Developing Process Variables Necessary to Operate Simulacrum: The LCLS Accelerator Simulator

Mirian G. Juan Estrella[1], William Colocho[2]

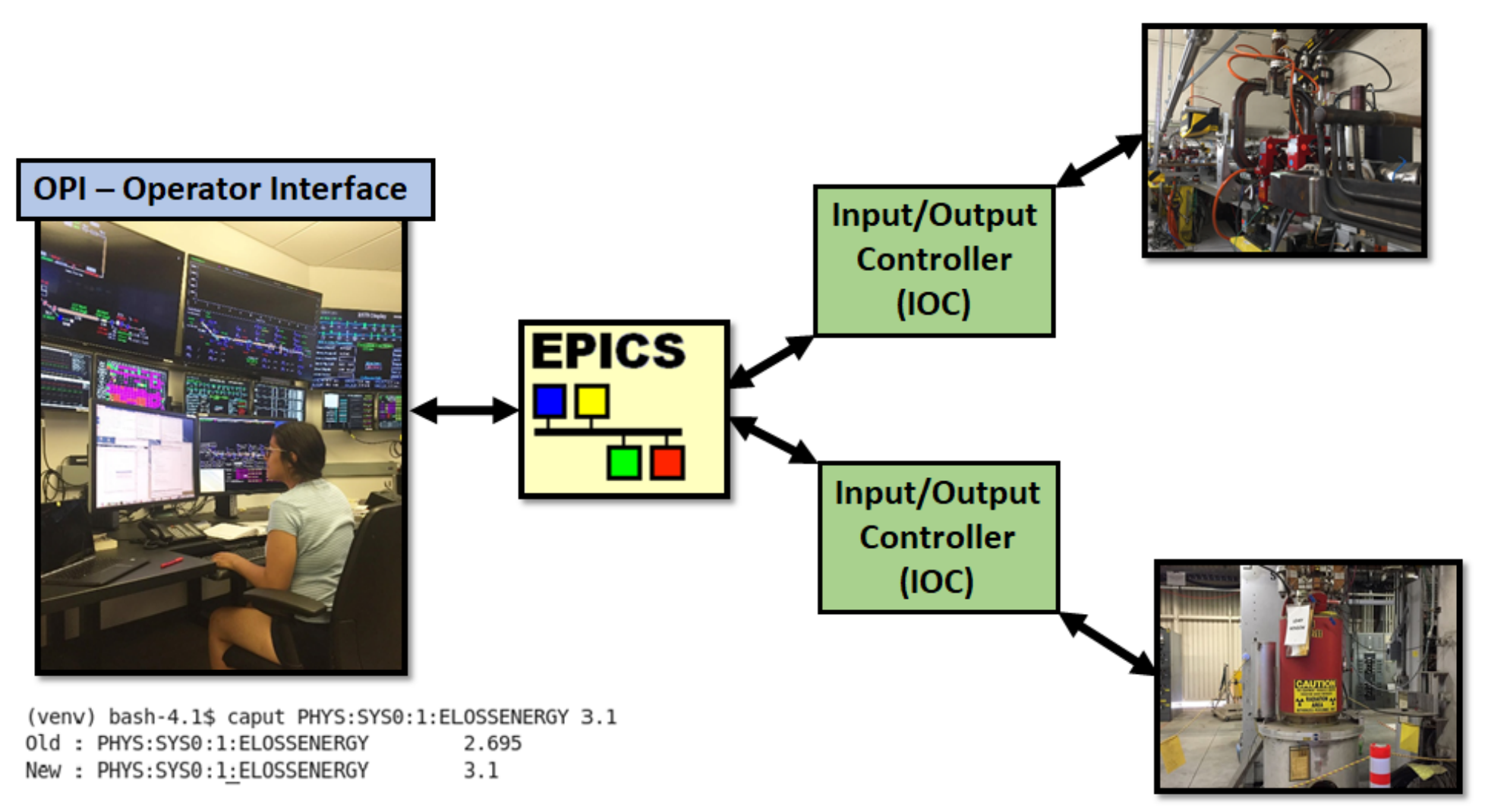
[1]Department of Mathematics and Statistics, California State Polytechnic University, Pomona
[2]Normal Conducting Linac and Free Electron Laser Division, SLAC National Accelerator Laboratory

