AIMSYS: AI for model-based diagnostic at System level

Final Presentation 2024

Maciej Prokopczyk	GMV PL
Maciej Szreter	GMV PL
Anna Banachowicz	GMV PL
David del Val	GMV ES
Leandro Garcia	GMV ES
Yusra Al-Khazraji	GMV DE
Inmaculada Perea	GMV ES
Vanessa Navarro	GMV ES
Sepideh Rahimian	GMV DE
Daniele Segneri	ESA/ESOC
Federico Antonello	ESA/ESOC

INNOVATING SOLUTIONS

Index

TL;DR

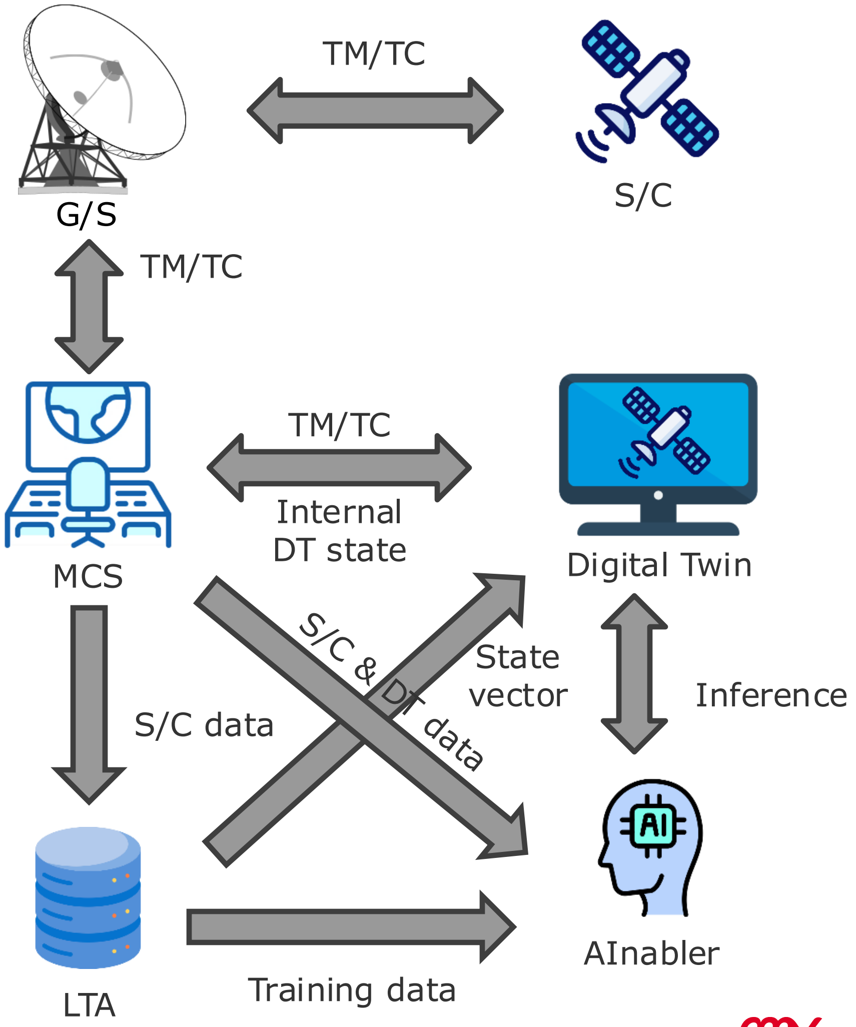
Overview

Goals:

- **Creation of DT infrastructure based on ESA/ESOC ground SW**
- **Definition of AI/ML logic supporting selected use cases**
- **Validation with a real mission**

Achievements:

- **Analysis of existing SW**
- **Learning end-users needs**
- **Design of generic DT infrastructure**
- **Implementation of a prototype with core functionality**
- **Use of flying mission for validation**



Demo

Presented aspects:

- Creation of new DT instance from GUI
- Creation of the DT dedicated session in MCS (for data separation)
- Setting the DT state to a be as close as possible as its S/C counterpart at given time
- Establishing links between MCS and the DT instance
- Receiving DT TM data by MCS and their visualization
- Checking that TM data include parameters generated with use of the AI model incorporated into the simulator
- Commanding DT by invoking a predefined activity list
- Receiving TM parameters describing the DT internal state

