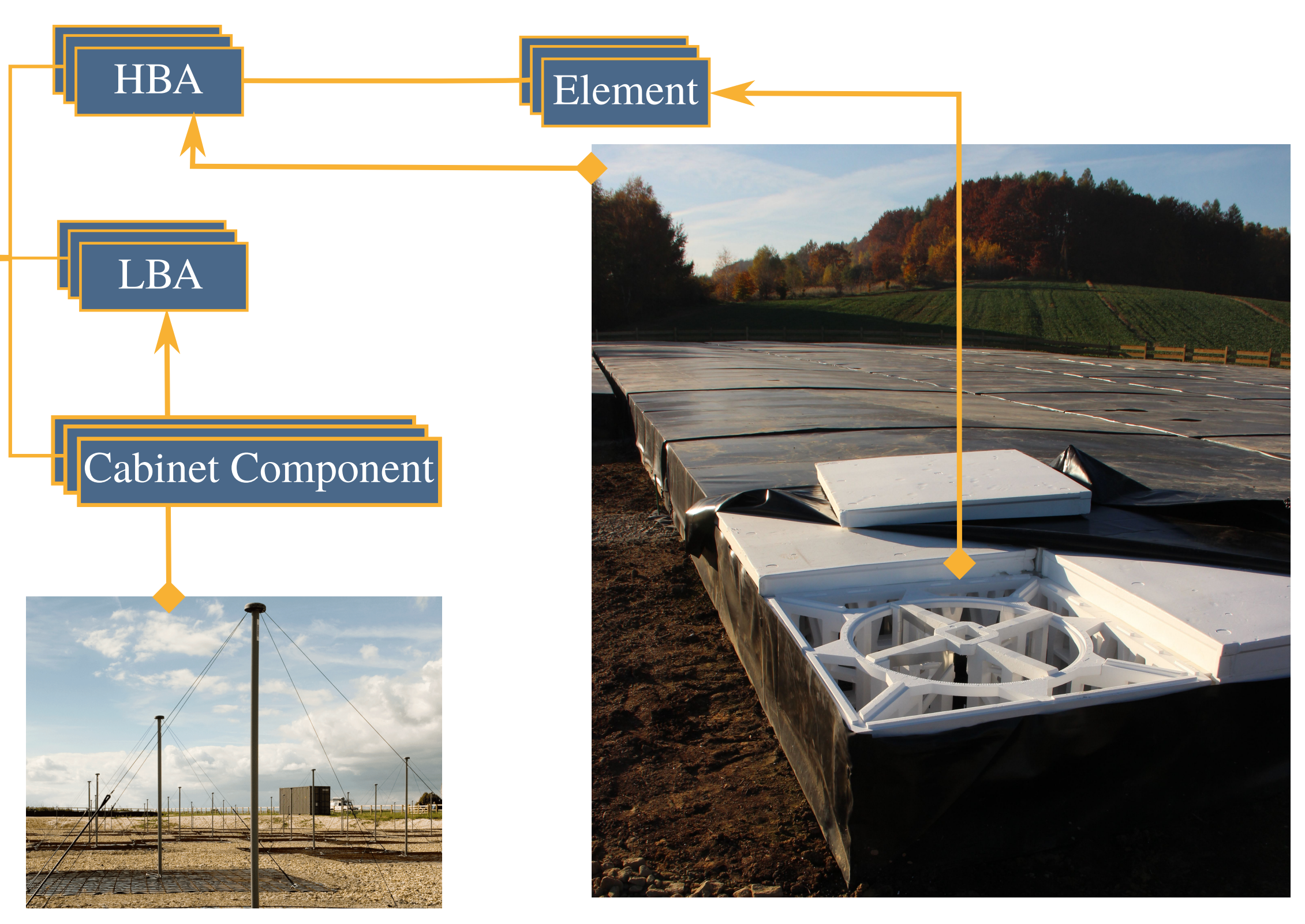
ASTRON

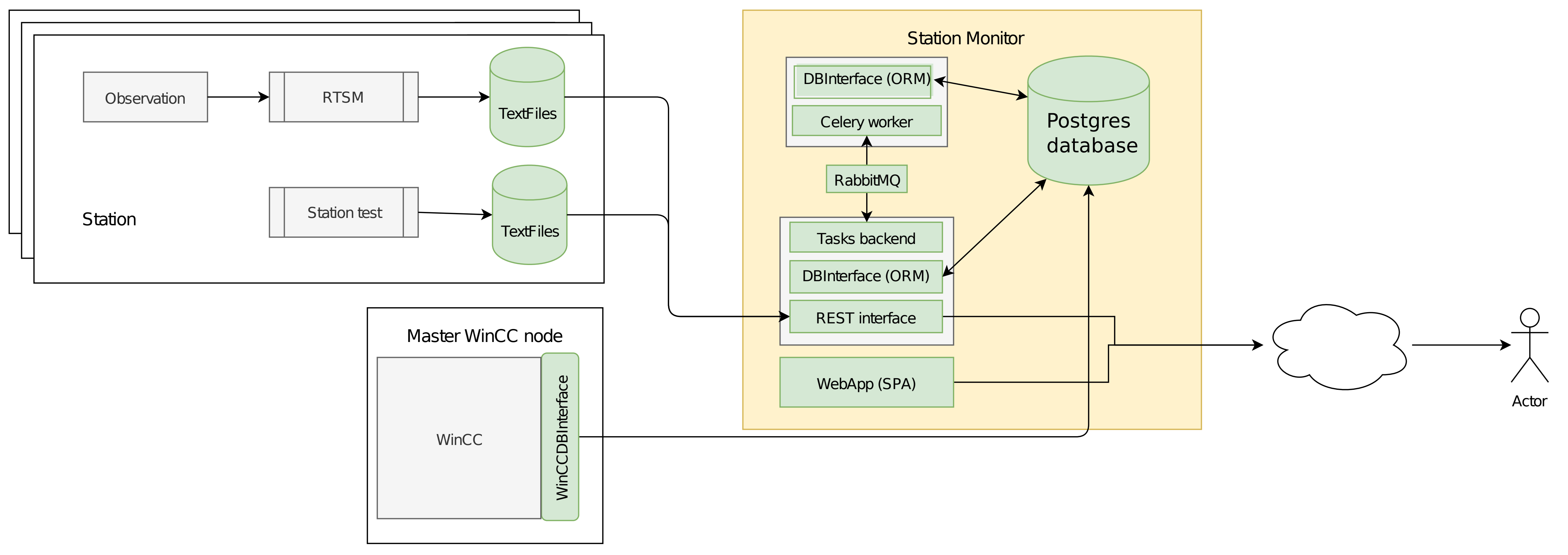
combining different tests data to provide an overview of the LOFAR array status

The LOFAR telescope

The LOFAR telescope is composed of several stations spread across Europe. A station consists of many components. However, only a subset of such components is tested by the station tests. For sake of simplicity, we will focus on the antennas only. There are two kind of antennas in a LOFAR station: the Low Band Antenna (LBA) and the High Band Antenna (HBA). LBA antennas are sensitive for frequency range ~10-90 MHz while HBA's are sensitive for ~100-200 MHz. An HBA antenna is composed of 16 elements. Each element hosts a couple of dipoles, one per polarization. The LBA antenna is considered a single component with a dipole per polarization.