Motivation

Simulacrum is a system that will simulate the Linac Coherent Light Source (LCLS) and its control system. This will allow future operators to train on a simulated linear accelerator and provide research scientists the opportunity to test future experiments. We will study the network systems in Simulacrum to write process variables (PVs) that will update graphical user interfaces from LCLS.

Background

Network communications with LCLS begin when the user on an OPI machine that contains EPICS, a computer software, sends commands to Input/Output Controllers (IOCs). IOCs are computers that communicate with each other to provide information about a specific (PV) that measures the value of various parts of a device (See Figure 1.)



```
(venv) bash-4.1$ caput PHYS:SYS0:1:ELOSSENERGY 3.1
Old : PHYS:SYS0:1:ELOSSENERGY        2.695
New : PHYS:SYS0:1:ELOSSENERGY        3.1
```

Figure 1: Network communications with LCLS

Conversely, Simulacrum contains services containing PVs that communicate via a high-performance messaging library called ZeroMQ (See Figure 2.)

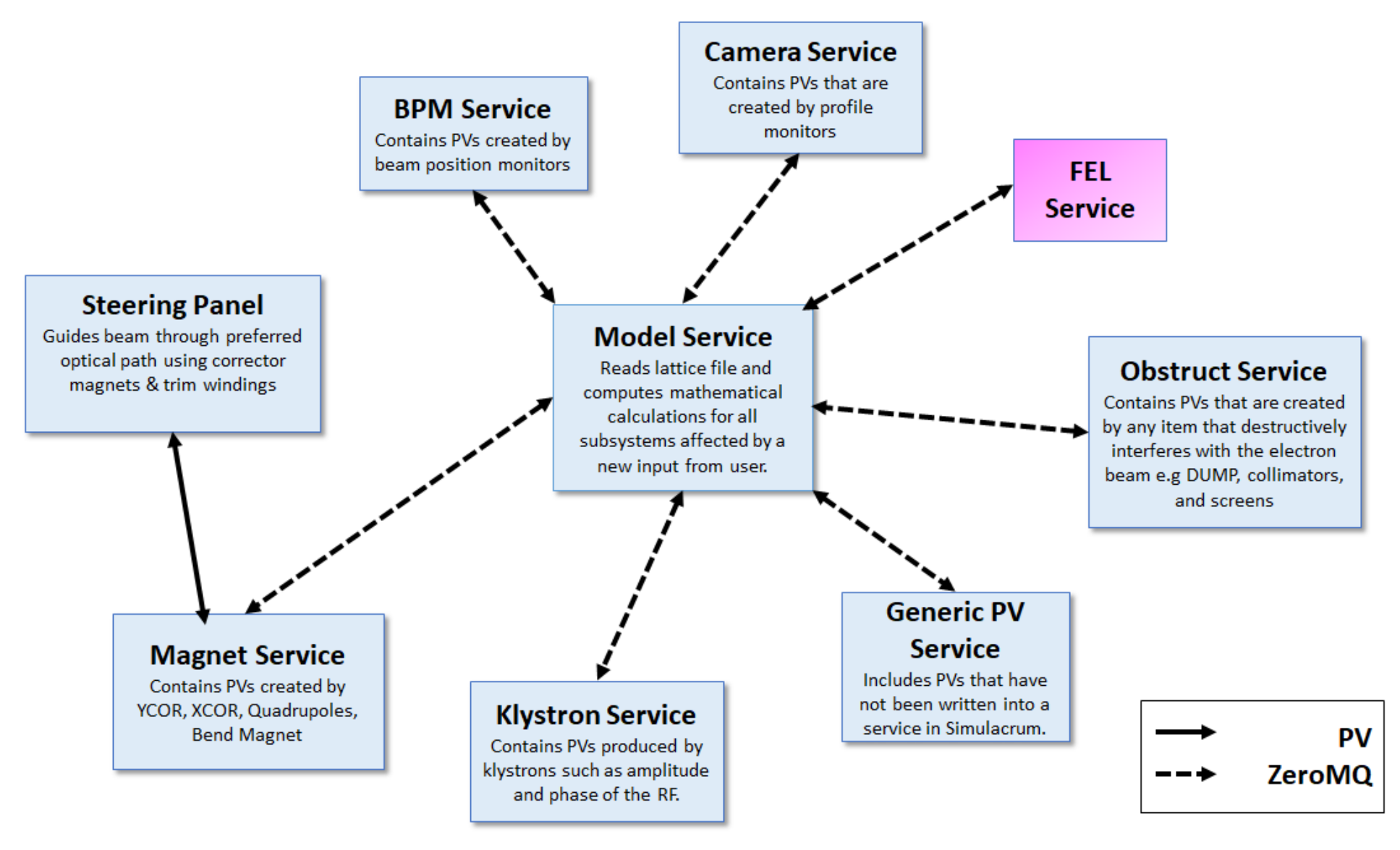

Camera Service
Contains PVs that are created by profile monitors
BPM Service
Contains PVs created by beam position monitors
FEL Service
Steering Panel
Guides beam through preferred optical path using corrector magnets & trim windings
Model Service
Reads lattice file and computes mathematical calculations for all subsystems affected by a new input from user.
Obstruct Service
Contains PVs that are created by any item that destructively interferes with the electron beam e.g DUMP, collimators, and screens
Magnet Service
Contains PVs created by YCOR, XCOR, Quadrupoles, Bend Magnet
Klystron Service
Contains PVs produced by klystrons such as amplitude and phase of the RF.
Generic PV Service
Includes PVs that have not been written into a service in Simulacrum.
PV
ZeroMQ

Figure 2: PV services in Simulacrum

A successful migration of LCLS PVs into Simulacrum will produce functioning displays that monitor and control a simulated LCLS.