Needs

Introduction

Objectives

Exploit the usage of Model Based diagnostics at System level for reliable anomaly detection as well as for pre-launch testing including pass/fail criteria evaluation for tests:

- ❑ Define, prototype and assemble the infrastructure platform running the AI
- ❑ Define AI/ML logic supporting anomaly detection, root cause analysis, ground testing, resource usage predictions & margins refinement
- ❑ Validate the implemented prototype against a real mission

Concepts

Spacecraft Digital Twin (S/C DT): Dynamic and self-evolving digital representation of the exact S/C state at given time

- ❑ Improve understanding of Spacecraft behavior
- ❑ Support and improve efficiency of FCT tasks
- ❑ Forecast future spacecraft state
- ❑ Detect anomalies
- ❑ Many more …

Alignment: setting the state of the simulator so that it matches a provided state vector

- ❑ Synchronization - adjusting the simulator state so its TM matches the values of the real asset.
- ❑ Calibration - adjusting simulator analogue coefficients inside the simulator models

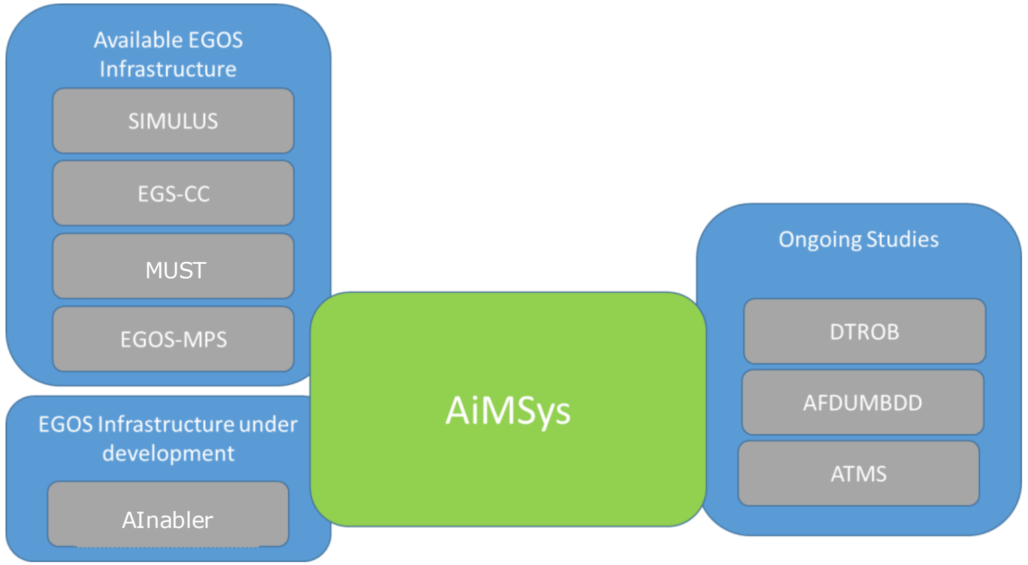
State Vector (SV): digital representation of the S/C state at a given point in time

- ❑ Synchronization State Vector - telemetry parameters that characterize the operational state of the spacecraft
- ❑ Calibration State Vector – values of simulator analogue coefficients to use in simulator SMP models

Context

Available infrastructure

- ❑ Use the SIMULUS infrastructure for system modelling
- ❑ Interact with EGS-CC for monitoring and control
- ❑ Interact with MUST for housekeeping storage and retrieval
- ❑ Interact with the EGOS-MPS for mission planning
- ❑ Rely on the AInabler infrastructure for AI/ML development and training
- ❑ Reuse, consolidate and focus the outcomes of the ongoing studies
  - ❑ pre-launch domain studies' outcomes did not match project needs
- ❑ Pre-launch functionality was descoped



