Brainstorming Document for Middle Layer Collaborative Development

Authors: L. Nadolski (LSN). S. White (SW), S. Liuzzo (SL), L. Farvacque (LF), etc…

History:

* 2 November 2023, LSN, Document Creation
* 10 November 2023, LSN, SL
* 23 November 2023, SL, SW, LSN
* 5 December 2023, ???

# Introduction

Presently, the Matlab Middle Layer[1] (MML) software is used by most synchrotron light source laboratories to link the beam dynamics simulations with commissioning and operation activities. It acts as a **digital twin**. All required high-level software (from magnet calibration to storage ring optics tuning (LOCO[2] algorithm)) is developed based on beam dynamics simulations in the Accelerator Toolbox (AT[3]). The same code (including user interfaces) is used directly with the real beam by simply switching a global flag ("physics"/"machine"). This tool has tremendous advantages for the development of new storage rings and for their commissioning and operation. Moreover, the tools developed in one laboratory within MML are immediately usable by all other laboratories using MML, thus fostering the exchanges between accelerator physicists.

However, the Matlab Middle Layer, developed in the 90's, is becoming difficult to maintain in a collaborative way. The last non-laboratory specific version update dates from 2017: as it does not benefit from modern coding techniques and libraries and does not implement scientific open data management it will soon become obsolete and put our facilities at risk. Updating and maintaining it would require re-writing of most code and its user interfaces.

Python, on the other hand, is open-source, is the most widely used code in the world and benefits from extensive open-source scientific libraries integrating modern algorithms. Choosing Python to replace MML allows reaching a much wider audience, simplifies interfaces to many other codes used by our community, and will help in the future developments critical for the new fourth-generation light sources. In the past years, ESRF has led the developments of Python AT (*pyAT****[4]***), making it a solid foundation for the definition of a new, advanced, and open-source “Python Middle Layer”. At the DIAMOND light source, Python Toolkit for Accelerator Control (*pyTAC*[5]*)* and pyAT Interface for pyTAC (*atip*[6]*)* provide the seed to build an accelerator-oriented software library very similar to MML based on Python (presently linked to EPICS and to the local storage ring layout). DESY and ESRF joined efforts to translate error setting and correction functions from Simulated Commissioning (*SC*[7] based on Matlab AT) to Python (*pySC*[8]), extending the tuning possibilities presently available in MML and in particular reproducing the features of LOCO optics correction in Python language. Most building blocks for a new Python version of MML are being developed independently in several laboratories.

The newly designed **Python Accelerator Digital Twin** will then be a common work among all institutes, to put together the existing tools and create the missing components (such as graphical interfaces and GPU support). The common project of a **Python Accelerator Digital twin** would profit from all these recent developments and have all the features of the old MML software. It would also: 1) include the possibility to be used on large computing clusters for automated commissioning simulations, 2) profit from recent beam dynamics developments to be faster and more precise than the existing MML (GPU, analytic Jacobians[9]) using state-of-the-art programming techniques and state-of-the-art accelerator-oriented library, 3) enable seamless connection to our facilities' control system (EPICS, TANGO, DOOCS, and so on).

The **Python Accelerator Digital Twin** represents a virtual machine that could be set up to simulate the accelerator in real-time. Its use could then be extended compared to MML to monitor characteristic quantities (orbit, tunes, optics, etc..) and their drift concerning the expected values and feed Artificial Intelligence (AI) models that can be used to support operation or efficiently detect or predict failures. The operation of accelerators will strongly benefit from such modern numerical tools.

# Current Situation

Overall, in the accelerator community, there is a strong push to switch to Python, and this is why a proposal to the LEAPS consortium was initiated by ESRF to obtain EU funding in 2025.

However today, many laboratories do not have yet the means to switch to full Python.

* Hybrid (Matlab-based and Python-based) use is more and more promoted in many facilities
* There is a strong need to maintain interoperability Matlab/Python and identify the required resources for it
* It will be profitable to identify the obstacles (applications, training, expertise) to switch to Python

Regarding Matlab tools, we are facing possible difficulties since we have fewer and fewer MATLAB developers which is going to be an issue to maintain interoperability if the Python community keeps growing as it is now

Concerning the organization of the meeting, we have 2 separate working groups: WG1: Accelerator Toolbox developments (AT) and WG2: MiddleLayer developments. The AT working group's first meeting will be held separately. S. White will organize the meeting.

The host laboratory for the next pyML/pyAT meeting (twice a year) still needs to be defined as well as a draft program.