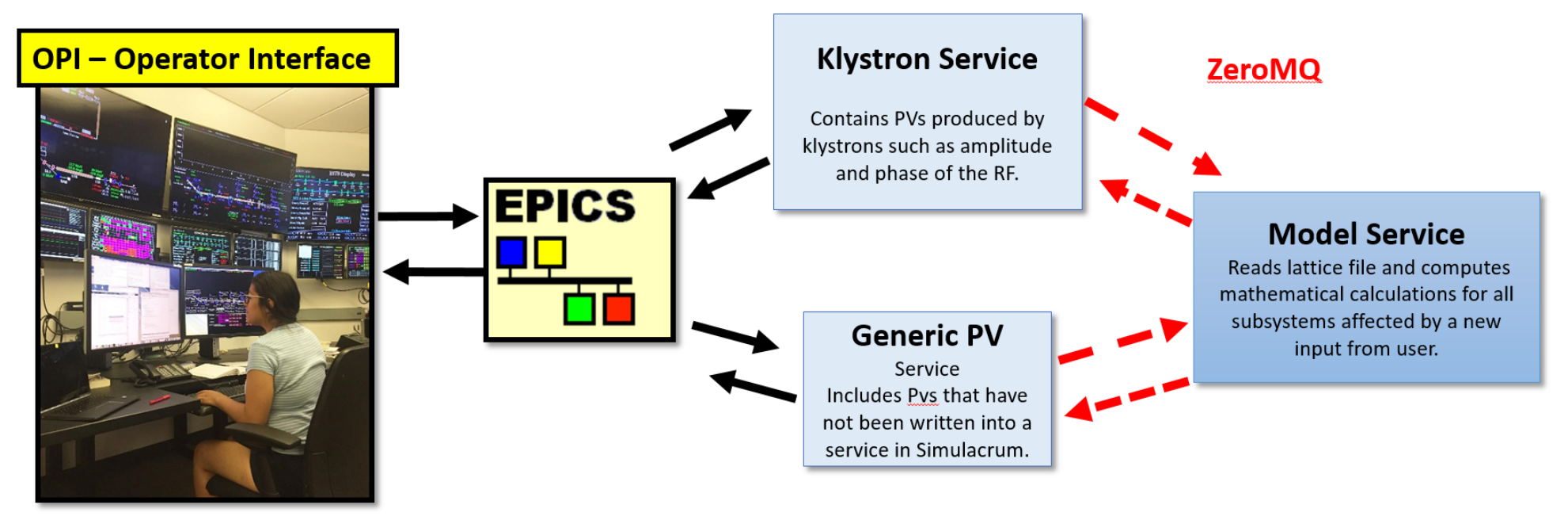
