Brainstorming Document for Middle Layer Collaborative Development

Authors (random order):
S.M.Liuzzo (ESRF), W. Sulimankhail (HZB), J. DAHENG (IHEPS), J. Biernat (Solaris), P. Goslawski (HZB), M.Gaughram (Diamond), G.Benedetti (Alba), Z. Marti' (Alba), N.Carmignani (ESRF), Chong Shik Park (K4GSR), M.Ries (HZB), X.Yang (BNL), L.Malina (DESY), T.Hellert (LBNL), J.Kellestrup (PSI), A. Khan (BNL), G.Bassi (BNL), Y. Hidaka (BNL), I. Agapov (DESY), S.Krecic (Elettra), X.Huang (SLAC), L.Farvacque (ESRF), P.Schinzer (HZB), R.Fielder (Diamond), L.Nadolski (Soleil), T. Olsson (HZB), M.Bree (CLS), S.White (ESRF), S.Mengyu (IHEPS), M.Apollonio (MAXIV), Yong-Chul Chae (DESY), Magnus Sjostrom (MAXIV), Vyacheslav Kubytskyi (IJClab), Hung-Chun Chao (Diamond), Tobyn Nicolls (Diamond), Paul Bennetto (Australian Synchrotron), T.Charles (Australian Synchrotron), Samira Kasaei (Sesame), Song, Minghao (BNL), R.Tomas (CERN), B. Nash (Radiasoft), J.Edelen (Radiasoft), Edmund Blomley (KIT)

Other institutes to contact:

EuXFEL, Brazilian light source, …

Represented Labs (random order):

ESRF, HZB, Diamond, Alba, K4GSR, NSLS-II, DESY, LBNL, PSI, Sesame, Elettra, SLAC, Soleil, IHEPS, CLS, MAX-IV, IJClab, Solaris, CERN, KIT, Radiasoft, Ansto

Represented companies (1, random order):

Radiasoft

History:

- 2 November 2023, LSN, Document Creation
- 10 November 2023, LSN, SL
- 23 November 2023, SL, SW, LSN
- 5 December 2023, document reviewed during meeting
- 11 December 2023, SL includes in text comments raised during the meeting and updated tables/figures. First draft of strategy timeline. SW, add python strong points.
- 18 December 2023, SL includes comments from M.Apollonio. SL further details LEAPS meeting outcome and WP list/description. SL adds a draft budget.
- 12 February 2024, general meeting to finalize this document. Name still not selected, reduced number of candidates. Added a few names to each subject, updated budget/resources table.

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" or “simulator/online”). 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. MML is nevertheless a most valuable tool that will continue to be used for many years and by several labs. MML will serve as a reference for the present project proposal, thus including efforts for its maintenance in the mid-term.

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.
Available python packages allow integrating modern computing techniques such as heavy parallelization (CPU/GPU) or to interface for modern scientific libraries involving advanced data analysis, optimization algorithms or machine learning algorithms that are now a critical aspect of accelerator design and operation.
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).
The packages *aphla* (NSLS-II), *py4syn* (Brazilian Light Source), SLS-PSI python-epics virtual accelerator are other examples of existing “python Middle Layers”.

---

[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

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 (TEMPORARY NAME)** 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 (TEMPORARY NAME)** would profit from all these recent developments and have all the features of the old MML software:

1. Control system agnostic
2. Accelerator agnostic
3. Works for transfer-lines, linacs, ramped accelerators
4. Digital twin: allows to test tuning tools in real life conditions without need of expensive and rare beam time

It would also, thanks to existing python packages and features:

1. Fully open source code (with licence to be defined)
2. include the possibility to be used on large computing clusters for automated commissioning simulations,
3. 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,
4. enable seamless connection to our facilities' control system (EPICS, TANGO, DOOCS, and so on),
5. make use of most modern OPENDATA and OPENSCIENCE and FAIR (Findable, Accessible, Interoperable, Reusable) principles, data labeled for Machine learning and Artificial intelligence algorithms
6. include off-line digital twin for tuning tools development and on-line digital shadow for monitoring of indirect observables.
7. enable straightforward implementation of available python based Artificial Intelligence and Machine Learning tools (e.g.: pytorch, Badger/Xopt)
8. Enforce thorough documentation and easier/shared maintenance (github)

The **Python Accelerator Digital Twin (TEMPORARY NAME)** 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.