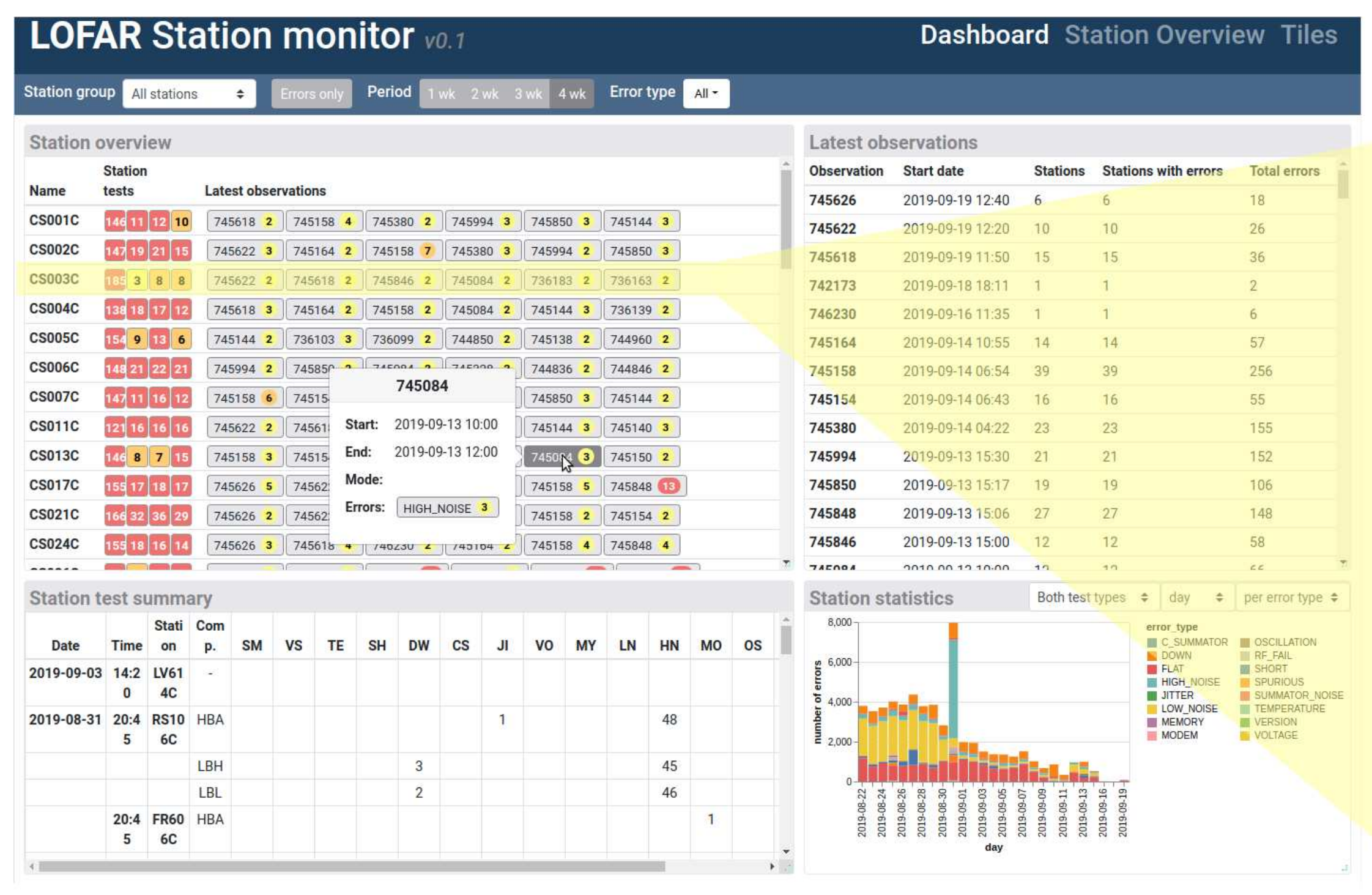
Implementation

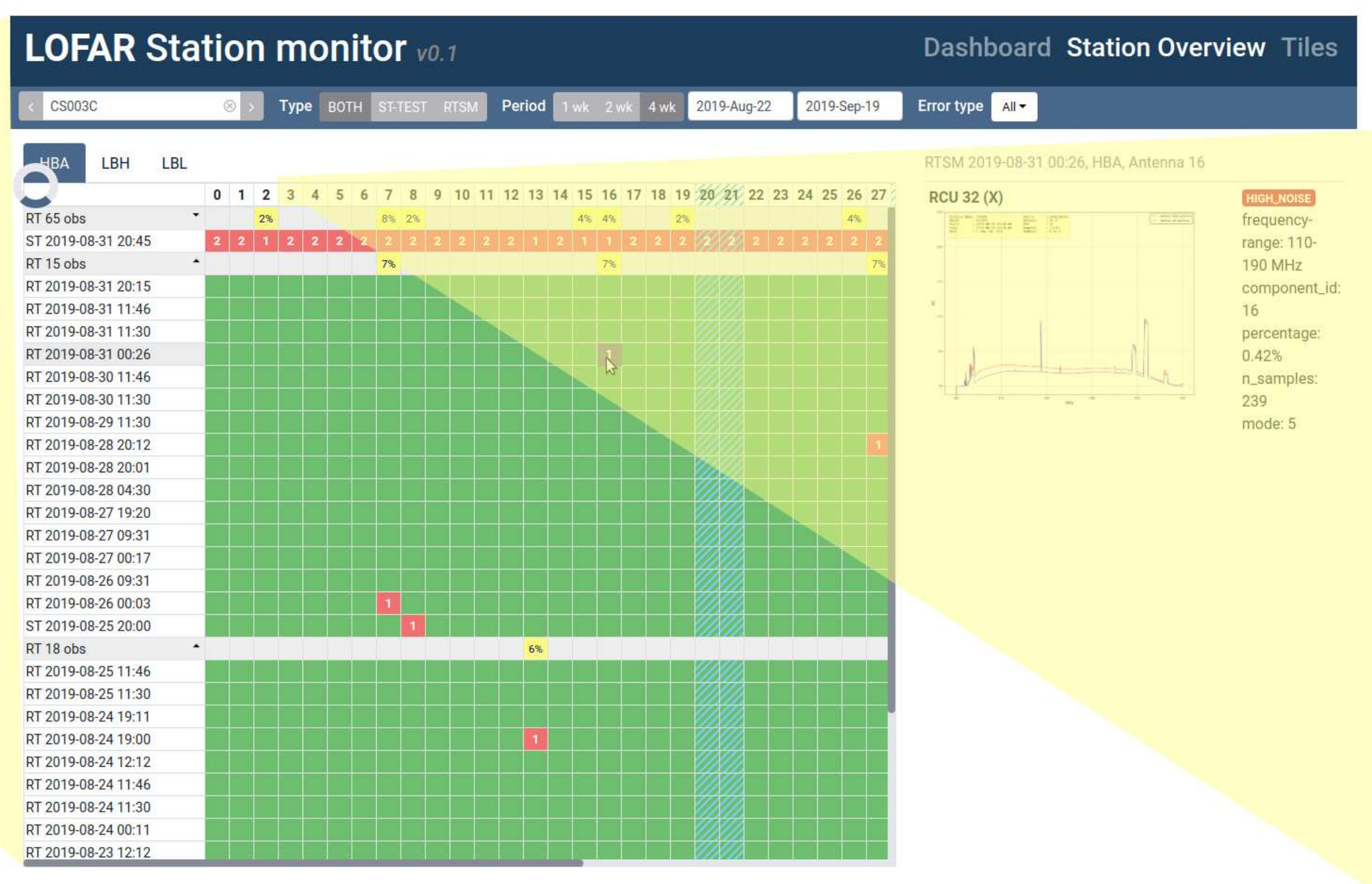
To expose the WinCC data to the station monitor we developed a c++ application called WinCCDBInterface using the WinCC C++ API and the Qt5 to react on every datapoint change listed in a configuration file and store the datapoint value directly in a table on the Postgres database. The WinCCDBInterface can also be configured to do a checkpoint at a given time where all the value of the configured datapoints are stored regardless if their value changes.