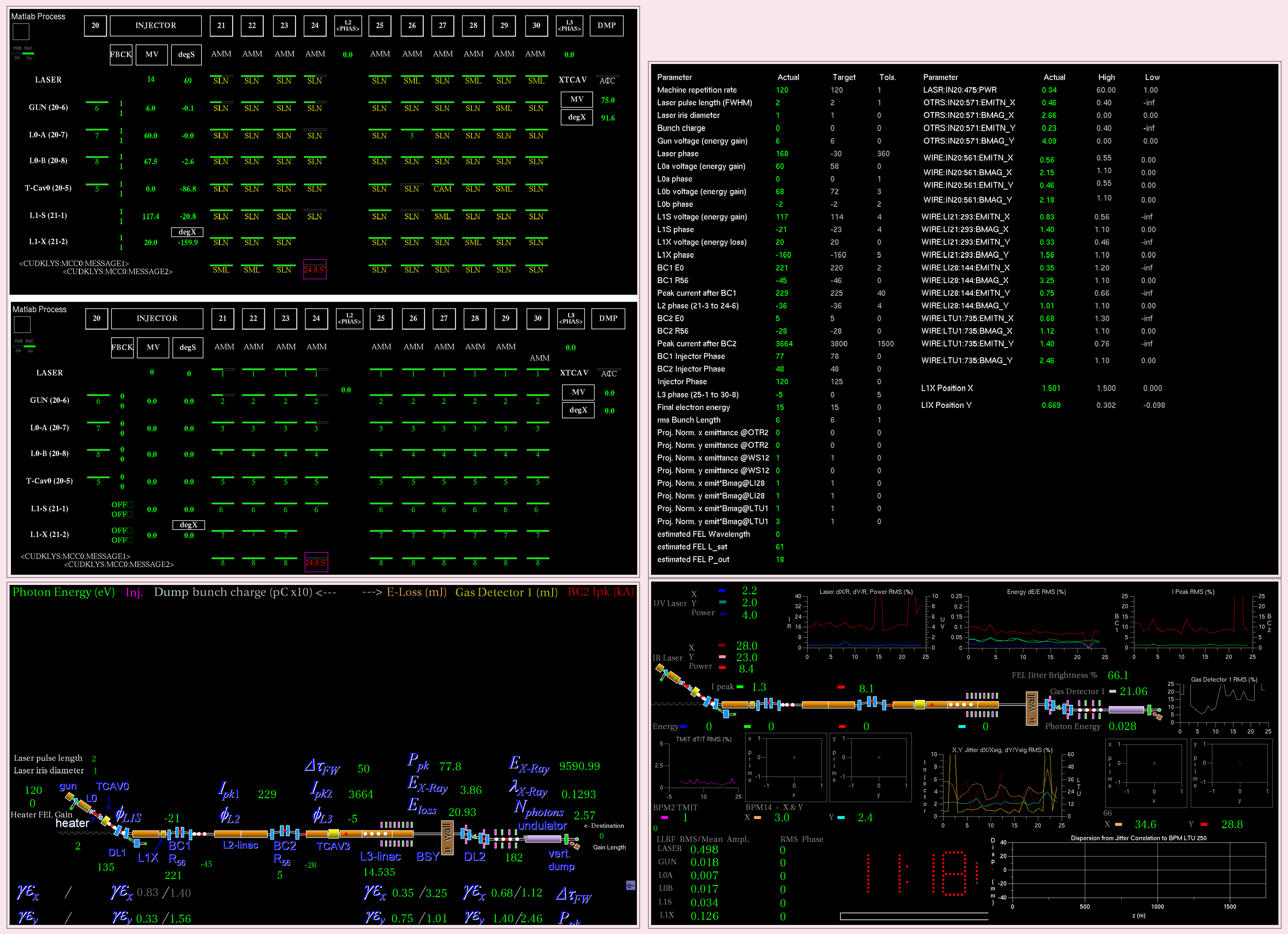