[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

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 yet have 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. (DESY candidates, June, if no other volunteer, Berkley second candidate)

Machine Middle Layer Working group

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: https://forms.gle/7PmV2nGzRpjtewwE7

Timeline strategy

Immediate term:

WP leaders will become scientific board of the project.

Could define also a “project management board” (max 5 people, frequently changing) to take decisions if needed in case of absence of unanimous agreement among labs.

Short term: setting the base for development

- Create from the existing codes a control agnostic python middle layer
    - Gather features of existing tools (developed within the accelerator community or elsewhere). Either choose an existing software as a starting point or define detailed specification of a new software.
    - Define architecture of software: modular, easy to maintain, install & configure
    - Verify that all tools will be control and accelerator agnostic as for MML.
    - Define it all tools will also be simulation back-end agnostics (pyAT, Elegant, MAD, SAD, Xsuite)
    - Define if units will be enforced (SI for example, or software enforcement) [10]
    - Define if there is need for dynamic lattice element additions
    - Define best tools to provide GUI to pyML tuning tools.
    - Define list of desired features, including those already present in MML. Keep in mind the need for Turn-by-turn features and FAIR data principles.
    - Agree on lattice formats naming structure to be used
    - Agree on documentation / unit testing rules for code updates
    - Test in at least 3 facility with different control systems
    - Define licencing terms for all software packages
- Define if calibration package is needed
- Create from the existing codes a control agnostic digital twin for development of higher level tuning tools
    - Gather features of existing tools. Either choose a starting point or define specification of new tool.
    - Include fault cases and realistic conditions in digital-twin (vibration of magnets, BPM incoherency/offsets, lattice errors, beam loss, etc..)
    - Include the possibility to perform user defined lattice updates while the digital twin is running (simulate simultaneous action of other applications than the one being tested).
- Set up first/basic tuning tools in the digital-twin mode: such as orbit, tune and chromaticity based on existing tools in MML. The underlying tuning tools should be usable:
    - on the real accelerator and on a simulated model
    - either used from command line (for commissioning simulations) or from a GUI (if available)
    - Should work for linac/booster if possible / applicable
- Test tools in operation-mode in at least 3 facilities with different control systems
- Create first user interfaces and test them in digital-twin mode
- Test user interfaces operation-mode in at least 3 facilities with different control systems
- Verify all documentation is present.
- Verify unit testing is present for all code
- Release 0.0.0

10 https://github.com/hgrecco/pint

Mid term: recover all MML tools, add digital-shadow, use for virtual commissioning

- Set up a process of 1 release every 6 months (June/December).
- Set up 2 x 2-days workshops per year (1 full remote, 1 hybrid) for updates and progress with training sessions.
- Develop and test in digital-twin mode:
    - LOCO,
    - BBA,
    - Turn by turn optics tuning,
    - First turns beam threading
    - NOECO
    - DA
    - AC-BBA
    - AC-LOCO
    - etc…
- Test above tools in operation-mode in at least 3 facilities with different control systems
- Test above tools for linac and ramped accelerators (boosters) if applicable
- Set up tools for digital-shadow, test them in at least 3 different laboratories
- Apply tuning tools to run virtual commissioning in a large computing cluster
- Use available artificial intelligence for optimization (ex: pytorch, Badger/Xopt)
- Test use of developed tools for simultaneous tuning of injectors (linac, booster and main storage ring tuning).
- Show test cases of specific tools designed for specific lab needs (ex different injection tuning tools in different labs due to different injection schemes)
- Verify all documentation is present.
- Verify unit testing is present for all code
- Release 1.0.0. This release should feature the equivalent of most basic MML tools.
- Maintain Matlab Middle Layer (MML) operational and evolve with new functionalities:
    - Make MML compatible with most recent matlab visions (matlab 2023)
    - Bunch by Bunch middlelayer library - functions to make it easier for physicists to visualize and turn by turn process data (from any system)[11].
    - Turn by Turn (Spark) middlelayer library – different sets of code exist for SPEAR that could be used as starting examples.
    - PlotWaveform – similar to PlotFamily but used for waveforms. We have a machine-independent version for SPEAR where the input waveform families are file driven.
    - PlotImage – define the data structure, provide some image/zoom, enhancements, get/save, etc
    - Transport line emittance interface. Work exists for SPEAR. LCLS should have versions and algorithms
    - Transport line LOCO – work exists for SPEAR
    - Transport line steering interface – work exists for SPEAR

