HIGH LEVEL APPLICATIONS FOR SIRIUS

I. Stevani*, N. Milas, X. R. Resende, L. N. P. Vilela
Brazilian Synchrotron Light Laboratory (LNLS), Campinas, Brazil

Abstract

Has been decided that Sirius will use EPICS as its distributed control system and this year the development of its High Level Applications started. Three development frameworks were chosen for building these applications: CS-Studio [1], PyQt [2] and Matlab Middle Layer [3] (MML). Graphical user interfaces (GUI) and machine applications have already been designed and implemented for a few systems using CS-Studio and PyQt: slow orbit feedback, lifetime calculation and top-up injection. Specific Sirius data structures were added to the MML scripts in order to allow for EPICS communication through LabCA [4].

INTRODUCTION

Sirius is a Synchrotron Light Source Facility based on a 4th generation storage ring that is presently under construction at LNLS in Campinas, Brazil. The project initiated in 2008 with the first lattice studies and in the beginning of 2015 the building construction started. Machine commissioning is expected to start mid 2018 [5]. By this time, indispensable control systems for beam stability – such as tune measurement, orbit correction, injection – should be ready for use, and so should their high level applications (HLAs).

To achieve this goal, the Accelerator Physics Group started developing these applications this year and should intensify its activity with HLA next year, after most of the group's work related to machine components' specifications – that is currently taking most of the group efforts – is done. A Virtual accelerator with channel access server (VACA) [6] has been implemented to emulate the real machine, thus providing a testbed environment for high level software development.

APPLICATIONS FRAMEWORK

Figure 1 shows a schematic view of the application framework Sirius will use for HLA development. On a lower level, services are categorized in two groups: IOCs and machine applications. Both of them are Channel Access servers, the first providing control access to machine hardware while the second providing machine services. Machine applications developed so far were written in Python using PCASpy [7]. Virtual IOCs were implemented using EPICS Database on top of VACA in order to provide simulated fluctuations for machine parameters. EPICS V4 is another option for implementing machine applications that will be studied and discussed next year.

On top of Fig. 1 are the client applications. For basic process variable monitoring and settings with graphical user interface capabilities, CS-Studio was chosen due to its simplicity and fast learning curve. Most of the control system interface is expected to be implemented with CS-Studio. For software that requires elaborate logic, algorithms will either be implemented in the IOC level with simple driving GUI clients or using other options, such as PyQt or MML.

The control system can benefit from various support applications for storing and organizing system's parameters, IOC software, device and PV name lists, and so on. The plan is to use a few of the applications in development under the DISCS collaboration effort [8, 9]. The device naming and configuration modules of DISCS are currently being tested and used in the HLA development framework. Other DISCS modules, such as its logbook, IOC and machine save/restore services, will soon be tested as well.

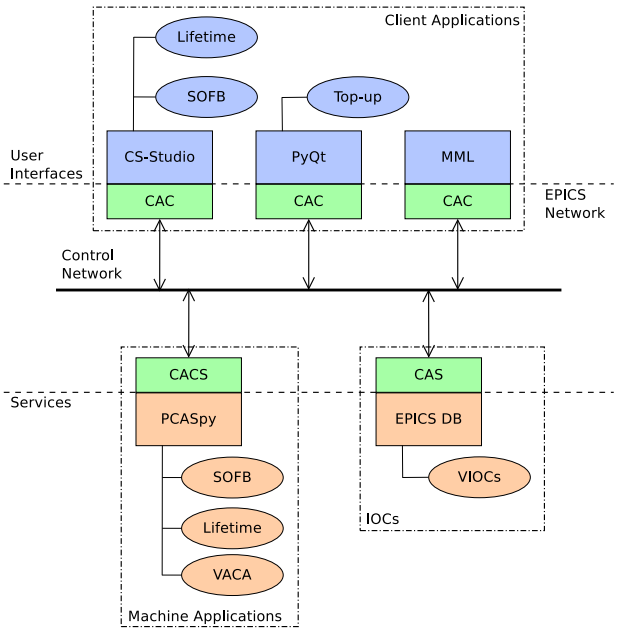

Figure 1: Applications framework schematic.

