# Dynamic Control Room Interfaces for Complex Particle Accelerator Systems

B. E. Bolling, G. Fedel, M. Munoz, D. Nordt

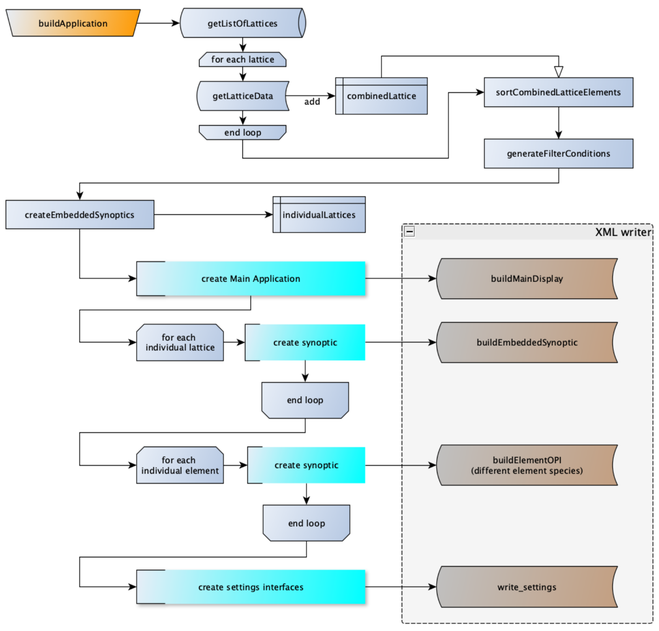


Fig. 1. A high-level flowchart depicting the script constructing the Accelerator Synoptics User Interface.

## OPIs Navigator OPI (prototype)

The openness of Phoebus introduces the issue that having a large number of user interfaces scattered across a large amount of directories, with many cases having support files that cannot be opened as standalone applications, the user interfaces needed become difficult/time-consuming to be found. The Operator Interfaces (OPIs) Navigator OPI is a prototype interface aimed at solving the issue by dynamically constructing an OPI to enable users to navigate amongst relevant user interfaces to save time and effort.

### Methodology

A Python script renders the Navigator OPI which loops through a given set of directories using a recursive strategy for any subdirectory and with a set of filters applied, such as methods for identifying support files such that they can be omitted. The script then launches a PyQt5-based user interface with a table of OPIs identified such that the developer may select which OPIs are to be included in the Navigator OPI, which may be referred to as accepted OPIs. A flowchart of the construction method can be seen in Fig. 2

### Results

The end result is a Python script dynamically constructing the OPIs Navigator OPI which enables fast and efficient navigation between accepted OPIs, with a screenshot attached in Fig. 3 in which the Navigator OPI shows the RF Overview OPI. There are further discussions ongoing on how to implement the Navigator OPI as a Phoebus element directly - hence it being referred to as a prototype - since it is now clearer what is needed from an Operations perspective with regards to an OPI Navigator.

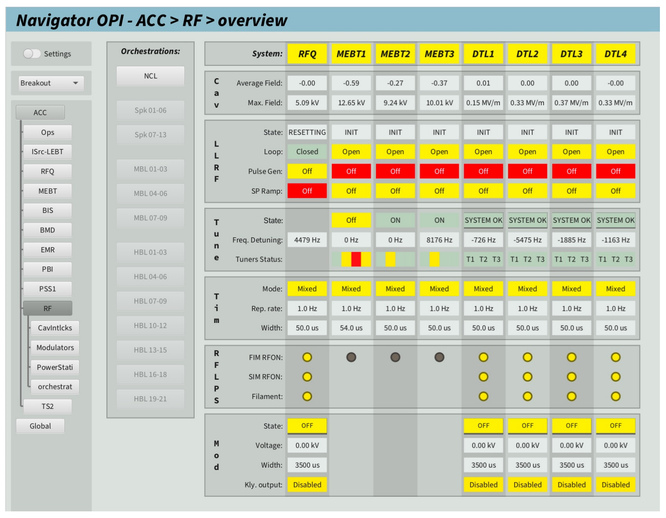


Fig. 3. The OPIs Navigator OPI currently showing the Overview Interface for the Radiofrequency components of the NCL.


## References

R. Garoby et al., "The European Spallation Source Design", Physica Scripta, vol. 93 (12), 2018, IOP Publishing, doi:10.1088/1402-4896/aaecea.

K. Shroff et al., "New JAVA Frameworks for Builing Next Generation EPICS Applications", in Proc. 17th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS'19), New York, NY, USA, 2019, paper WESH1002, pp. 1497-1500, doi:10.18429/JACoW-ICALEPCS2019-WESH1002.

D. Nordt, "ESS Rules for the Visual Design of EPICS Operator Interfaces", European Spallation Source, Lund, Sweden, 2023, Internal European Spallation Source report, ESS-4752055, unpublished.

G. Van Rossum, Fred L. Drake, "Python 3 Reference Manual", Scotts Valley, CA, USA, 2009, CreateSpace, ISBN:978-1-4414-1269-0.


