A generic HMI for physics experimentalists developed with R, rJava, htmlwidgets, plotly and shiny tools

L. Pantera, M. Savanier

A generic HMI for physics experimentalists developed with R, rJava, htmlwidgets, plotly and shiny tools

Laurent Pantera[1], Marion Savanier[2]
[1] *CEA, DEN, CAD/DER/SPESI/LP2E, Cadarache, F-13108 Saint-Paul-lez-Durance, France*
[2] *GE Healthcare, 283 Rue de la Minière, 78530 Buc, France*
emails: laurent.pantera@cea.fr, marion.savanier@ge.com

Context and objective

- Our Laboratory works in the Framework of the CIP (CABRI International Program) Project operated and managed by the French Nuclear Safety and Radioprotection Institute (IRSN). **CABRI** facility is a **nuclear pool-type reactor** designed to study the **behaviour of fuel rods at high burnup under Reactivity Initiated Accident conditions** that could occur in nuclear power plants (Pressurised Water Reactor).
- Experimentalists use R scripts for data-processing and to visualise their data, which is paramount in an exploratory context.
- As the number of developed scripts increases over time, our aim was to make them available through an **ergonomic Human-Machine Interface**. This way, experimentalists would be able to **add their own contributions without concerted efforts** in terms of HMI and OOP programming. This would also lessen the risk of making manual mistakes and significantly **enhance the speed with which analyses can be conducted**.

Software Framework

Java environment

Java: from the Swing and AWT framework to the JavaFX framework providing the Web Engine functionality
Code: using the Design Patterns' scalability (COMMAND, MVC, DAO) to easily fit future menus
Windows: created with JavaFX Scene Builder tools
IDE: eclipse (with ObjectAid for the UML diagrams visualisation)

R environment for Java connection

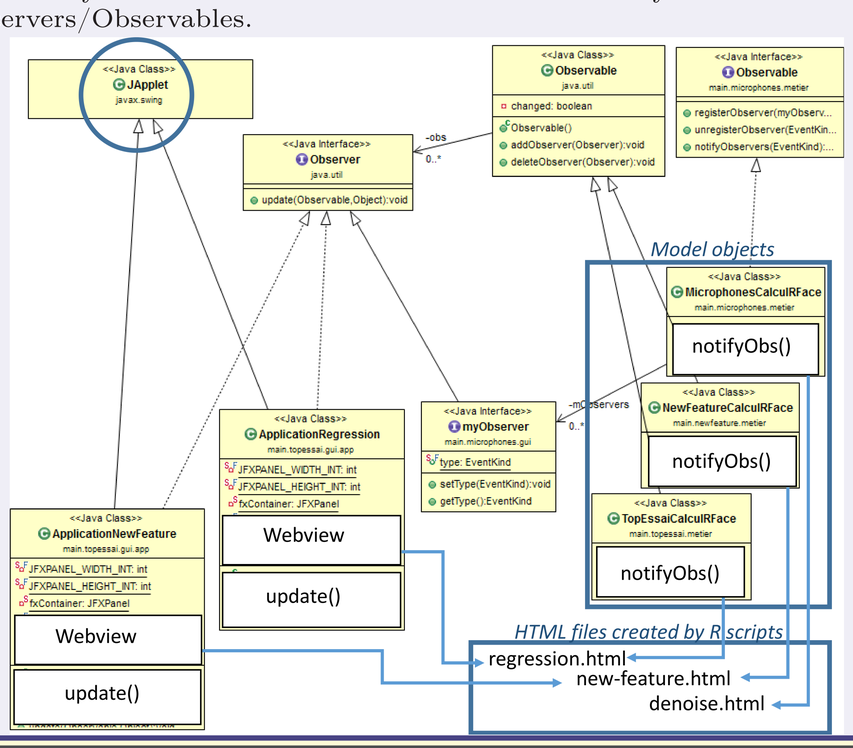
- **rJava** package [4], Java/R Interface **JRI** (org.rosuda.JRI) and **REngine** (org.rosuda.REngine)

