What is Simulacrum? (Bmad, Tao, Services, and PVs)

Scope of this guide

This is intended for people who may have heard about simulacrum at SLAC, but want to become more familiar with the project.

This article will discuss the difference between Bmad, Tao, simulacrum services, and PVs.

At the end, I include links to other digital sources for learning more.

How to read this guide

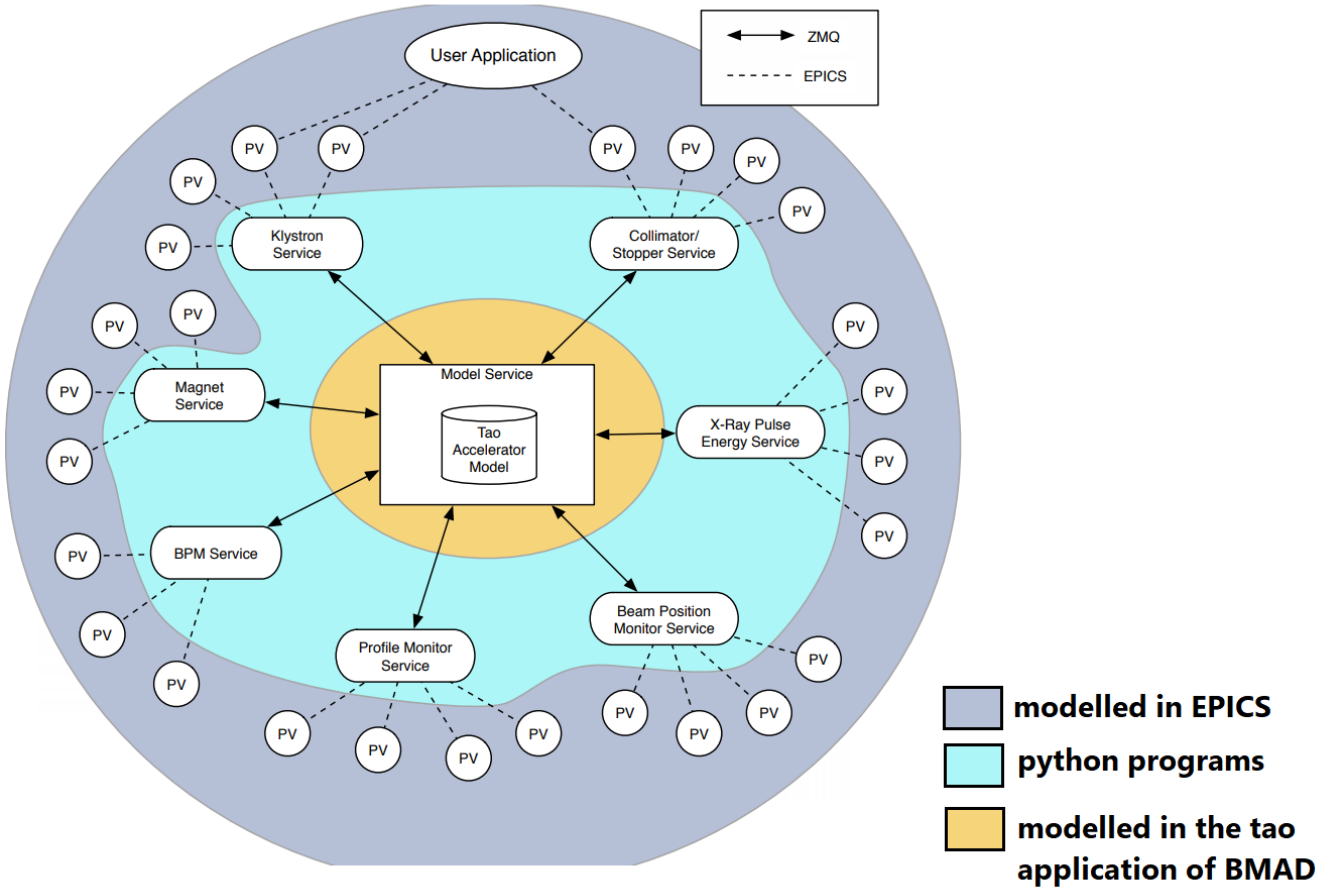
This article was written in Spring 2021. Depending on when you view this, simulacrum may be used in ways that are not implied in this article. Simulacrum is first and foremost a tool for researchers, and its functionality may change over time depending on the needs of the research and operations.

Simulacrum: What, where, how?

Simulacrum is a **private**, **simulated** instance of a particle accelerator. If you work with live machine data, you know that it is possible to play with beamline parameters from the comfort of your computer, usually by adjusting the configuration of some instruments over remote control systems. However, getting the beam to yourself is hard to come by, and some experiments or calibrations might be preferable to run on a simulation of the accelerator. Operators often need a way to test their software or produce their own data in a way that does not require beam time. Simulacrum tries to address that demand by creating a mathematical simulation of the beam and programming virtual klystrons, undulators, and other instruments to interact with that beam in a way that should feel intuitive to operators here at SLAC. Let's break down how this is designed and how you can get involved.

3 Layer of Simulacrum: EPICS, python services, and tao

Figure 1.



Above is a visual guide to the three "layers" of simulacrum. Likely, your work with simulacrum will be predominantly confined to one region, and it's important to know where the scope of your project begins and ends.

Here, I will go over the differences between the three regions and what their work consists of, so you can better understand what information is essential to your project.

Bmad and the Tao Accelerator Model

According to the access site for Bmad and Tao (link), "Bmad is an object oriented, open source, subroutine library for relativistic charged-particle dynamics simulations in accelerators and storage rings". More on that in a moment. Meanwhile, Tao is an application that runs on Bmad's internal framework to make Bmad modelling compatible with other programs (like simulacrum!).