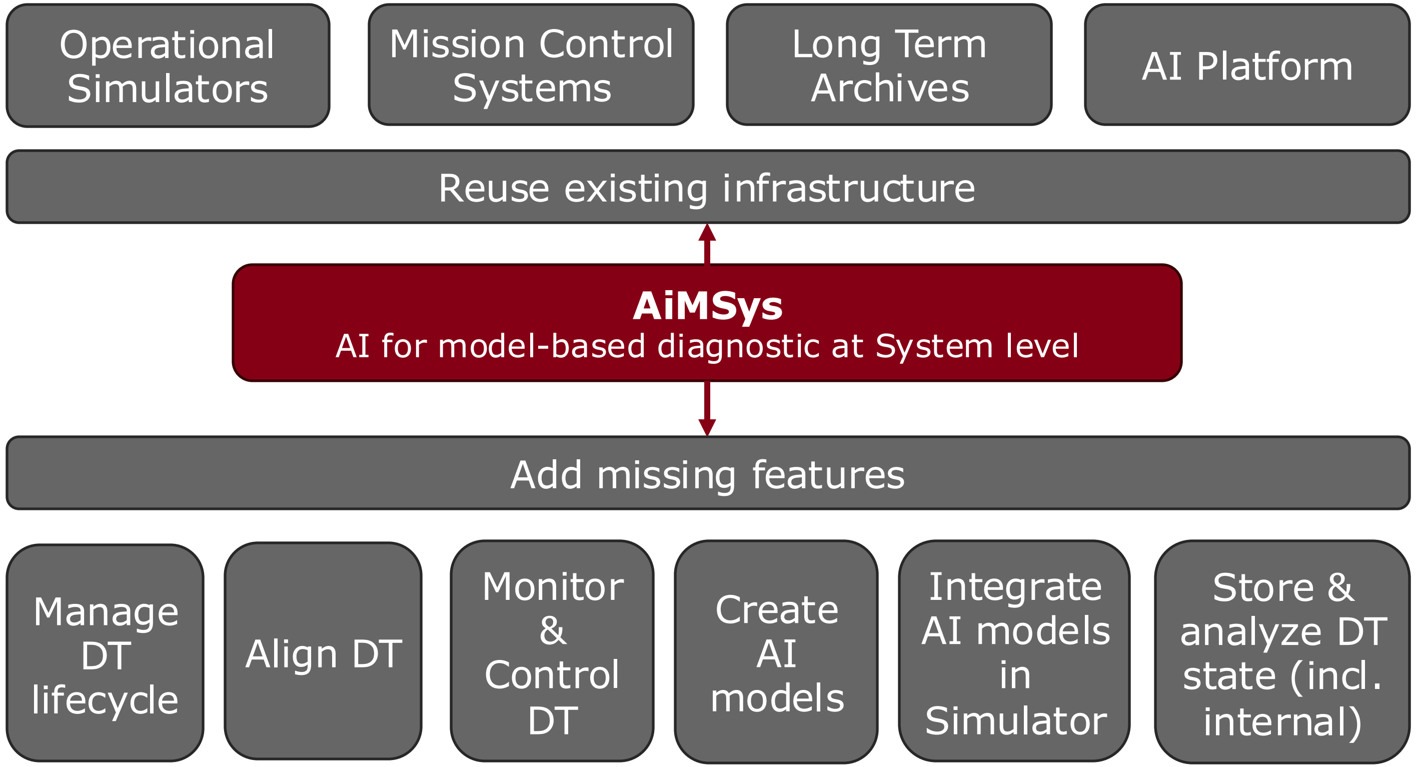
Gaps in current infrastructure

Missing mechanisms:

- Simulator with DT capabilities (i.e. automatic alignment)
- DT lifecycle management
- Quick evaluation of past or custom scenarios
- Storage and analysis of simulator-generated data
- Improvement of simulator fidelity by AI
- Analysis of internal DT state