An ergonomic HMI

1. A settings panel displays the variables and their values which can be edited manually.
2. An embedded R console allowing the user to perform additional analysis tasks
3. A zone of action buttons triggering the execution of the scripts
4. Tabbed WebView panels with the interactive graphics linked to the html files generated by the external R scripts

Methodology to add a new feature

Step 1: Write R scripts
Step 2: Fill in a parameter file that contains a list of initialisation parameters and their default values from which the panel of input/output parameters is generated
Step 3: JavaFX's visual layout tool is used to customize the GUI and integrate as many WebViews as one wants in the classical dynamic structure of Observers/Observables.



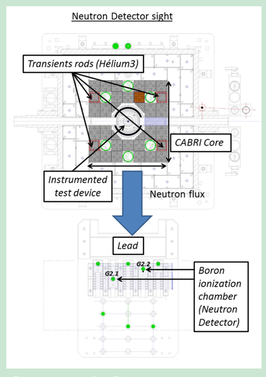
References

[1] Ramnath Vaidyanathan, Yihui Xie, JJ Allaire, Joe Cheng, and Kenton Russell. *htmlwidgets: HTML Widgets for R*, 2018. R package version 1.3.
[2] Htmlwidget for r - gallery. http://gallery.htmlwidgets.org/. Accessed : 2019-02.
[3] Carson Sievert. *plotly for R*, 2018.
[4] Simon Urbanek. *rJava: Low-Level R to Java Interface*, 2018. R package version 0.9-10.
[5] L. Pantera and P. Querre. Development, Automation, and Validation of a Numerical Methodology to Assess the TOP Onset for the RIA CABRI Experiments. *Nuclear Science and Engineering*, 189(1): 56–68, 2018.
[6] O.I. Traoré, L. Pantera, N. Favretto-Cristini, and S. Viguier-Pla. Emission Acoustique et Traitement du bruit : cas de signaux expérimentaux en contexte nucléaire. In *13e Congrès Français d'Acoustique*, CFA 2016, Le Mans, France, April 2016.

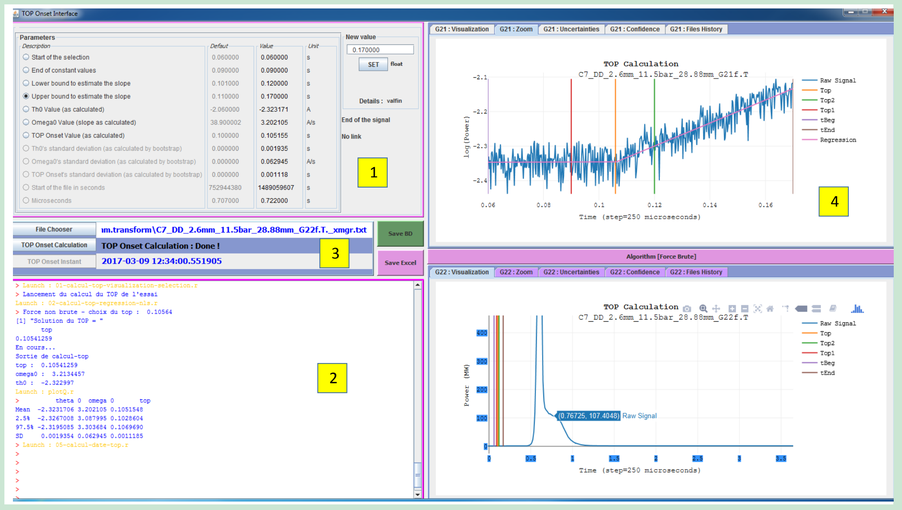
Idea and adopted solution

- JavaScript is what allows a user to interact with a web page. Hence it was embedded into a HTML document to create **dynamic visualisation** (e.g. allowing users to alter input values and see how outputs change virtually in real-time).
- JavaFX framework provides a WebView component used as an embedded web browser supporting JavaScript in a JavaFX application.
- R provides binding connectors from JavaScript to R:
  - **htmlwidgets** package [1] and its many libraries [2]. Worthy of note is **plotly** which can be applied on the top of the broadly used **ggplot2** graphs [3],
  - **Shiny** web application where visualisation tools can be updated by direct clicks on the web panel.
- Experimentalists will only have to write their scripts with htmlwidgets or plotly commands outside the HMI.