Think of Bmad like a big excel file containing hundreds equations. These equations describe all the physics properties of the particle beam and how they may evolve through time. Tao uses these Bmad equations (often called Bmad lattice files) to create the initial conditions for a beam simulation and calculate any changes over the beam's lifetime. Furthermore, Tao knows about different kinds of instruments in our facility which may impact the physical properties of the beam (undulators, klystrons, etc.), and asks Bmad to describe the interaction between those instruments and the beam. Tao uses the output of the Bmad equations to update its model as it encounters various beam-altering instruments.

Simulacrum Services: An Operator Interface for Human-Friendly Control

So if Tao can simulate all the possible beam interactions, why do we need anything else? **Beam interactions alone do not tell a human-friendly story**. T ao is mostly concerned with the beam's immediate surroundings, and does not understand how the machines were configured to *achieve* their impact on the beam. In other words, Tao doesn't bother modelling all the work the operators do to fine-tune machine output. It only understands physics attributes that impact the beam. This is too abstract, from a controls standpoint. We want the simulation to look and feel like controlling the real machine. Operators should be able to use the same controls system software whether they're working in a simulation or in real life. This is the goal of simulacrum. Tao needs supplemental programs to make it compatible with common controls.

Simulacrum introduces 2 new types of programs: a model service and a machine service. These are written in python and can be found on the simulacrum github (link). The **model service** tells Tao which beamline to model, how much jitter to apply, if any, and handles the twiss parameters. The **machine services**, on the other hand, describe certain *classes* of instruments, not the beam itself. For instance, we have different services for undulators and beam position monitors. The machine service provides *generic* instructions for each class of device, then tells Tao to apply those instructions for each device in that class. As a result, each device gets its own set of PVs (more on those soon) to describe its relationship with the beam and viable responses to operator input. Critically, each device is initialized in a data format that we use in real-world remote control system, as we'll see in the next section of this article.

Despite describing different machines, all simulacrum machine services should read pretty similar to each other, lending themselves to iteration and expansion. This is possible because the services outsource most of the physics to tao. Services configure the relationships between variables, capable of multi-step interactions (See Simulating PVs 2: putter functions). This expands the scope of what is easy to model by coordinating properties into human-readable concepts such as status messages and pre-programmed modes.

PVs & EPICS Domain

PVs (process variables) and EPICS occupy the lavender region of Fig. 1. EPICS software are the backbone of many control systems at SLAC, and you can read more about it on the EPICS website (https://epics.anl.gov/about.php). Process variables are anything that an operator can read and/or write to a real-life instrument in the linac. This can include a beam position monitor's (BPM) x and y coordinates for the beam position, a status message on a cavity, the B-value of a quadrupole, or many more. One EPICS application, caproto, is a robust library for creating PVs that are shareable on EPICS servers. More on caproto here: https://nsls-ii.github.io/caproto/iocs.html#why-write-an-ioc-using-caproto.

PVs are typically written with a device's "control system name" (also called device name), followed by a suffix. The control system name typically tells you something about the devices location in a series of related devices, for example:

- ACCL:**L1**B:**02****5**0
    - Cavity located in the **L1** section of the LCLS-II beamline. The **5**th cavity on the **2**nd cryomodule.

As you can see, device names usually have several rules that govern their naming convention. LCLS-II alone has hundreds of proposed PVs and ways to organize them (LCLS-II Naming Conventions). You can find a database of in-use EPICS devices on the Oracle database: Looking Up Devices. We can add other terms to the end of this device to make PVs that characterize our device:

- ACCL:L1B:0250:PHAS
    - Phase (numerical)
- ACCL:L1B:0250:GRAD
    - Gradient (numerical)
- ACCL:L1B:0250:IS_ON
    - On/off status of the cavity (boolean)

PVs make up the digital interface of devices, allowing operators to control devices remotely over the EPICS server. If we run an EPICS server privately (so there is no cross-talk with the real machine), and simulate ***virtual devices*** to respond to PV input, then we have the basis for simulacrum user controls. To put it another way, simulacrum is designed with the same controls as the real thing! For more about EPICS devices accessible to simulacrum modeling, read the article Using tao_shell.py.

Conclusion

Simulacrum is a project that bridges physics modeling with our control systems at SLAC, resulting in simulations that look and feel like data from the real machine. Hopefully, this article will be a jumping-off point to check out simulacrum's github (https://github.com/slaclab/simulacrum) and reach out to the contributors to get involved. You can ask for an invitation to the simulacrum channel on Slack if you have any questions about this article or any other article in this series.

Accelerator Modeling Workshop Talks from Spring 2021:

- (link) "Introduction to Accelerator Beam Dynamics for Modeling", Tor Raubenheimer & "LCLS Infrastructure Data Systems", Greg White (April 30th, 2021)
- (link) "Use of Simulacrum", Matt Gibbs (May 7th, 2021)
- (link) "Bmad, Tao and Related Tools", Chris Mayes, Jackie Garrahan (May 7th, 2021)
- (link) "Bmad Live Model from Python and Matlab", Matt Gibbs, Greg White (April 30th, 2021)

Also see the article "A Model-Based Simulator for the LCLS Accelerator" (2019) by Matt Gibbs, William Colocho, Jane Shtalenkova, and Ahmed Osman as well as the accompanying presentation:

Article: https://inspirehep.net/files/b12f0b55fa9dd0a91de1c9ea4dc57d2b

Presentation: http://accelconf.web.cern.ch/icalepcs2019/talks/tucpl04_talk.pdf

If the links are broken, reach out the authors via SLAC email.