OPI – Operator Interface
EPICS
Klystron Service
Contains PVs produced by klystrons such as amplitude and phase of the RF.
Generic PV Service
Includes Pvs that have not been written into a service in Simulacrum.
ZeroMQ
Model Service
Reads lattice file and computes mathematical calculations for all subsystems affected by a new input from user.

Figure 3: Network communications in Simulacrum

LCLS Displays on Simulacrum



Parameter	Actual	Target	Tols.
Machine repetition rate	120	120	1
Laser pulse length (FWHM)	2	2	1
Laser iris diameter	1	1	0
Bunch charge	0	0	0
Gun voltage (energy gain)	6	6	0
Laser phase	168	-30	360
L0a voltage (energy gain)	60	58	0
L0a phase	0	0	1
L0b voltage (energy gain)	68	72	3
L0b phase	-2	-2	2
L1S voltage (energy gain)	117	114	4
L1S phase	-21	-23	4
L1X voltage (energy loss)	20	20	0
L1X phase	-160	-160	5
BC1 E0	221	220	2
BC1 R56	-45	-46	0
Peak current after BC1	229	225	40
L2 phase (21-3 to 24-6)	-36	-36	4
BC2 E0	5	5	0
BC2 R56	-28	-28	0
Peak current after BC2	3664	3800	1500
BC1 Injector Phase	77	78	0
BC2 Injector Phase	48	48	0
Injector Phase	120	125	0
L3 phase (25-1 to 30-8)	-5	0	5
Final electron energy	15	15	0
rms Bunch Length	6	6	1
Proj. Norm. x emittance @OTR2	0	0	0
Proj. Norm. y emittance @OTR2	0	0	0
Proj. Norm. x emittance @WS12	1	1	0
Proj. Norm. y emittance @WS12	0	0	0
Proj. Norm. x emit*Bmag@LI28	1	1	0
Proj. Norm. y emit*Bmag@LI28	1	1	0
Proj. Norm. x emit*Bmag@LTU1	1	1	0
Proj. Norm. y emit*Bmag@LTU1	3	1	0
estimated FEL Wavelength	0		
estimated FEL L_sat	61		
estimated FEL P_out	18		