# Machine Middle Layer Working group

**Organizer: Simone Liuzzo**

## Goals

* Establish governance (5-6 people, from different labs, which could be the same as the AT one)
* Building a roadmap (ROADMAP) and reasonable priorities according to our resources
* Identify critical competence (software architect, developers, CD/CI, etc.)
* Preparing documents for LEAPS funding
* Choosing an acronym (e.g. python Middle Layer (pyML, used in this document, but to be changed as could confused with pyMachineLearning), python Accelerator Digital Twin (pyADT), python Control Agnostic Synchrotron Agnostic Tuning and Twin (pyCASATT, or CAST))

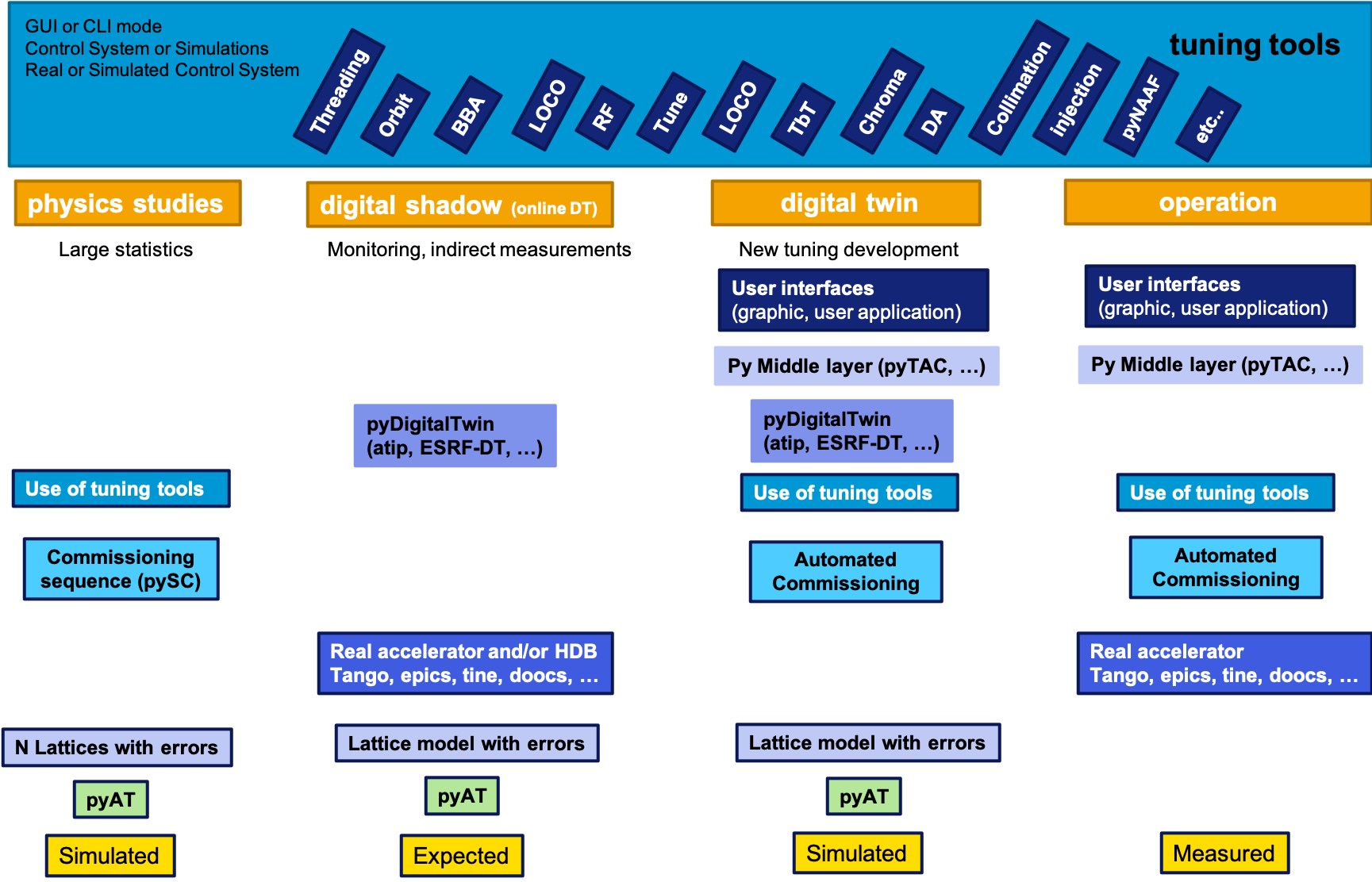
## Defining what Python middle layer will be

### Follow a very similar philosophy as the MML:

* + Having transparent access to a simulator, a digital twin, and the real accelerators
  + Providing a library of high-level generic functions
  + Provision of adapted generic GUIs
  + Link to dedicated layer for the controls (TANGO, EPICS…)

### Middle Layer Architecture (Software architecture)

* Modern layering for easier maintenance, decoupled by domain experts
  + Facilitate identification of maintainers, and reduce stress by empowering layers instead of the whole architecture
  + Interaction with other packages
    - Enabling the integration of AT in the Middle Layer such that the AT maintenance is well separated from that of the middle layer (software quality assurance)
    - pySC, [pyNAFF](https://pypi.org/project/PyNAFF/) (GNU GPL v3), etc.
    - A future pyLOCO featuring all MML LOCO features and more (analytic Jacobians and ORM)
    - possible scheme (python AT only):



* + Proposal by P. Schnizer (Bessy)?
* Is DIAMOND development a good starting point? Python Toolkit for Accelerator Control (pyTAC ) and pyAT Interface for pyTAC (atip)

