

MINT, AN ITER TOOL FOR INTERACTIVE VISUALIZATION OF DATA

L. Abadie, G. Carannante, I. Nunes, J. Panchumarti, S. D. Pinches, S. Simrock, M. Tsalas
ITER Organization, Saint-Paul Lez Durance, France
D. Makowski, P. Mazur, P. Perek, Lodz University of Technology Department of
Microelectronics and Computer Science Wolczanska, Lodz, Poland
A. Neto, Fusion For Energy, Barcelona, Spain
S. S. Kalsi, Tata Consultancy Services, Pune, India

Abstract

ITER will produce large volumes of data that will need to be visualized and analyzed. This paper describes the development of a graphical data visualization and exploration tool, MINT (Make Informative and Nice Trends), for plant engineers, operators and physicists. It describes the early development phase from requirements capture to first release covering the mistakes, lessons learnt and future steps. The requirements were collected by interviewing the various stakeholders. The initial neglect of the architecture and user-friendliness turned out to be key points when developing such a tool for a project with a long lifetime like ITER. A modular architecture and clear definition of generic interfaces (abstraction layer) is crucial for such a long lifetime project and provides a robust basis for future adaptations to new plotting, processing and GUI libraries. The MINT application is based on an independent plotting library, which acts as a wrapper to the choice of underlying graphical libraries. This allows scientists and engineers to develop their own specific tools, which are immune to changes of the underlying graphical library. Data selection and retrieval have also been developed as a separate module with a well-defined data object interface to allow easy integration of additional data sources. The processing layer is also a separate module, which supports algebraic and user-defined functions. The development is based on Python [1] and uses Qt5 [2] as the visual backend. A first release of the 1-D trend tool (MINT) has already started and will be used for the ECH (Electron Cyclotron Heating) system commissioning. Other visualization tools will be developed in the future that build upon the same underlying modules.

INTRODUCTION

ITER is already producing data which need to be plotted and analyzed quickly. This paper describes the requirements and challenges, the development phases and various lessons learnt.

REQUIREMENTS

The top level requirement for data visualization is to be able to plot data for a time range or a pulse identifier.

Stakeholders

To define the detailed requirements, the first thing was to identify the main stakeholders. We identified three basic categories of users:

- Plant engineers whose main objective is to make system investigations; some of them will also perform research activities.
- The science team whose main objective is to analyze pulses and carry out research.
- The operation team whose primary focus will be the analysis of pulses.

Types of Data

The next step was to list the different data types of interest. After interviewing the stakeholders, the following list was constructed:

- Time and profile traces;
- Spectra;
- Fluxes (see Figure 1);
- Images and videos (see Figure 2)

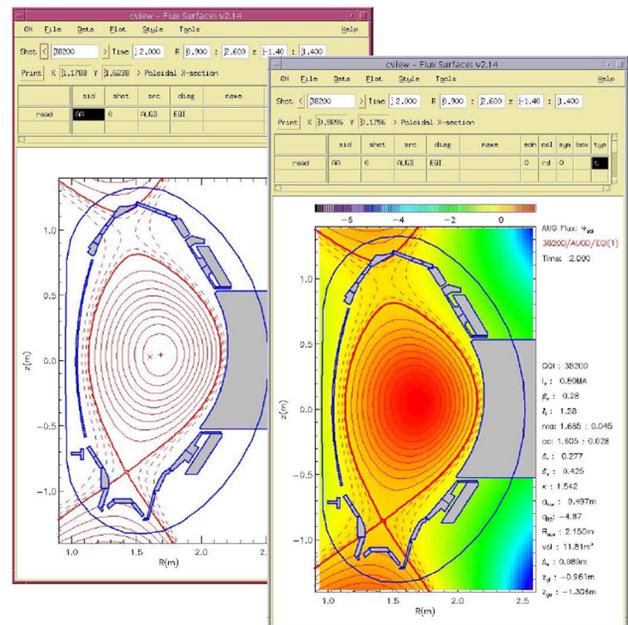


Figure 1: Example of magnetic flux surfaces as represented in CVIEW used at ASDEX (courtesy from G.Conway).