IMPLEMENTED APPLICATIONS

Some prototypes were built on these platforms listed before to help the developers familiarize with their functionalities and evaluate if they are suitable for Sirius requirements. The HLAs are presented next.

Slow Orbit Feedback

The slow orbit feedback (SOFB) system software is an example of a HLA that has already been written for Sirius. It consists of two modules: a machine application that reads BPMs and controls orbit correctors and a user graphical interface application.

* isabella.stevani@lnls.br

The first one is responsible for all data processing and control, such as orbit/response matrix measurements and orbit corrections based on singular value decomposition (SVD). Standard useful functionalities were implemented: device selection, response matrix and reference orbit configurations, variable-size buffers with orbit data for averages, optional inclusion of RF frequency in the correction loop and corrector strength adjustments. These functionalities were implemented in Python modules that are imported in a PCASpy SOFB server. Python threads were used to allow for uninterrupted CA service while data processing and calculations are performed in response to prior user interactions.

As depicted in Fig. 2, the second SOFB module is a user-friendly GUI application with a CA client layer underneath that allows for monitoring and controlling the SOFB system with embedded graphics tools, thus facilitating machine commissioning and operation. This module was designed in CS-Studio and it interacts with the SOFB machine application module through a set of PVs.

The prototype application is shown in Fig. 2. Its design was based on the orbit correction interface that is being used in the existing UVX storage ring at LNLS which has a proprietary control system. The main features of the prototype application are plot displayed with measured orbit, widgets set for manual correction, the machine application mode selection and widgets for configuring sampling parameters for orbit average calculations. Although not all functionalities have been implemented on the client application yet, the current version should suffice for initial commissioning stages of the storage ring. Selection of BPMs and correctors for the correction algorithm, as well as response matrix and reference orbit configurations, will be added in the near future.

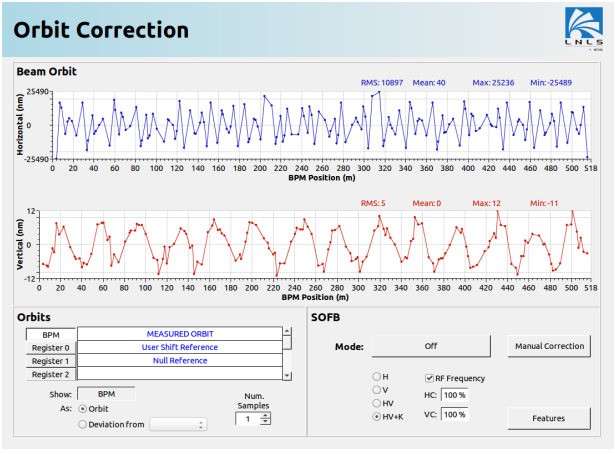

Figure 2: Orbit correction display application in the CS-Studio environment.

Lifetime Calculation

Another example of implemented HLA is the lifetime calculation. Just as in the case of the SOFB system, here two separate application modules were created. In order to simplify the user interface, the algorithm that computes beam lifetime was moved to a machine application implemented in EPICS DB, while the client application was designed in CS-Studio. The calculation method in the machine application is the same as the one already in use in the UVX storage ring and that has been proven reliable and stable over the years. The details of its implementation can be found here [10].

As always with beam lifetime calculations, there is a trade off between precision and quick response to lifetime variations. This trade off is realized by means of two input parameters, precision and sampling time, that define the time window and number of points for the lifetime calculation. In the user interface it is possible to display the lifetime in units of hours, minutes or seconds, as shown in Fig. 3. Being able to display lifetime in minutes or seconds might be useful specially on the early days of machine commissioning.

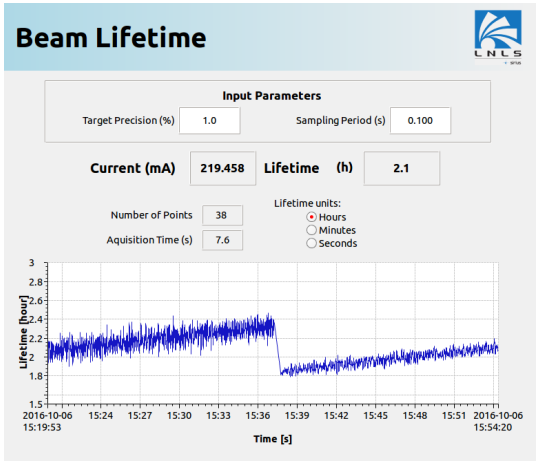

Figure 3: Lifetime display application in the CS-Studio environment.