### Wanted improvements compared to MatlabMiddleLayer

* A better separation between what is common between facilities and what is facility-specific
* Fine-tuning to each installation without breaking the code (true for both AT and middle layer)
  + Facility-specific settings, control (performance)
    - *Depends on the complexity and size of the installations to be controlled*
    - *depends on the IT infrastructure of each plant*
    - *Tuning pyAT (needed forks/branches) to your accelerator facility specificities in a more generic way (true for AT, pyML, etc). Would be nice if cross-compatibility with pyAT is kept (unit tests: new pyAT releases are not harming the pyML infrastructure).*

The table below gives an incomplete list of features to keep, improve or add to pyML (temporary acronym) compared to MML. Priority 1 is the highest. It means that it is needed by the other developments.

| **FEATURE** | **Status** | **Description** | **priority** | **Contributors** |
| --- | --- | --- | --- | --- |
| Control system agnostic python middle layer | wished | e.g.: pyTAC | 1 |  |
| Easy installation and configuration | wished | pip install, config file | 1 |  |
| Collaborative development | to improve | GIT, pull requests, github discussion, workshops | 1 |  |
| Storage ring agnostic applications | keptto improve | transfer lines, boosters | 1 |  |
| Linac tuning applications | to add |  | 2 |  |
| Graphical interfaces | re-write |  | 3 |  |
| Digital twin | to improve |  | 1 |  |
| Digital shadow (online digital twin) | wishedto add |  | 2 |  |
| LOCO | to improvere-write | analytic formulas [9] | 4 |  |
| BBA | to improve |  | 4 |  |
| Orbit, tune, chroma | keptre-write |  | 1 |  |
| AC-LOCO, AC-BBA | wished |  | 5 |  |
| Turn by turn optics | wished | see OMC3 for example | 4 |  |
| GPU tracking | wished | work in progress in pyAT | separate |  |
| First turns | to improve |  | 4 |  |
| Commissioning sequence | wished | pySimulatedCommissioning https://github.com/lmalina/pySC | 5 |  |
| Automated commissioning | wished |  | 5 |  |
| Matlab maintenance for benchmark | kept |  | 2 |  |
| … ADD OTHER FEATURES during meeting … | to add |  |  |  |

### Supporting and strengthening the community, with the long-term aim of facilitating migration from MML to pyML

* Promoting best practices keeping in mind that there are diverse use cases
  + Use AT alone
  + Use AT/ML alone
  + Development for AT/ML
  + Operations
* Animating the community (biannual pyML/pyAT workshop, GitHub training, a soft steering committee)
* Building trust and ease of use
* Training and skills maintenance for rich collaborative development
* Software, Github, Continuous Integration / Continuous Development (CI/CD)
* Working with virtual environments
* Facilitating local and remote developments on various GitHub/GitLab servers

### Reinforce robustness and testing for each release

* Intrinsic fragility of fast-changing Python environments
* CI/CD aka Continuous Integration/Continuous Developments:
  + Unit testing,
  + Regression testing

### Enrichments compared to MML

* Data management
  + Focus on OPENDATA formats
  + The FAIR Principles (Findable, Accessible, Interoperable, Reusable) correspond to guidelines whose primary aim is to improve the reuse of research data. They were published in 2016 in the article [The FAIR Guiding Principles for Scientific Data Management and Stewardship](https://www.nature.com/articles/sdata201618).
* High-Performance Parallel Computing (CPU/GPU)
* Machine Learning (for example, anomaly detection based on the continuous comparison among online-digital twin and measurements)
* Enabling connection to online digital twin (digital shadow) for TANGO/EPICS
* Improve and enforce Documentation (every new development is accepted only if documented). Creation and update of a manual.
* Setting and promoting a forum starting with the tool in place (GitHub)
* Work simultaneously on several accelerators at the same time (injector, linacs, booster, transfer lines, storage ring)
* Fast measurement algorithms based on Turn-by-turn data of Fast acquisition flow of the BPMs

# References

[1] https://github.com/atcollab/MML

[2] J. Safranek et al. Linear Optics from Closed Orbits (LOCO): An introduction, SLAC-REPRINT-2009-545

[3] https://github.com/atcollab/at

[4] https://github.com/atcollab/at/tree/master/pyat

[5] https://github.com/DiamondLightSource/pytac

[6]<https://github.com/dls-controls/atip>

[7] https://github.com/ThorstenHellert/SC

[8] https://github.com/lmalina/pySC

[9] A. Franchi et al. “Analytic formulas for the rapid evaluation of the orbit response matrix and chromatic functions from lattice parameters in circular accelerators” https://arxiv.org/abs/1711.06589

### GLOSSARY:

Middle Layer: software layer that allows connecting to any control system or digital twin. It works with any pyAT accelerator lattice model following a series of guidelines.

Digital Twin: simulated control system for off-line development of operation tuning applications

Digital Shadow: continuous simulation of accelerator properties based on magnets set point and measurements.

Commissioning Simulations: series of simulated tuning and corrections from first turns to fully corrected machine. Done for many seeds.