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

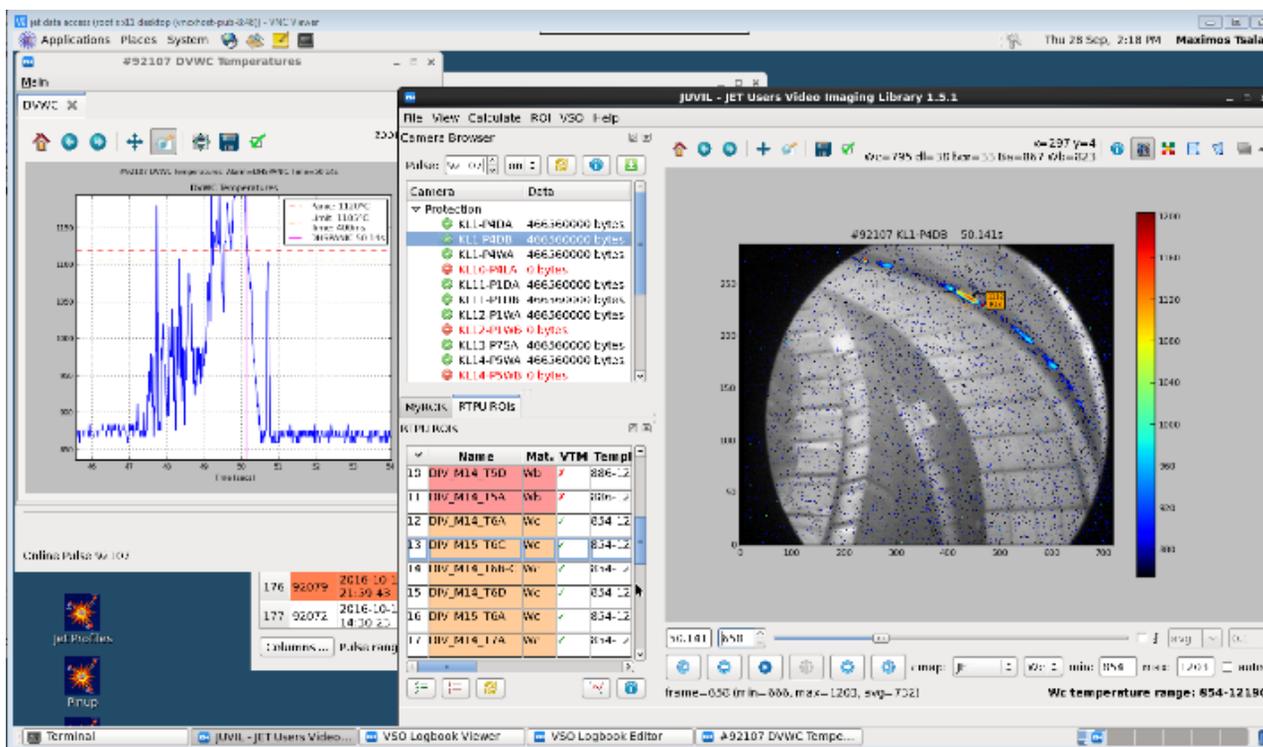


Figure 2: Example of image analysis (JUVAL used in JET).

Collecting Requirements

In order to get more practical users' feedback, a prototype was developed based on existing similar tools. It enabled collecting more precise user requirements. By making various demonstrations to the stakeholders a set of requirements were collected, including:

- Be able to overlay data from multiple pulses in one plot
- Be able to plot quickly 100 signals in multiple plots
- Be able to update the colors, to mark the points, to control how two points are connected
- Zoom/pan interactively: ITER has very fast sampled signals (kB-GB/sec). Loading all the points in memory on the client side would be very resource intensive. So there is a need to be able to dynamically retrieve data during zooming/panning operations with acceptable performance
- Be able to apply basic processing on one or more signals

DEVELOPMENT – PHASE I

Agile Development

As stated in the previous section, an existing product based on C++/Qt5 was initially extended. An agile development approach was opted to add new features. Bugzilla was used to track all new features, SVN to store the code, and Jenkins as the continuous integration system. Resources (CPU and memory) and performance optimization were the main objectives besides feature implementation. Thanks to ECH and ICH team, we could simulate the acquisition of fast signals from 2MHz to 125MHz. The demonstration of zooming interactively a signal sampled at 125MHz was successful: the zooming was done until one was able to measure the distance between 2 points (8 ns). Figure 3 shows the layout of our prototype.