Top-up Injection

After the commissioning phase, the Sirius injection system will operate in top-up mode and a control application will be necessary to conduct the required periodic injections. A simplified version of this HLA was developed to test the simulation of the injection process in VACA. It consists of a PyQt application that monitors the beam current in the storage ring and starts the injection cycle in order to mantain the current level within the desired tolerance. The connection between the Channel Access protocol and the Python language was done with the PyEpics package [11]. The user interface contains a display of the beam current over time and the fill pattern in the storage ring, as shown in Fig. 4.

In order to include this application in the Sirius control system, several functionalities must be added. For example, in this version, it is only possible to choose between multi-bunch or single-bunch injection mode. However, as the timing system will support the injection to any bucket, the final version of this HLA should allow the specification of any filling pattern.

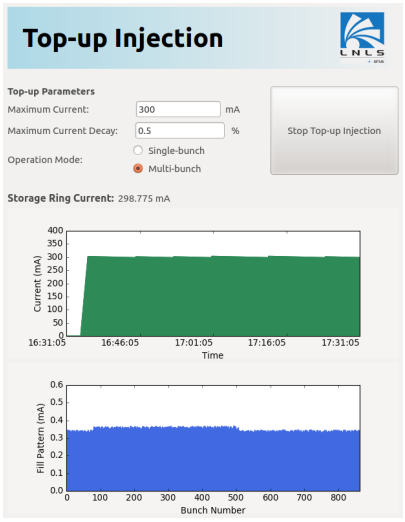

Figure 4: Top-up injection display application developed with the PyQt framework.

CONCLUSION

The development of Sirius' HLAs started this year. Three development frameworks were chosen and then tested on three important machine systems: SOFB, lifetime calculation and top-up injection. Results were promising and the plan is to continue improving these systems – adding missing functionalities, for example – and start the development of new HLAs such as LINAC gun and timing system controls, which are also fundamental to machine commissioning.

Also, has been decided that the control system could benefit from applications in development under the DISCS collaboration effort. Its device naming and configuration modules are currently being tested and used in the HLA development framework. Other DISCS modules will soon be tested as well.

ACKNOWLEGMENTS

The authors would like to thank L. M. Russo from the Diagnostics group for fruitful discussions on EPICS and for his help in configuring all services modules from DISCS which we have been testing.

REFERENCES

[1] Control System Studio, `http://controlsystemstudio.org/`

[2] PyQt, `https://wiki.python.org/moin/PyQt`

[3] G. Portmann, J. Corbett and A. Terebilo, "An accelerator control middle layer using MATLAB", in *Proc. PAC'05*, Knoxville, United States, May 2005, paper FPAT077, pp. 4009-4011.

[4] T. Straumann, "labCA – An EPICS Channel Access Interface for scilab and matlab", May 2008, `https://www.slac.stanford.edu/grp/ssrl/spear/epics/extensions/labca/manual/`

[5] A. R. D. Rodrigues *et al.*, "Sirius status report", in *Proc. IPAC'16*, Busan, Korea, May 2016, paper WEPOW001, pp. 2811-2814.

[6] X. R. Resende, A. H. C. Mukai, L. N. P. Vilela and I. Stevani, "Development of a virtual accelerator for Sirius", presented at the PCaPAC'16, Campinas, Brazil, Oct. 2016, paper WEPOPRPO21, this conference.

[7] PCASpy Documentation, `http://pcaspy.readthedocs.io/en/latest/`

[8] V. Vuppala *et al.*, "Distributed Information Services for Control Systems", in *Proc. ICALEPCS'2013*, San Francisco, United States, March 2014, paper WECOBA02, pp. 1000-1003

[9] DISCS, `http://openepics.sourceforge.net/about-discs/`

[10] A. J. Burns *et. al*, "Real time monitoring of LEP beam currents and lifetimes", in *Proc. EPAC'94*, London, England, pp. 1716-1718.

[11] PyEpics, `http://cars9.uchicago.edu/software/python/pyepics3/`