Parameter	Actual	High	Low
LASR:IN20:475:PWR	0.34	60.00	1.00
OTRS:IN20:571:EMITN_X	0.46	0.40	-inf
OTRS:IN20:571:BMAG_X	2.66	0.00	0.00
OTRS:IN20:571:EMITN_Y	0.23	0.40	-inf
OTRS:IN20:571:BMAG_Y	4.09	0.00	0.00
WIRE:IN20:561:EMITN_X	0.56	0.55	0.00
WIRE:IN20:561:BMAG_X	2.15	1.10	0.00
WIRE:IN20:561:EMITN_Y	0.46	0.55	0.00
WIRE:IN20:561:BMAG_Y	2.18	1.10	0.00
WIRE:LI21:293:EMITN_X	0.83	0.56	-inf
WIRE:LI21:293:BMAG_X	1.40	1.10	0.00
WIRE:LI21:293:EMITN_Y	0.33	0.46	-inf
WIRE:LI21:293:BMAG_Y	1.56	1.10	0.00
WIRE:LI28:144:EMITN_X	0.35	1.20	-inf
WIRE:LI28:144:BMAG_X	3.25	1.10	0.00
WIRE:LI28:144:EMITN_Y	0.75	0.66	-inf
WIRE:LI28:144:BMAG_Y	1.01	1.10	0.00
WIRE:LTU1:735:EMITN_X	0.68	1.30	-inf
WIRE:LTU1:735:BMAG_X	1.12	1.10	0.00
WIRE:LTU1:735:EMITN_Y	1.40	0.76	-inf
WIRE:LTU1:735:BMAG_Y	2.46	1.10	0.00
L1X Position X	1.501	1.500	0.000
LIX Position Y	0.669	0.302	-0.098

Figure 4: Top two left: LCLSklystronCUD, top right: LCLSOperatingPointControl, lower left: LCLSMapCUD, lower right: LCLSJitterCUD.

Methods

The creation of Simulacrum is a group effort. Therefore many Simulacrum environments can be created to built it and are distinguished by unique port numbers (See Figure 5).