Solution

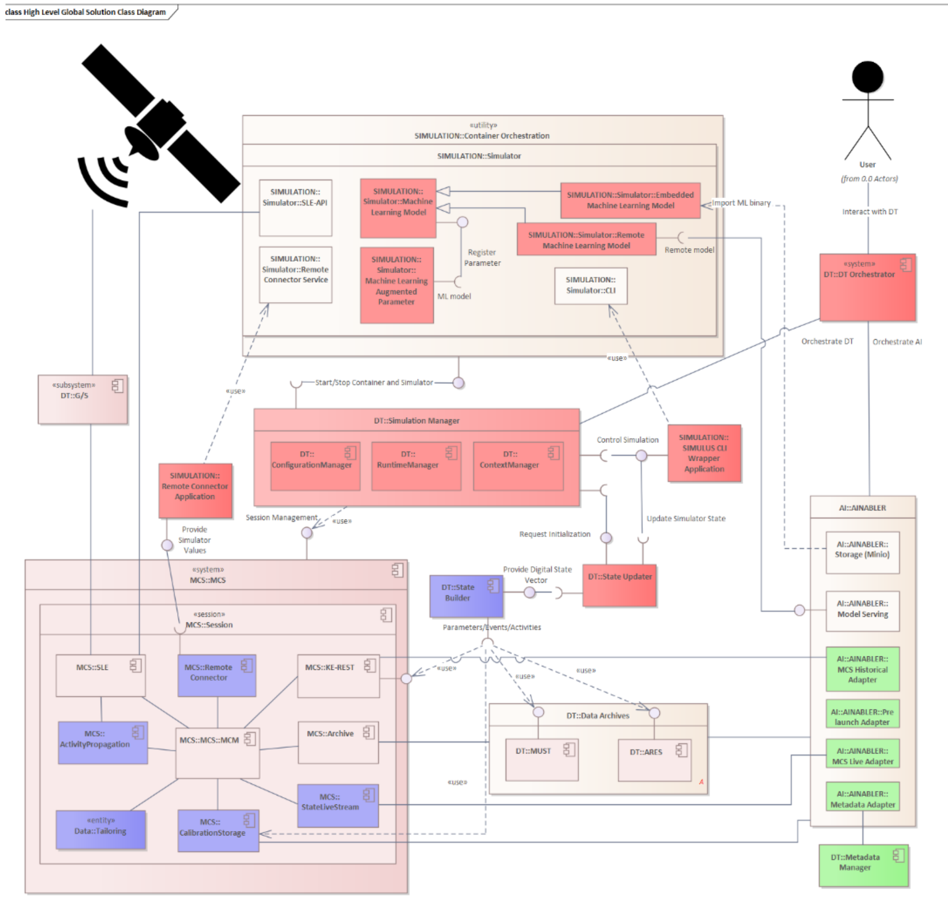
Proposed approach



Architecture

DT infrastructure

- DT Orchestrator – for handling high-level tasks
- Simulation Manager – for controlling DT
- Simulation Container Orchestration – for running DTs in containers
- State Updater – for DT alignment
- State Builder – for building SV needed by SU
- MCS – for Monitoring & Control of DT
- Remote Connectors – for storing internal DT state
- AI platform – for AI/ML creation
- AI application – for analysis of DT data
- ML Model – for incorporation of AI model in SIM



class High Level Global Solution Class Diagram
«utility» SIMULATION::Container Orchestration
SIMULATION::Simulator
SIMULATION:: Simulator::SLE-API
SIMULATION:: Simulator::Machine Learning Model
SIMULATION::Simulator::Embedded Machine Learning Model
Import ML binary
SIMULATION::Simulator::Remote Machine Learning Model
Remote model
Register Parameter
SIMULATION:: Simulator::Remote Connector Service
SIMULATION:: Simulator:: Machine Learning Augmented Parameter
ML model
SIMULATION:: Simulator::CLI
User
(from 0.0 Actors)
Interact with DT
«system» DT::DT Orchestrator
Orchestrate DT
Orchestrate AI
«use»
Start/Stop Container and Simulator
«subsystem» DT::G/S
DT::Simulation Manager
DT:: ConfigurationManager
DT:: RuntimeManager
DT:: ContextManager
Control Simulation
SIMULATION:: SIMULUS CLI Wrapper Application
SIMULATION:: Remote Connector Application
Session Management
Provide Simulator Values
Request Initialization
Update Simulator State
AI::AINABLER
AI::AINABLER:: Storage (Minio)
AI::AINABLER:: Model Serving
AI::AINABLER:: MCS Historical Adapter
AI::AINABLER::Pre launch Adapter
AI::AINABLER:: MCS Live Adapter
AI::AINABLER:: Metadata Adapter
DT::Metadata Manager
«system» MCS::MCS
«session» MCS::Session
DT::State Builder
Provide Digital State Vector
DT::State Updater
Parameters/Events/Activities
MCS::SLE
MCS::Remote Connector
MCS::KE-REST
MCS:: ActivityPropagation
MCS::MCS::MCM
MCS::Archive
MCS:: StateLiveStream
«entity» Data::Tailoring
MCS:: CalibrationStorage
DT::Data Archives
DT::MUST
DT::ARES