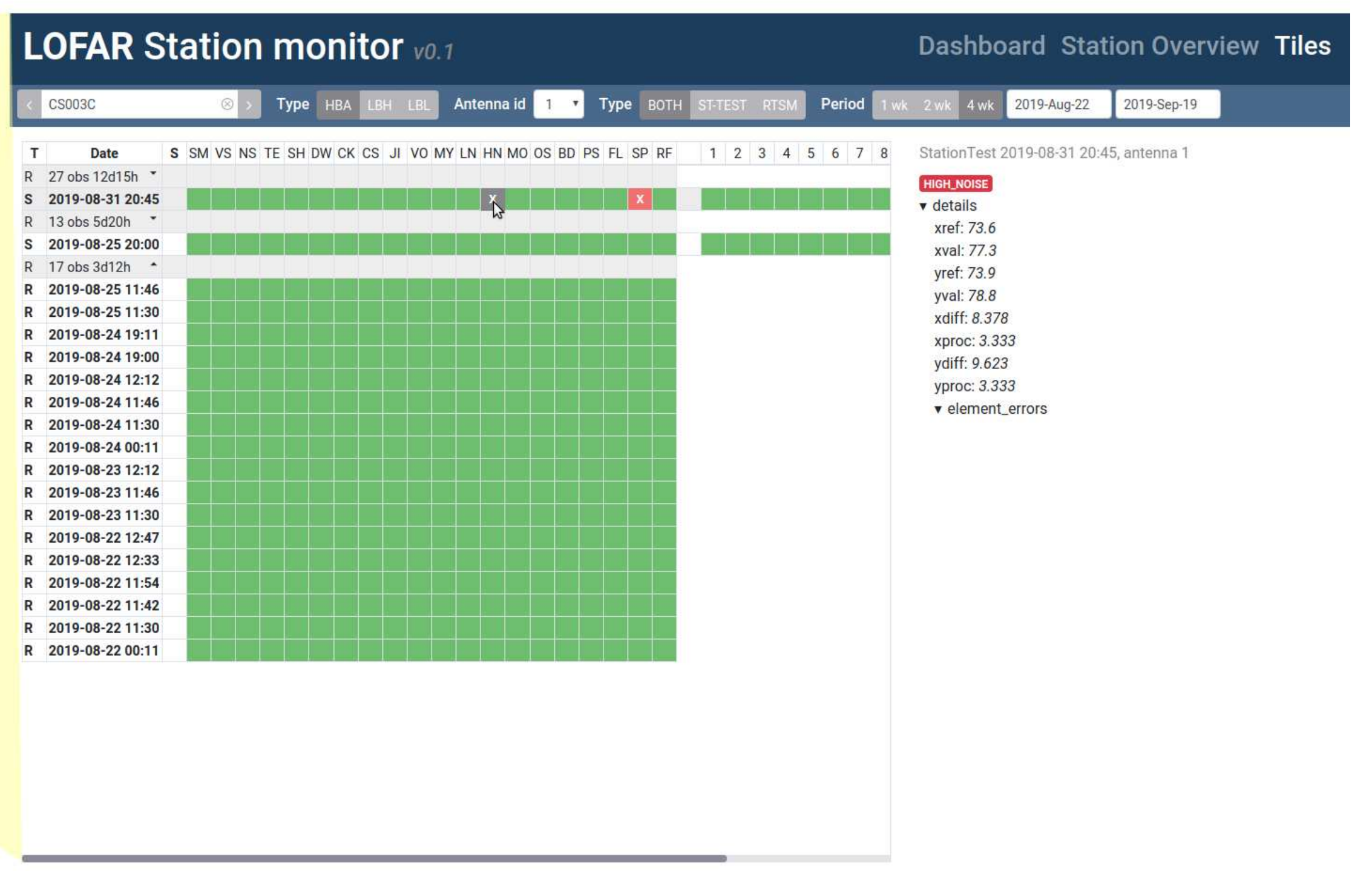
The database interface, tasks back-end and REST interface are implemented using the Django python framework. We used the Django Object relational mapping (ORM) to manage the database interaction and the tables and to expose a REST API through the django rest framework plugin. The text files from the RTSM and Station Test are automatically transferred to the Station Monitor server after the test is done. By using file system events, a REST call is performed with the test file and the corresponding task is fired. The tasks are carried out in the task backend developed with the help of the Celery framework. We choose the RabbitMQ message broker as result backend Celery. Using a asynchronous task queue prevents exhausting server resources and long blocking web server requests. A disadvantage is that in some cases a polling mechanism is necessary to check if a task has completed.

The user interface

from the station to the single component



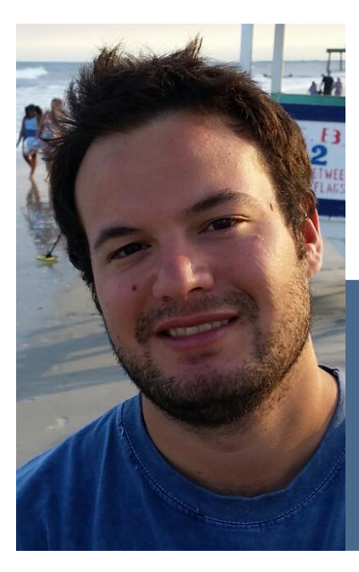




Future development

The first version is now operational. In the near future the Station Monitor will be extended with additional data types and functionalities. We will add the maintenance log containing repair actions executed by a field engineer and shown in the timeline of components and elements. This helps when interpreting the history of errors and to judge if a repair was successful. Moreover, we will add the possibility to create reports containing various statistics of the data containing in the station monitor. As this needs to be flexible we are currently experimenting with exporting the data to Elasticsearch and use Kibana as interface to create various reports and graphs. Finally, we would like to trigger on certain results of the Station Test or the RTSM so that problematic antennas can be automatically disabled or re-enabled once tested.

Reinoud Bokhorst



Mattia Mancini

ADASS XXIX