[11] https://accelconf.web.cern.ch/ipac2018/papers/wepaf056.pdf#search=%20domain%3Daccelconf%2Eweb%2Ecern%2Ech%20%20%2Bauthor%3A%22corbett%22%20%20FileExtension%3Dpdf%20%2Durl%3Aabstract%20%2Durl%3Aaccelconf%2Fjacow

- Applications for swap-out injection (?)
- Beamline steering – allows operators to steer the photon beam In x-x', y-y'. Work exists for SPEAR
- Corrector 'heatmap' – show if a corrector is having problems.
- Archiver for configurations, etc
- Conversion from GUIDE to hard code or AppDesigner
- New BBA routines using parallel BBA
- New optics correction routines for LOCOM.
- Perhaps it is also a good idea to have a universal tool for retrieving, viewing, and analyzing history data.
- A tool for viewing fast orbit data (for example, 4 kHz for SPEAR).
- An interface for viewing and analyzing Beam Loss Monitor data.
- LabCA maintenance

Long term: test new features, keep and improve high quality standards,

- Set up 1 workshop every 2 years (hybrid) for updates and maintenance.
- Install and test in all partner laboratories and compare to MML
- Develop in digital-twin-mode automated commissioning simulations sequences
- Test above automated tuning for at least 3 different facilities
- Include AI tuning in list of automated commissioning
- Test use of tuning tools in continuous mode (slow feedback)
- Feedback optics measurements to digital-shadow mode
- Empower digital-shadow with heavy computations such as Lifetime, injection efficiency, dynamic aperture, losses, and collective effects continuous simulations
- Add anomalies for more realistic virtual accelerator and more robust app development
- Consider possible inclusion of FEL. Ex PERSO (Mathcad) code by Luca Giannessi Elettra.
- Test automated commissioning simulations in at least 3 laboratories
- Verify all documentation is present.
- Verify unit testing is present for all code
- Release N.0.0. This release should be ready for most basic MML tools.

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…)

Project Architecture (software)

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 additional packages in the global project such that the maintenance of each element is well separated from that of the others (software quality assurance): ex pyAT, pySC, pyNAFF (GNU GPL v3), pyLOCO, etc…

Possible scheme in figure 1 (all code is python or C if needed for speed):