Figure 3: Look and feel of our first prototype based on Thermavip (tool used at CEA).

Training Stakeholders

After several months of development, a training was given to the stakeholders through dedicated separate sessions. They were provided the tool and requested to plot data for a particular time range and for a particular pulse identifier.

The results were not so good. Many had difficulties to execute basic actions. None of them could plot data without our help.

Scientists also challenged the tool by wanting to add new data sources for data retrieval, something that was not easy to do as per the implementation of the existing tool. As a consequence, the chosen approach was reflected upon in detail and the way forward discussed.

Lessons Learnt

During this initial development phase, it was realized that three key aspects were omitted:

- 1) Usability of the tool: non-developers shall be able to use the tool with very limited knowledge. When adding new features, the GUI became too complex and the main functionalities which needed to be kept simple, were not clearly identified.
- 2) Architecture: As a consequence of adopting an agile development approach, not enough thought was placed on the architecture, something that became clear when users requested to integrate a new data sources.
- 3) One big tool versus many tools: the initial concept was to have one big tool which could handle any data type and all stakeholders' views. Indeed it seemed attractive to provide users with one tool and then depending on their needs they can plot fluxes or profiles or time traces. With hindsight, it turned

out that the GUI was too complex in terms of usability and maintainability. A decision was thus taken to develop tools with a limited but well defined scope to ensure user-friendliness and improved maintainability.

With all this in mind, a new development strategy was implemented.

DEVELOPMENT – PHASE II

Review of Existing Tokamak Tools

It was decided to invest more time analyzing existing tools currently in use on other tokamaks and stellarators.

It was found that the visualization tools on most devices shared similarities:

- Multiple tools are often developed to cover different aspects of data visualization: one tool for time traces and limited support for images, surfaces; one tool for fluxes and another tool for image analysis
- The User Interface for the time traces is based on a table and not a tree: it allows to be agnostic to the data structure used for the experimental signals and also eases the configuration of the plots and their signals
- Plotting data versus a pulse or a time range, zooming/panning, displaying crosshair were very straight forward
- Importance of good default settings

Architecture

As a result of the above review, a modular approach was subsequently adopted and it was furthermore decided to develop a plotting library so that scientists could reuse this layer to develop their own expert applications. Python and

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2022). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

PySide [3] where selected as the basis for the development, Python being a commonly used language for scientific visualization.

The current architecture is based on 4 blocks:

- **Logging** : module which allows logging to file/console
- **DataAccess**: module which allows retrieving data from various data sources. To add a new data source, it is sufficient to add a new Python class and implement method to retrieve data. The data object used in this class is generic.
- **Processing** module: this module is responsible for processing the data as required after it has been fetched. It is responsible for time-base alignment when an operation involves more than one signal. Note that the processing is reapplied in case of a zoom/pan operation.
- **Iplotlib**: module which is responsible to plot the data and react to events such as zooming/panning. To demonstrate that the interface is independent from the graphics backend (here Matplotlib [4]), a VTK [5] back-end has also been developed. This allows a user to choose to use iplotlib either with Matplotlib (default) or VTK.
- **M.I.N.T.** (Make Informative and Nice Trends) is a tool that makes use of all the above modules. It is the primary tool for displaying time traces and analysis.

Current Status

Many features have been developed, as can be seen in Figure 4, such as plotting data for a given time range, or for a pulse identifier.