@ESSneutron

The European Spallation Source (ESS) is a research facility under construction aiming to be the world's most powerful pulsed neutron source. It is powered by a complex particle accelerator designed to provide a 2.86 ms long proton pulse at 2 GeV with a repetition rate of 14 Hz. Commissioning of the first part of the accelerator has begun and the requirements on the control system interfaces varies greatly as progress is made and new systems are added. In this paper, three such applications are discussed in separate sections.

A Navigator Operator Interface (OPI) was developed for the control room interfaces aimed towards giving operators and users a clear and structured way towards quickly finding the needed interface(s) they need. The construction of this interface is made automatically via a Python-based application and is built on applications in any directory structure both with and without developer interference (fully and semi-automatic methods).

The second interface is the Operations Accelerator Synoptic OPI, which uses a set of input lattices and system interface templates to construct configurable synoptic view of the systems in various sections and a controller panel for any selected system.

Lastly, there is a configurable Radiofrequency (RF) Cavities Orchestration OPI for Operations, allowing in-situ modification of the interface depending on which systems and components are selected.

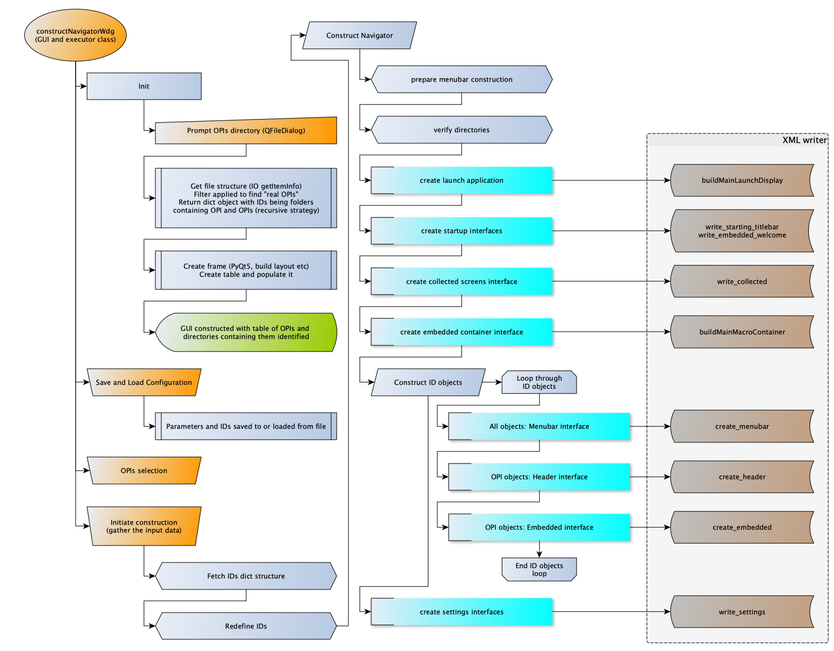


Fig. 2. A high-level flowchart depicting the script constructing the Operator Interface Navigator User Interface.


Fig. 4. The Accelerator Synoptics Interface showing the high-level overview of the Linac MEBT section.

## Accelerator Synoptics OPI

The Accelerator Synoptic OPI is designed to provide a general overview of the state of a linear accelerator as a whole combined with ways to quickly interact with each accelerator component. As with the Navigator OPI, it is constructed by activating a Python script with a set of embedded display templates, blockicons (pushbuttons combined with visual elements), filter setups, and one or more accelerator lattice files. The script sorts all components in terms of position along the accelerator, and builds the synoptic application with interactive elements and their associated embedded displays.

### Methodology

A Python script creates one embedded display per accepted element and filter combination, with an accepted element having a valid blockicon (a symbol combined with a push-button) associated with it as well as a name of the physical equipment in the input lattice file. For each synoptic combination, separate files are constructed and displayed as embedded screens per section (or combined section).

By using templates and different identifiers, the developer can easily modify and add or remove different components to information displays and the embedded displays (including controllers, readbacks, history plots, etc.). A simplified flowchart depicting the construction process is shown in Fig. 1.

### Results

The end result is an Accelerator Synoptic OPI which is easy to use and enables users to quickly navigate through the different components as well as have an understanding of the overall state of the machine. Sectional overviews further adds to the completeness of the application, as can be seen in Fig. 4.

Furthermore, a filter function can be set up to show or hide different accelerator component species, interacting with/showing live data from any component (presuming that it had a template file enabling this) as well as information regarding it (with e.g. lattice data and a picture of it).

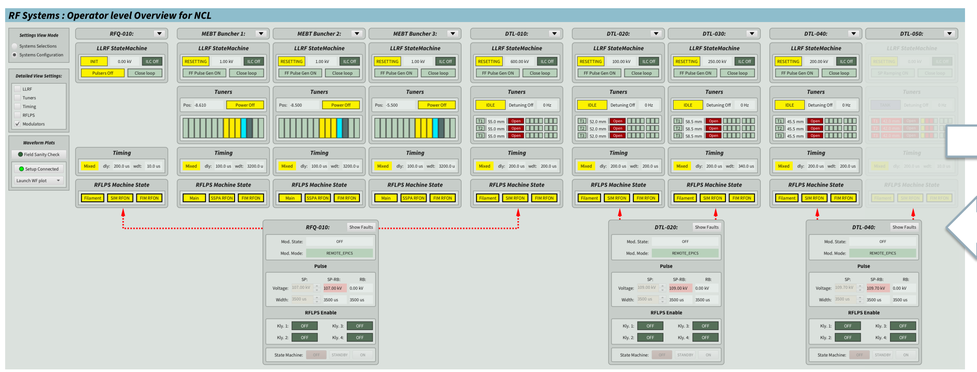


Fig. 5A. The RF Orchestration Interface with all (except modulator) subsystems collapsed.