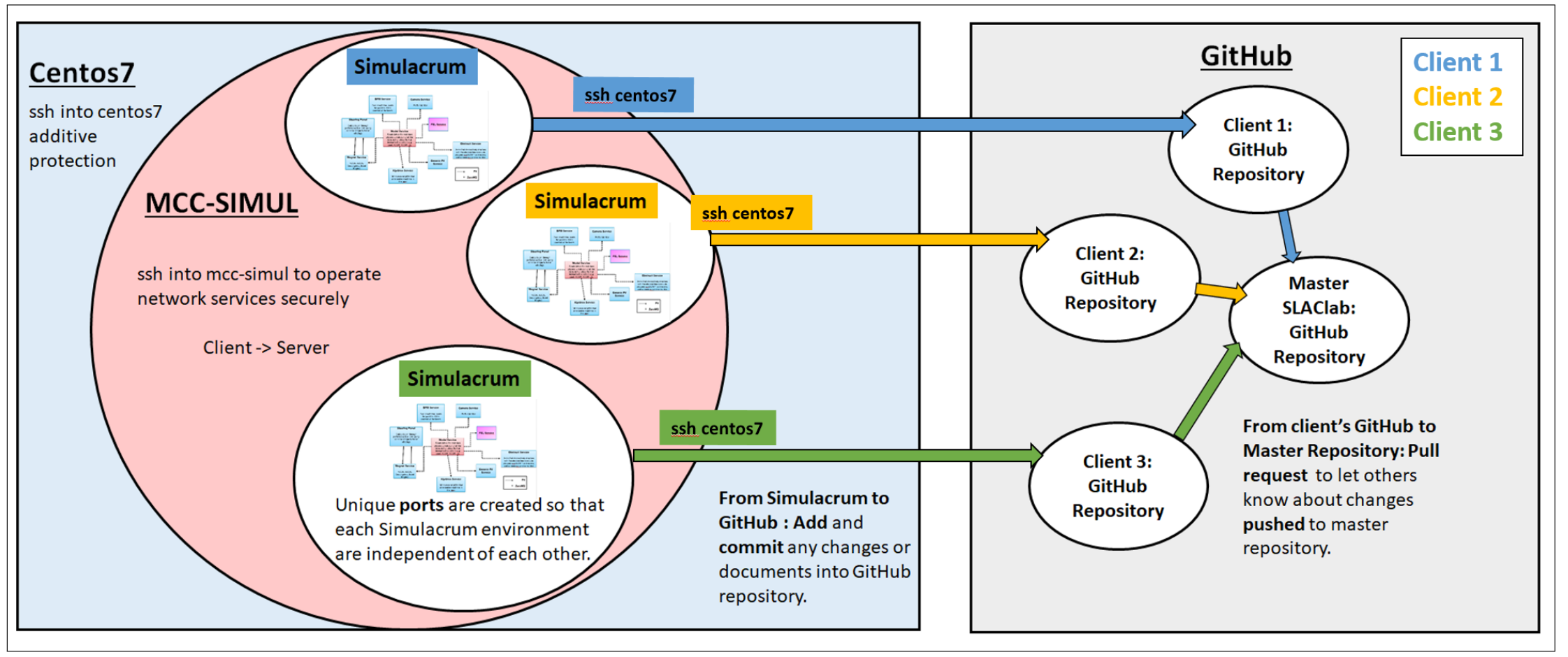

Centos7
ssh into centos7 additive protection
MCC-SIMUL
ssh into mcc-simul to operate network services securely
Client -> Server
Simulacrum
Simulacrum
Simulacrum
Unique ports are created so that each Simulacrum environment are independent of each other.
ssh centos7
ssh centos7
ssh centos7
From Simulacrum to GitHub : Add and commit any changes or documents into GitHub repository.
GitHub
Client 1
Client 2
Client 3
Client 1: GitHub Repository
Client 2: GitHub Repository
Client 3: GitHub Repository
Master SLAClab: GitHub Repository
From client's GitHub to Master Repository: Pull request to let others know about changes pushed to master repository.

Figure 5: Simulacrum environments & it's collaboration process

After creating a Simulacrum environment, we began consolidating PVs from LCLS and modifying MATLAB and Python scripts in Simulacrum to ensure that those PVs resided in the klystron_service and generic_pv_service (See Figure 2.)

Methods Continue

Next, to populate the PVs in the klystron and generic PV with valuable data we used the following methods:

- write MATLAB code to call the `caget('pvName_here')` command for all PVs and
- write a Python script that imported sensible data from the archive.

Since LCLS was in operation on December 12, 2018, we chose to run hourly data to produce a movie of the events on that day.