Features
View one or several 1-D trends (X axis is the time)
Support for pulse-based and time range data access (all the plots show the same time range/pulse)
Support for time range or pulse specialization at row level (allow comparing two plots side by side)
Support for pulse overlay
Support for interactive zoom
Support for crosshair (over multiple plots)
Support for envelope display
Support for pan interactive
Support for preferences update (color of a signal, draw-style (interpolated, stair), display points with markers)
Add a plot title
Support for automatic X-axis scale (if I zoom or pan on one plot, all the plots will be rescaled to the same time window).
Import/Export list of variables
Save/load configuration a plot
Support for streaming
Support for full-mode
Support for basic processing of one signal
Support for Y and X-axis label edition
Generate a screenshot given a preference file in an headless environment (to allow report automation)

Figure 4: List of main features which have been actually implemented.

The tool layout with both back-ends is shown in Fig. 5 and Fig. 6.

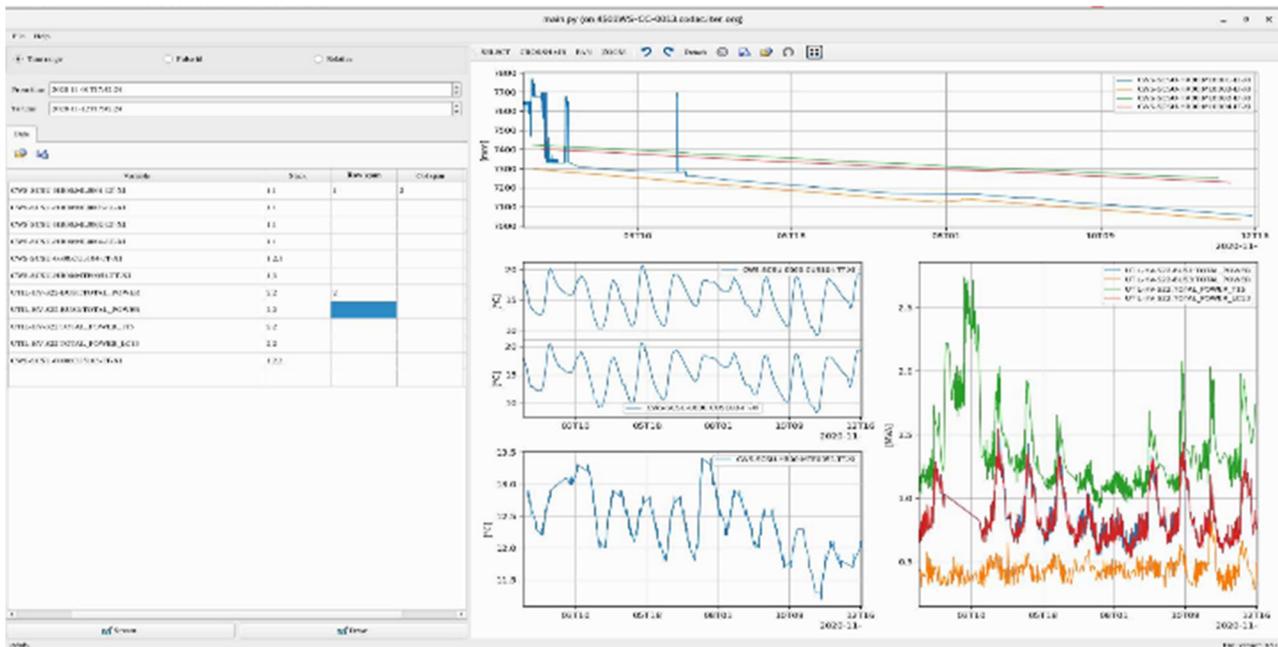


Figure 5: Layout of MINT (using Matplotlib back-end).