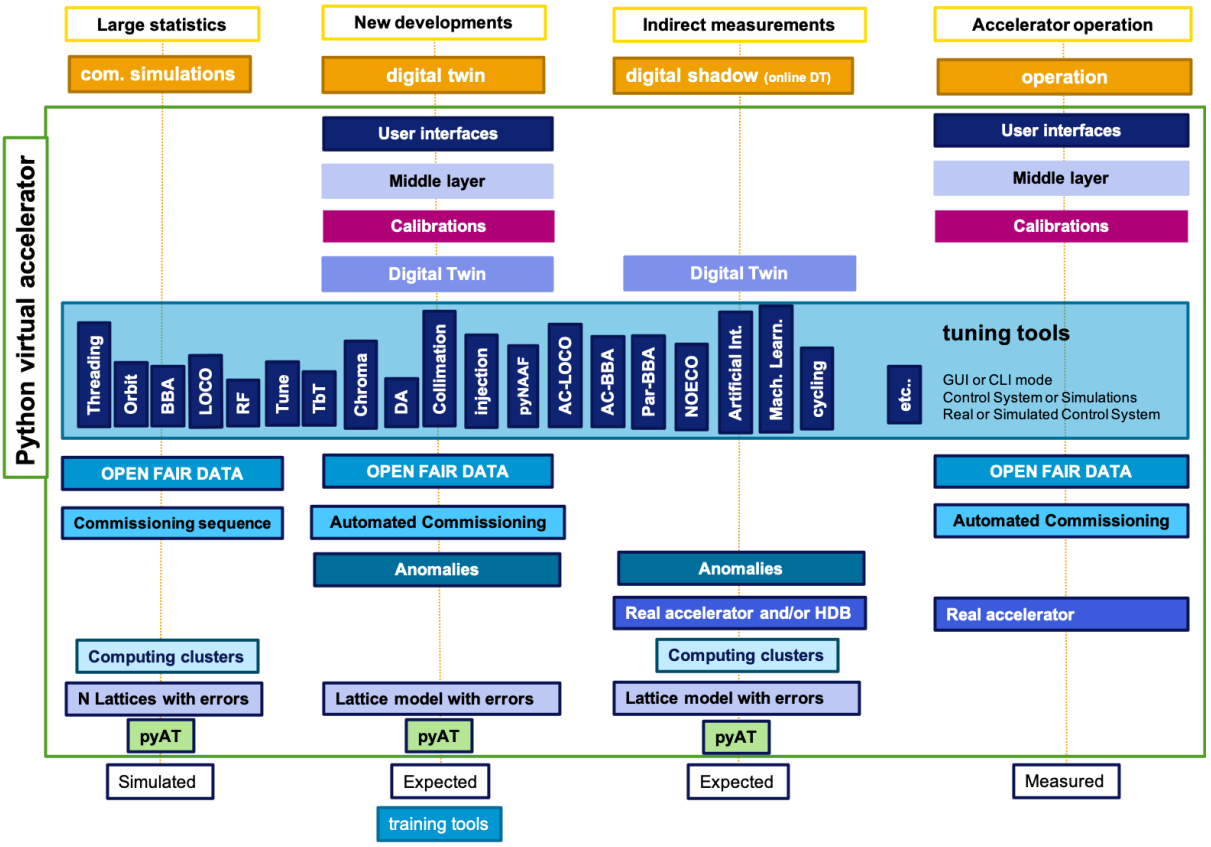

Figure 1: possible scheme of (TEMPORARY NAME) project. 4 columns represent the 4 uses of the code. Each block represents a stand alone software package or group of software packages. Identical blocks indicate that the software is developed such that it works in all cases unchanged. The green block encompasses the software tools.

List of software packages to be developed:

- **Middle Layer (pyML))**. Main objective: switch among different control systems and different accelerators. (if needed among different simulation backends)
- **Magnet Calibrations (pyMC).** Sets a framework to assign calibrations Current-strengths including combined function magnets. Could make use of Radia models or completely by-passed by facilities that perform calibrations at the control system level. May also include handling of hysteresis when/if possible.

- **Digital Twin (pyDT)**: replaces the electron beam with particle accelerator simulations. It would take magnet/RF/etc.. set points data from a pyAT model (off-line, digital twin) or from the real magnets (on-line, digital-shadow)
- **Tuning Tools (pyTT)**: Several packages, one for each tuning tool. If some features can be grouped a layer of code will be added. Main objective:
    - measure and correct specific accelerator quantities. Ex: orbit, tune, chromaticity, optics, DA, lifetime, Artificial Intelligence / Machine learning based tuning, Turn by turn optics...
    - set operation point for accelerator components. Ex: collimators, non-linear magnets, fast kickers/pulsed elements, etc…
    - Acquire, store and analyze data following FAIR principles
    - All tools work from command line (GUI is not compulsory and is a separated code, without any active computation)
    - All tools work on the real control system or in pure pyAT simulations (act via Middle Layer)
    - All tools work based on a real or simulated diagnostics (act via Middle Layer)
- **Graphic User Interfaces (GUIs)**. Several, one for each tuning tool (~ 20). Main objective: control room operation.
- **Commissioning sequence (pyCS)** / automated commissioning: run in a sequence several tuning tools. Either acting on real accelerators or on individual simulated error seeds. Use pySC.
- **Anomalies (pyAN)** trigger anomalies in the digital twin: random errors, beam losses, lifetime issues, etc…