Example 1: Automation of the TOP onset assessment [5]



G12 and G22 neutron detectors location



Top onset calculation based on trial-and-error approach using the interactive representation of the G21 and G22 neutron detectors' signals

The TOP (Transient Over Power) onset is defined as the instant when the reactivity increases in the CABRI core. To detect it, measurements are done by the 2 experimental boron chambers closest to the core coined G2.1 (the most sensitive) and G2.2 (the most noised). The TOP onset is characterised by an exponential rise of the count rate. After a logarithmic scaling, the evolution follows a horizontal line before the TOP (steady rate) and an ascending line afterwards. The chosen model is:

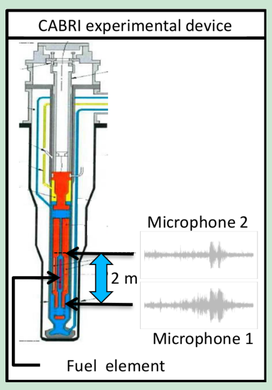
$$\ln[power(t)] = \begin{cases} \theta_0 & if\ x(t) \leqslant top \\ \theta_0 + \omega_0 \cdot (t - top) & if\ x(t) > top \end{cases}$$

The idea is to choose 4 values according to the curve shape detector signal at the prompt jump's beginning in order to roughly estimate θ_0 and ω_0:

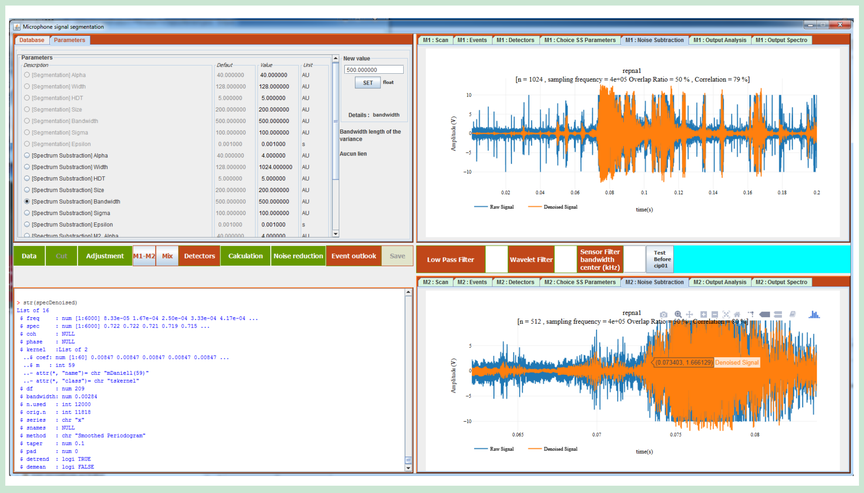
tbeg: beginning of the selection
top1: end of the first part of the selection
top2: beginning of the second part of the selection
tend: end of the selection

These 4 values are chosen by the user thanks to the touch-on graphs and the TOP is assessed by minimisation of the Residual Sum Squared.

Example 2: Microphones' signals' segmentation [6]



Microphones' location



Denoising microphone signal and segmentation

During the experimental phase, the behaviour of the fuel element generates acoustic waves detected by two microphones (piezoelectric AE sensors) placed upstream and downstream from the test device. Each recorded signal is post-processed through the following steps:

Step 1: Detection of events based on the evaluation of the signal-to-noise ratio after a spectral noise subtraction
Step 2: Storage of the events in a Relational Database PostgreSQL.
Step 3: Visualisation of the events and possibility to fine-tune their selection

To detect an event, a sequential analysis technique based on the moving variance of the signal was used. As unsatisfying detection of events may happen, the integration of a Shiny application enables the user to edit the detection manually.