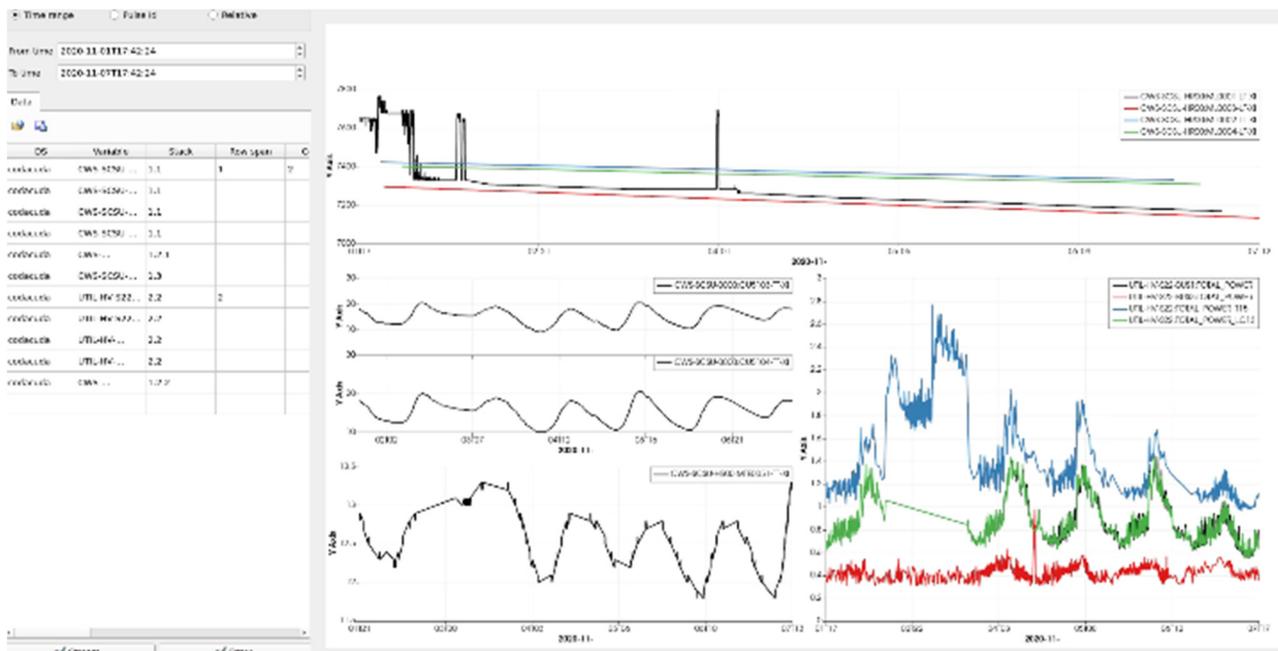


Figure 6: Layout of MINT (using VTK back-end).

Next Steps

There are still important features to be implemented including:

- Extending the data processing to support the manipulation of multiple signals and support for user-defined functions
- Extending the ipplotlib module to support new plot types such as surfaces, a 2-D slice of 3-D data chosen with a slider, histograms, etc.
- Implement a browsing plugin to search for variables and metadata to ease filling the table.

It is also important to note that the performance for this tool will be critical. If zooming is too slow, users will give up and seek alternatives. That's why one of the next big investments of effort will be to improve the data access layer.

CONCLUSION

Good progress has been made on the development of a tool for data visualization. A lot has been learnt from the initial development experience and together with a more rigorous capture of the requirement this has provided a robust basis for subsequent development. The first version of the tool is now being actively used by the plant engineers (Cooling Water).

ACKNOWLEDGEMENTS

We would like to thank all stakeholders Trevor Blackman, Laura De Frutos, Peter Des Vries, Fabienne Kazarian, Arthur Leveque, Victor Moncada, Marco Riva, Marc-Henri Sautenet, Mireille Schneider, Luca Zabeo and the management Mikyung Park, Anders Wallander, and Tim Luce for their support in this work.

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

REFERENCES

- [1] Python, <https://www.python.org/>
- [2] Qt5, <https://www.qt.io/home>
- [3] PySide, <https://pypi.org/project/PySide2/>
- [4] Matplotlib, <https://matplotlib.org/>
- [5] VTK, <https://vtk.org>