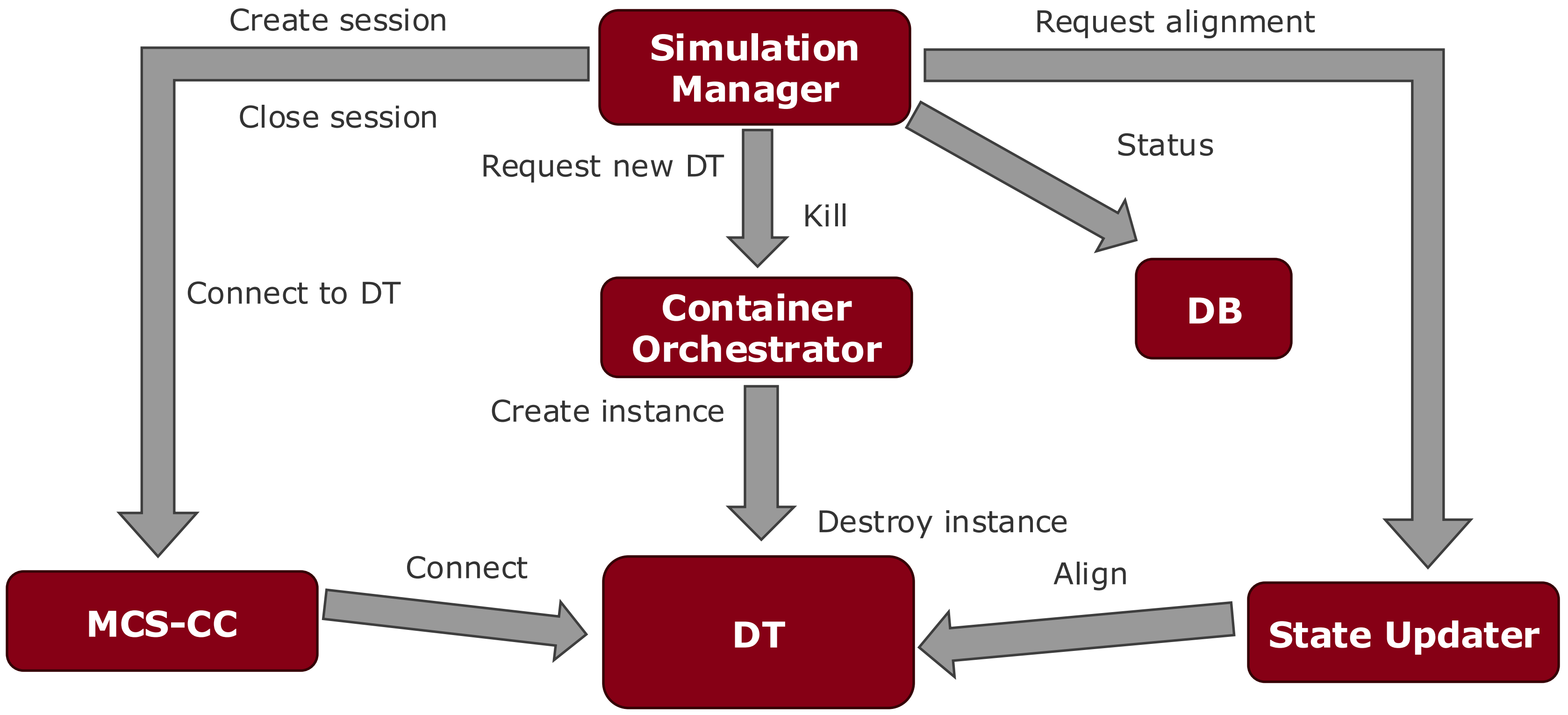
DT Lifecycle Management

DT Lifecycle Management

Required functions

- Start Digital Twin instance
  - