VIRTUAL VELA-CLARA: THE DEVELOPMENT OF A VIRTUAL ACCELERATOR

T. J. Price, R. F. Clarke, H. M. C. Cortes, G. Cox, D. J. Dunning, J. K. Jones,
B. D. Muratori, D. J. Scott, B.J.A. Shepherd, P. Williams
STFC Daresbury Laboratory, Warrington, U.K.

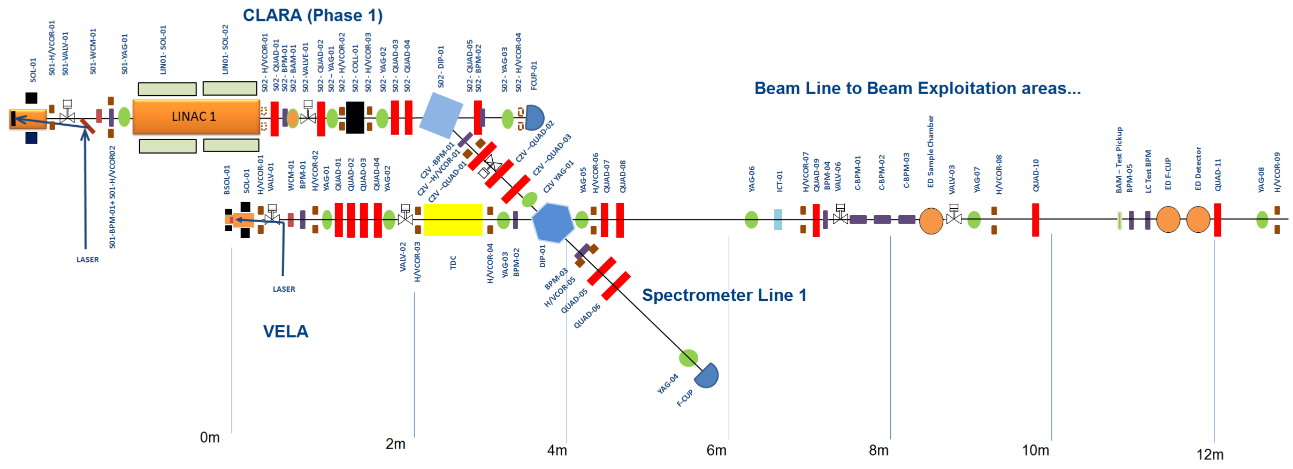

Figure 1: Schematic indicating sections of VELA-CLARA currently in the Virtual Accelerator. Not all elements have been added, but will in the upcoming future.

Abstract

A Virtual Accelerator (VA) has been developed to mimic the accelerators Versatile Electron Linear Accelerator (VELA) and Compact Linear Accelerator for Research Applications (CLARA). Its purpose is to test control room applications, run start-to-end simulations with multiple simulation codes, accurately reproduce measured beam properties, conduct 'virtual experiments'and gain insight into 'hidden beam parameters'. This paper gives an overview into the current progress in constructing this VA, detailing the areas of: developing a 'Virtual EPICS'control system, using multiple simulation codes (both particle tracking and analytic), the development of a 'Master Lattice'and the construction of a Python interface in which to run the VA.

INTRODUCTION

VELA and CLARA are adjoined linear accelerators based at Daresbury Laboratory, STFC, U.K. Once fully built, CLARA will be a 90 m long Free Electron Laser (FEL) producing ultra-short photon pulses [1] and provide beam to experimental stations for novel accelerator research and development. In building CLARA the development of a VA was deemed necessary to aid: underpinning simulation work, the production of software applications to automate procedures, the development of machine learning techniques to generate bespoke beam/laser setups, and preliminary testing of novel experiments.

The following requirements of the VA were set to provide support for VELA-CLARA and also aid in the development of a future UK X-FEL. The structure of the VA must be generic enough to accommodate another accelerator. The requirements were:

- A Virtual EPICS controls system to interact with the VA.
- The ability to use multiple simulation codes.
- Use of a Master Lattice which is simulation-independent and extendable.
- For all simulations to generate simulation-independent output files.
- The ability to carry out start-to-end simulations.
- Have dedicated outputs in the Virtual EPICS for hidden beam parameters.
- Run different simulation codes in a similar way.
- Interface between different simulation codes.

Hidden beam parameters are values that can be obtained by simulations but that cannot be easily observed without destroying the beam in a physical accelerator. This greater level of available information on the VA has powerful implications when constructing virtual experiments and machine learning applications.

Fig. 1 indicates what elements are currently available in the VA. In building the current VA of VELA-CLARA the following requirements have been achieved:

- Can interact with the VA using a Virtual EPICS control system.

- Can use two simulation codes (ASTRA [2] and SAMPL [3]).
- Have a framework that enables easy-use of multiple simulation codes.
- Use of a Master Lattice.
- Use of simulation-independent output files.

Automation of certain procedures (for instance momentum measurements and beam alignment) are already being developed into software applications and they are being tested with this VA.

STRUCTURE OF THE VA

The VA has been split into two main areas: the Virtual EPICS control system and a framework that runs simulations named the "Online Model" This overall structure is illustrated in Fig. 2.

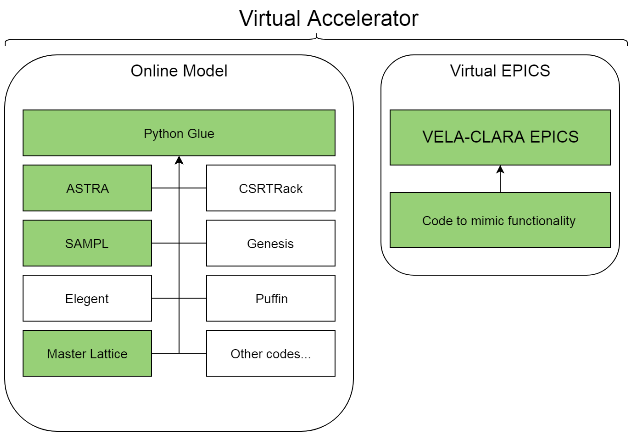

Figure 2: Block schematic indicating the components of the VA. The highlighted blocks show components already implemented.

Virtual EPICS (VE)

EPICS [4] is a Supervisory Control and Data Acquisition (SCADA) system used to present a uniform and simple interface to VELA-CLARA subsystems. It provides a standard way to send and read data via process variables (PVs) over a control network. The VE emulates the VELA-CLARA machine via cloned PVs from the EPICS used on VELA-CLARA and are hosted on a virtual Linux environment [5]. Within the VE system the functionality to read and set variables was enabled with added realistic effects, like the ramping of magnet currents, altering the positions of moving screens, and the option of adding noise to certain signals (see Fig. 3). The VE is maintained on a local server with the help of GitLab [6], and can be accessed for updates. The virtual Linux environment is set up on a users local computer using the third party software OracleBox [7]. The VE is broadcast on a local IP address with a specific port number so only the user can access that VE system. Prefixing all the VE PVs with the characters "VM-" further ensures that

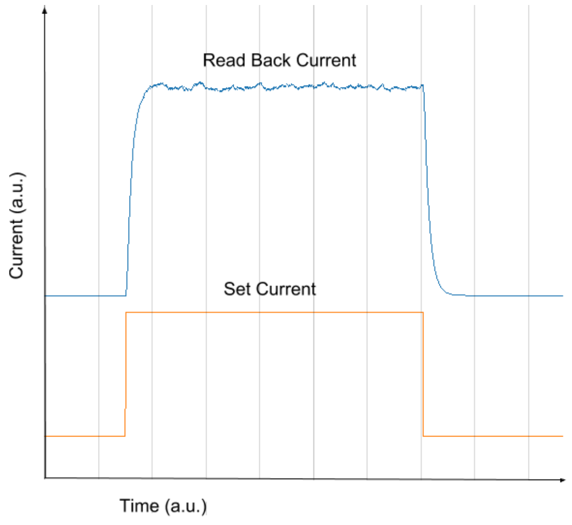

Figure 3: Demonstration of a VE solenoid current changing with time. The read back current can be seen to have an added ramp/decay time and noise. The level of noise can be adjusted.

the user does not accidentally alter PVs on the real machine.

Controllers To set variables in the VE high-level software controllers are used. The controllers are APIs that allow users to set and read EPICS PVs via Python scripts [8]. They were initially developed for VELA-CLARA but have been extended so that the user can also choose to set and read PVs on their own VE. The controllers have been written in C++ and wrapped up into a .pyd file in order to be used in Python [9–11]. As well as reading and setting PVs they provide a place to implement automated procedures and process data. Once these calculations and procedures are established in the controllers, they can be implemented more permanently in lower-level control systems.

Using these controllers allows procedures to be written and tested using the VE. Simulation runs with the Online Model can provide realistic output to the VE for the user during this testing.

Online Model (OM)

The OM [12] is a combination of simulation codes, a Master Lattice and the Python code (named 'Python Glue') that allows the user to run different simulation codes as simply and as seamlessly as possible. This section goes into these current components contained in the OM.

The Python Glue This code creates uniformity between all the simulations codes. Its job is to use the VE variables or settings files and conduct simulations. Depending on the type of simulation code this can involve building specific lattice files, creating particle distributions, standardizing output files and coordinate transformations between different simulation runs. The aim was to make the interface as model-independent as possible and flexible enough to start multiple points down the beam line (see Listing 1).

```python
import OnlineModel.SAMPL.v2.sampl as sampl
import OnlineModel.ASTRA.v2.astra as astra

# Define and initialise controllers.
# Use controllers to setup the VE
# in the same way as VELA-CLARA.

# Run OM
#start element name (RF Gun on CLARA line)
start = 'CLA-HRG1-GUN-CAV'
#stop element name
#(Screen in section joining VELA and CLARA)
stop = 'CLA-C2V-DIA-SCR-01'

ASTRA = astra.Setup(controllers=[])
ASTRA.go(start, stop, 'inputFile.ini')
# or
SAMPL = sampl.Setup(controllers=[])
SAMPL.go(start, stop, 'inputFile')
```

Listing 1: A Python script example of how to run different simulation codes using VE settings.

Listing 1 indicates the the key lines of code needed to run an ASTRA or SAMPL simulation. Behind the functions `go` and `Setup` for each of the codes, the Python glue code generates lattice files from the Master Lattice and the necessary settings in the VE. The simulation is then run and the output data interpreted and sent back to PVs in the VE (e.g. a beam's horizontal position through a BPM). The input file will determine the initial position, charge and momentum of the macro-particles making up a bunch. Depending on the simulation code used, the execution of creating lattice files and running simulations will be slightly different due to the differences in how certain codes are designed.

ASTRA ASTRA is a simulation code that has been used for some of the underpinning simulation work for VELA and CLARA [13]. The VA (and therefore ASTRA simulations) needed to have the flexibility to: start at two gun positions; go through any possible pathway through the machine; and for the beam to travel off the designed trajectory. ASTRA simulations were split up into straight sections, each in their own local coordinate system to help create such flexibility when running simulations.

During a simulation, if a bunch is tracked through a dipole, the simulation would end just before the bunch reached the dipole, get rotated, and inserted into a separate simulation which would take the bunch round the bend. If travelling along the ideal beam trajectory the bunch will have a centroid position x=y=z=0 upon exiting the dipole and be travelling along the z-axis.

SAMPL An analytic simulation code called Simple Accelerator Model in MatLab (SAMM) [14] was converted into Python code for the OM as an alternative simulation code to ASTRA. The converted version of SAMM was named the Simple Accelerator Model Python Library (SAMPL). Taking ownership of such a code has provided a place to test out novel beam tracking techniques and determine trends in VELA-CLARA faster than ASTRA. An example of one such technique being developed is an analytic approach to tracking particles through combined RF and Solenoid fields [15]. This approach is currently being included in SAMPL.

Space charge effects can be included in the tracking through certain elements within a simulation, however the speed decreases significantly as the number of particles increases. Due to the object-oriented construction of SAMPL adding in other quicker methods to calculate space charge effects is easy to implement and future work to do so is underway.

The Master Lattice This is the collection of files that defines all the machine elements within the VA. For each element there is data stored on its position, rotation, calibrations, field maps, and so on. The choice was made to use YAML files to store this data [16]. This markup language provides a readable, extensible structure in which data can be edited and added if needed. The Python glue uses these files to determine pathways through the machine based on the user's start and stop points in a given simulation.

FURTHER WORK

The upcoming improvement to the VA are:

- Moving the VE off virtual Linux environments and onto a sever.
- Extend the Master Lattice to include more of VELA-CLARA.
- Add Elegant, Genesis, Puffin, CSRTrack and any other codes needed.
- Add in other ways to calculate space charge effects in SAMPL.
- Get realistic RF signals and image signals.
- Standardise output files of all simulation codes.
- Add hidden beam variables to the VE.

REFERENCES

[1] J.A. Clarke *et al.*, "CLARA Conceptual Design Report", in *Journal of Instrumentation*, May 2014, Vol. 9, pp. T05001 (2014)

[2] ASTRA, `http://www.desy.de/~mpyflo/`

[3] SAMPL Source Code, `https://github.com/VELA-CLARA-software/OnlineModel/tree/master/SAMPL/sourceCode`

[4] EPICS, `https://epics.anl.gov/index.php`

[5] R.F. Clarke et al, "CLARA Virtual Accelerator", in *Proc. 16th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2017)*

[6] GitLab, `https://about.gitlab.com`

[7] OracleBox, `https://www.virtualbox.org`

[8] D.J. Scott, *Hardware Controllers: A 'Mid-Level' System for Controlling Hardware and Accessing Data on VELA / CLARA*, in Internal Notes of *ASTeC Accelerator Physics Group*, Nov. 2015

[9] Controllers Source Code, `https://github.com/VELA-CLARA-software/VELA-CLARA-Controllers`

[10] D.J. Scott, "Beam characterisation and machine development at VELA", *Proc. 7th International Particle Accelerator Conference (IPAC 2016)*, Busan, Korea, THPOW019

[11] Boost, `http://www.boost.org`

[12] Online Model, `https://github.com/VELA-CLARA-software/Online-Model`

[13] P. Williams *et al.*, "Developments in the CLARA FEL Test Facility Accelerator Design and Simulations", in *Proc. 7th International Particle Accelerator Conference (IPAC 2016)*, May 8-13, 2016

[14] SAMM Manual, `http://pcwww.liv.ac.uk/~awolski/SAMM/SAMM.pdf`

[15] B.J.A. Shepherd, `http://projects.astec.ac.uk/VELACLARAManual/index.php/Parasol`

[16] YAML Website, `http://www.yaml.org`