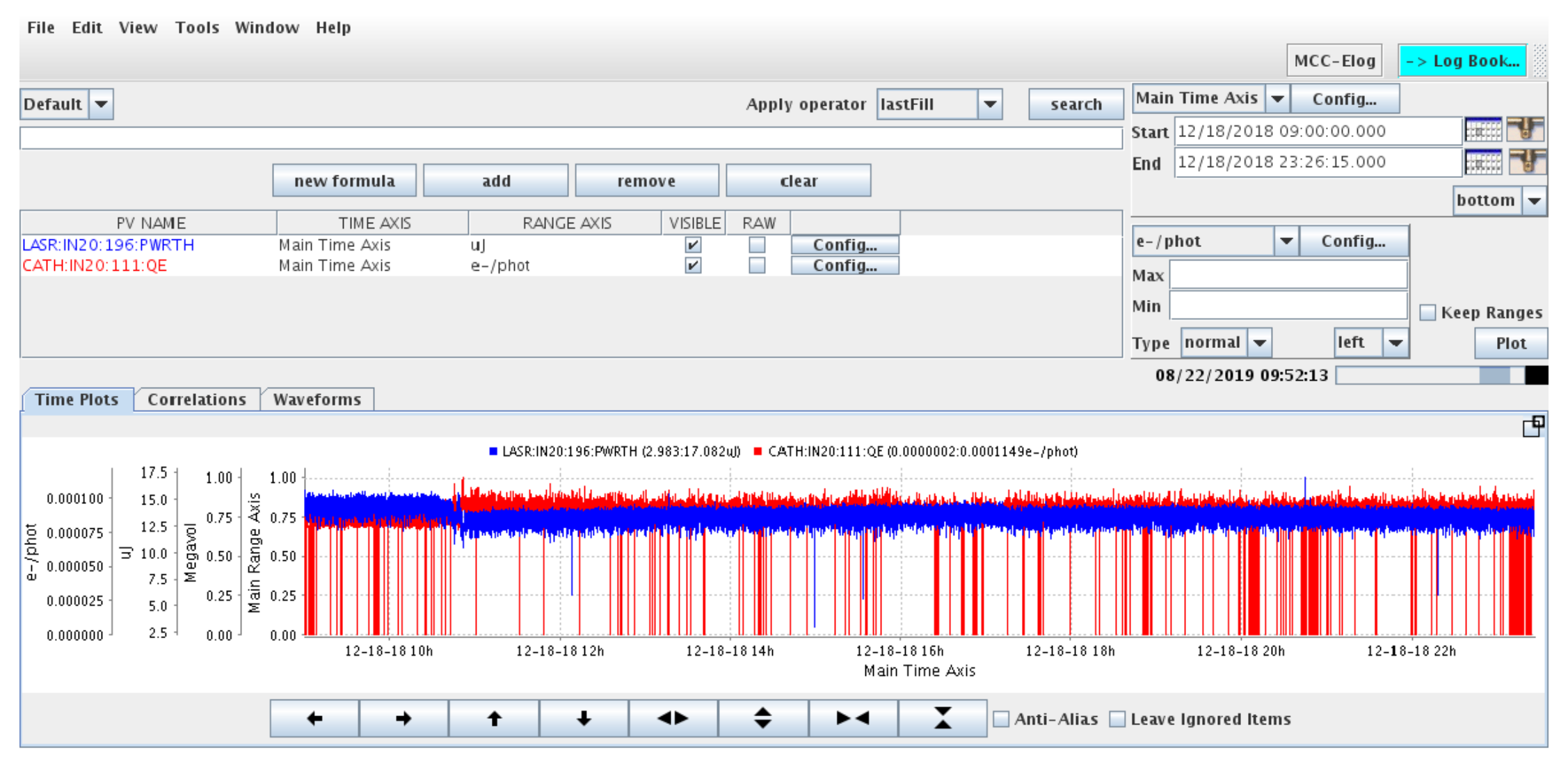


Figure 6: Archive Viewer

Results

After running the model, klystron, and generic services and other MATLAB and Python scripts, we get can observe four graphical user interfaces (See LCLS Displays on Simulacrum). For instance, the LCLSJitterCUD is a snapshot of the movie simulated by data stored in the archive on December 12, 2018. This can be useful to an operator that is interested in studying how certain parameters in LCLS behaved on that specific dated. It is also important to note that the PVs populated from either of our methods serve as initial values for the PVs. Therefore, if an operator calls a `caput('pvName_here') new_value` command on a specific PV, then the OPI machine will send that message to the service where the PV lives. It will update. Then, the model service will be notified and will read the lattice file to recalculate its physics parameters. The model service will then send messages back to the services so that updates can be made. If these communications are successful, then the LCLS displays in Simulacrum will allow users to simulate the linac with accuracy.

Discussion

Our results lead to several questions left for discussion:

- What happens when a PV does not have a value in the archive or in production?
- Where will the values come from?

We encountered PVs that had to be populated with fake data, but data that still made sense. For instance, PVs can take in many values like integers. float numbers, vectors, strings, and bytes. Therefore, understanding the role of these PVs is important. A possible solution to addressing such unique PVs may be in creating a new service that houses them. This may make it possible for services that currently do have PVs with values to communicate efficiently.

Figure 7: Accelerator Control Room, Building 52.

Acknowledgements

I am very grateful for the invaluable direction and mentorship provided by William Colocho. Moreover I am very thankful towards the 2019 STEM Teacher and Researcher Program (STAR), which could not have been possible without the support from Chevron (www.chevron.com), the National Science Foundation through the Robert Noyce Program under Grant 1836335 and 1340110, the California State University Office of the Chancellor, and California Polytechnic State University in partnership with SLAC National Accelerator Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funders.