This will result in more than 50 different independent software packages. Each package could be forked at a given lab for specific development. Each package will run unit-tests at each commit and will be released only if all new features are fully documented (Sphynx). Each package will make use as often as possible of software developed elsewhere to simplify long term maintenance.
The architecture of the software will be tailored to the developers skills that are mostly accelerator physicists. Some simplifications to the architecture will be accepted in view of the long term maintenance. Else more expert software engineers will have to be involved (hired for the project).
At the global level (green box in Figure 1) all accelerator specific configurations are set and the installation takes care of installing the relevant packages. This configuration may be extended at will to cover local needs (generic/mandatory and specific configuration files).

DIAMOND development is a good starting point for pyML and pyDT, evolution from NSLS-II *aphla*. Python Toolkit for Accelerator Control (*pyTAC*) and pyAT Interface for pyTAC (*atip*) LBNL/DESY pySC could be a good start for modernized tuning tools.
The BlueSky[12] code (??, also used by X-ray beam lines) could be used and integrated for data analysis and acquisition.

12 blueskyproject.io

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).*
- Increase the use of operations that require synchronization, e.g. orbit feed-back processes, and in general evolution in time (e.g. tools to have a continuously updated ORM). At present MML is used in a rather static way, our wish at MAXIV was that pyML can become more “active”. In Tango we could exploit the TANGO/EPICS “event” feature to have a flow of data continuously updated, instead of using the time/network consuming polling feature. Maybe my comments can be included in the concept of digital shadow.

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 (short term development). 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	Y. Hidaka
Easy installation and configuration	wished	pip install, config file	1	
Collaborative development	to impr…	GIT, pull requests, github discussion, workshops	1	
Digital twin	to impr…	E.g.: atip	1	Y. Hidaka, S.Liuzzo
Calibrations	kept to impr…	Combined function magnets, hysteresis, cycling	1	S.Liuzzo
Storage ring agnostic applications	kept to impr…	transfer lines, boosters	2	
Linac agnostic tuning applications	to add		2	

FEATURE	Status	Description	priority	Contributors
Digital shadow (online digital twin)	wished to add		2	S.Liuzzo
Graphical interfaces	re-write		3	
GPU tracking	wished	within pyAT	separate	
Commissioning sequence	wished	e.g.: pySC	5	Y. Hidaka, S.Liuzzo,Jacek Biernat
Automated commissioning	wished		5	
Matlab maintenance for benchmark	kept		2	
OPEN FAIR data	to add		1	
high-performing feedbacks	to add	Could use event	5	
Connection to History data base	to add	Update DT based on HDB data	5	
OTHER FEATURES	to add			

Table of tuning tools (priority 1 is reserved for pre-requisites: middle layer, calibrations, digital twin,):

Tuning tools	Status	Description	priority	GUI Priority (+1)	Contributors
Orbit	kept re-w…		2	YES	S.Liuzzo
Tune	kept re-w…		2	YES[1]	S.Liuzzo
Chromaticity	kept re-w…		2	YES	S.Liuzzo
Optics plot navigator	kept re-w…		3	YES	
COD bumps	wished		3	YES	
BBA	to im…		3	YES[2]	Minghao Song

LOCO	to im… re-w…	analytic formulas	3	YES[3]	S.Liuzzo
LOCOM	wished		4	YES[4]	
Transfer-line LOCO	wished		4	YES	
NOECO	wished		4	YES[5]	
Turn by turn optics + NAFF	wished	see OMC3 for example	4		
ICA	wished		4		X.Huang
First turns	wished to im…		4		S.Liuzzo
Turn-by-turn tune	wished		4	YES[1]	
AC-BBA	wished		5	YES[2]	
AC-LOCO	wished		5	YES[3]	
AC-LOCOM	wished		5	YES[4]	
AC-NOECO	wished		5	YES[5]	
Artificial intelligence optimizations	wished	RCDS, Badger, Xopt, ML, AI	5		Xi Yang, S.Liuzzo
Machine learning optimizations	wished		5		Minghao Song
Anomaly detection	wished		5		Jacek Biernat, S.Liuzzo

For each tuning tool a graphical interface is expected. It is not mandatory for all applications as in some cases the tool is used very seldom or only by experts. Some interfaces could be shared among several tuning tools, for example BBA and AC-BBA. The development of tuning tools and of their corresponding interfaces is separated. The graphical interface will be linked to a given release of the underlying tuning tools.

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 OPEN DATA 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.
- 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

LEAPS remarks

LEAPS support to apply for (MARCH 2024!):
https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/horizon-infra-2024-tech-01-04

This call has been analyzed by LEAPS and ESRF EU grant experts. It is too big, not fitting the whole description (only digital twin, but no DestinationEarth and support to policy decision) and the deadline is to close.

For next project:
LEAPS will deal with the part A of the project proposal (about 150 pages of lab info, names and signatures)
We will write 45 pages of B part. template or from DITARI available. Likely more support from ESRF EU-grant office.

For workshops: LEAPS will support workshops.

Work packages (proposal)

WP 1: coordination

Description: organize meetings and workshops. Collect material for milestones and deliverables. Interact with LEAPS/EU. Finance external collaborators trips, conference fees, consultancies. Student grants.

WP 2: training, communication, dissemination, capitalization

WP 2.1: organize internal training for git, pyAT, hdb, python, etc…

Description: training for needed tools for virtual accelerator development: git, python

WP 2.2: use python virtual accelerator as a training tool

Description: for newcomers and control room operators. The training tools are powered by the Virtual accelerator.
Based on developments of the project, training tools are open to Accelerators schools, PhD-schools, operators, etc...

WP 3: python middle layer

WP 3.1: python middle layer

control system agnostic, accelerator (SR, linac, transferline,) agnostic middle layer. Calibrations. Collaborative development.

WP 3.2: tests python middle layer developments

Continuous development tests for git repository to guarantee safe/backward compatible releases.
Recurrent and easy to run tests of the python middle layer at several accelerator facilities.

WP 4: digital twin and shadow

WP 4.1: Digital twin, digital shadow

Description: Digital twin and digital shadow cores. Piloted by pyML commands (WP1). allows full tuning of digital model status: radiation, turn-by-turn, quantum effects, tapering, vacuum levels, filling patterns, etc…

WP 4.3: intensive computations quantities within the digital twin and shadow

Description: Touschek lifetime, vac lifetime, inj eff, losses (levels (relative, estimated from number of lost particles, location of deposited particles) data updated in digital shadow and twin, gpu powered (WP4)

WP 5: tuning tools

Description: define tuning tools and their corresponding graphical applications based on work of WP1. Test using digital twin (WP2) and real accelerator. Use Open FAIR Data. Computing cluster interface.

WP 5.1: command line tuning tools

WP 5.2: graphical user interfaces

WP 6: pyAT development

Description: development of pyAT and GPU tracking in pyAT, hardware for computations (GPUs, CPUs)

WP 6.1: pyAT developments

WP 6.2: GPU tracking development

WP 6.3: pyAT for linacs

WP 6.4: pyAT for ramped accelerators

WP 7: Matlab middle layer maintenance

Description: maintenance of Matlab middle layer code for potential later transition

WP 8: commissioning

WP 8.1: virtual commissioning

High statistics simulations for tolerance definitions
Several seeds of specific magnet and diagnostics errors are applied in the lattice. The commissioning sequence is applied to each seed and relevant quantities are computed and visualized.

WP 8.2: automated commissioning

Virtual commissioning sequences are applied to real storage rings, transfer lines and linac and the corresponding pyTT(WP3) tools are further tuned.

Description: exploiting tuning tools develop

WP 9: Artificial intelligence for anomaly detection and optimizations

Description: development of external tools that simulate anomalies and failures scenarios within the DT. For example: beam loss, vacuum leak, lifetime accident, failure of correctors, loss of synchronization, blocked scrapers, obstacles, etc... Build a scalable solution (easy addition of new failures/anomaly scenarios) that may be enabled with a given rate of events in the Virtual accelerator pyDT (WP4). This will allow more realistic virtual accelerators and robust applications developments (WP5) and more educative training tools (WP2).

Compile list of possible anomalies/failure scenarios: vac leaks, hole in chamber, melted rf finger, id liner folded, (list from operation of all partners

WP 9.1 Machine learning for optimizations

WP 9.2 Machine learning for anomaly detection

Resources & Budget (summary)

Assuming:
4 years project duration
~ 10 k€ for one person/month

	Available RESOURCES								total pm		83			red if too many pm!							
	person * month (pm)								MAX pm		370			project budget =		5000	kEuro				
	EU funds eligible											Associates		non-EU							
Work packages	**ESRF**	**CERN**	**Diamon**	**DESY**	**SOLEIL**	**ALBA**	**MAXIV**	**HZB**	**IJClab**	**Solaris**	**ELETTR**	**PSI**	**Sesame**	**NSLS-II**	**K4GSR**	**SLAC**	**CLS**	**LBNL**	**IHEPS**	**RADIASOFT**	**Needed resources (pm)**
WP 1 coordination	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
WP 2 trining/edu	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
WP 3 pyML	1	0	0	0	1	0	0	12	0	1	0	0	0	1	0	0	0	0	0	0	15
WP 4 pyDT	3	0	0	0	0	0	0	12	0	0	0	0	0	1	0	0	0	0	0	0	15
WP 5 pyTT	6	0	0	0	0	0	0	4	0	0	0	0	0	7	0	0	0	0	0	0	10
WP 6 pyAT	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
WP 7 MML	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WP 8 commiss.	6	0	0	0	0	0	0	8	0	2	0	0	0	1	0	0	0	0	0	0	16
WP 9 anomaly	1	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	8
total pm	**36**	**0**	**0**	**0**	**1**	**0**	**0**	**36**	**0**	**10**	**0**	**0**	**0**	**10**	**0**	**0**	**0**	**0**	**0**	**0**	**83**
total pm / 48 mont	**0,75**	**0**	**0**	**0**	**0,0208**	**0**	**0**	**0,75**	**0**	**0,2083**	**0**	**0**	**0**	**0,2083**	**0**	**0**	**0**	**0**	**0**	**0**	**1,7292**

	In kind contributin RESOURCES																			
	person /month																			
Work package	ESRF	CERN	Diamond	DESY	SOLEIL	ALBA	MAXIV	PSI	HZB	IJClab	Solaris	ELETTRA	NSLS-II	K4GSR	SLAC	CLS	LBNL	Sesame	IHEPS	RADIASC	in kind contribu
WP 1 pyML	k€50								k€0												k€50
WP 2 pyDT	k€10								k€0												k€10
WP 3 pyTT	k€30								k€0												k€30
WP 4 pyAT	k€20								k€0												k€20
WP 5 MML									k€0												k€0
WP 7 logistics									k€0												k€0
WP 7 training									k€0												k€0
WP 8 com sim	k€20								k€0												k€20
WP 9 anomaly	k€20								k€0												k€20
TOTAL kE	k€150																				
detail:																					
	beamtime								beam time at two rings with multiple optics each												
	computing cluster								computing cluser (CPU)												
	GPU																				

The above tables describe the available resources and potential in kind contributions (beam time, computing clusters, etc…) . The total (p/m + in kind) commitments are usually doubled when asking for EU funds, depending on the call.

External labs (non-EU funds eligible) could be financed for trips, conference fees and consultancies.

Other references

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.

Virtual accelerator: simulated control system for off-line development of operation tuning applications

Digital Twin: returns beam parameters including optics, injection efficiency, lifetime (vacuum and touschek) , collective effects, etc… based on a pyAT digital model. Anomalies act on the digital twin to drive abnormal states for the machine.

Digital Shadow: as the digital twin but running based on strength continuously updated from the real accelerator. Updates properties based on magnets set point and measurements on the accelerator. Comparing the measured values to the digital shadow enables to detect beam anomalies.

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