## Conclusion

A set of OPIs were created to give Operators a quick overview and simpler means to control a very complex machine that currently is in installation phase, meaning that the OPIs have to be quickly adaptable and dynamic to facilitate operational flexibility. For this purpose, the complex OPIs used for Navigating between other OPIs (Navigator OPI) and between different accelerator components (Accelerator Synoptic OPI) are successfully generated using Python scripts when a change is made to the accelerator lattice, a request to include more system species is made, or newly developed and deployed OPIs that Operators added are to be included. Therefore, the script-generated OPIs are fully scalable to the full accelerator whilst the RF Orchestration OPI needs slight changes to some embedded system components in order to support e.g. 4-klystron-per-modulator and Spokes RF systems.

## RF Orchestration OPI

The configurable RF Orchestration interface for Operations allows in-situ modification of the setup depending on which systems and components are to be controlled for different purposes. Using local signals, users can select which accelerator components (such as Medium Energy Beam Transport (MEBT) Buncher 1) to show and which subsystems (Modulator, Low-Level RF, etc.) to show detailed information about and controllers and for which to show only the essential information.

### Methodology

The OPI consists of submodules with different inputs for their naming and can hence be reused for other similar systems. When combined, they span e.g. the RFQ and the DTL RF systems (2-klystrons-per-modulator system), meaning that a relatively small amount of work is needed to create the OPIs for different system species - such as solid state amplifier bunchers having no modulators or 4-klystrons-per-modulator systems.

### Results

A fully expanded and an almost fully collapsed view (in which only the Modulator module is expanded) is shown in Fig. 5 (parts *B* and *A*, respectively. As can be seen, the collapsed view offers the user a quick overview of the system states via colour codes based on the state relative to its expected state during normal operation, enabling the user to quickly gain an understanding of the state of the systems as well as means to quickly interact with systems or launch OPIs with more advanced (less interacted with) settings:

- Red: System state is not OK (e.g. off or in fault).
- Yellow: System state might need attention, as it is not in the state it should be for normal operation.
- Green: System state is OK for normal operation.

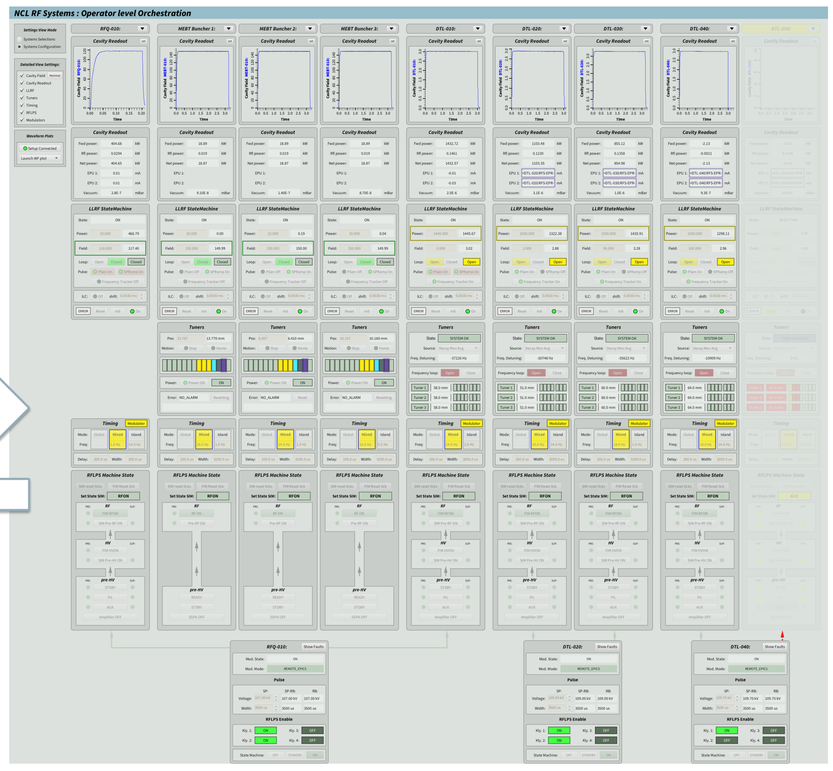


Fig. 5B. The RF Orchestration Interface with all subsystems expanded.


## Acknowledgements

The authors acknowledge the great support and collaboration across multiple divisions at ESS, including the Operations Division and the Integrated Control System Division, and to all colleagues reviewing and providing feedback both on the OPIs and on this paper itself.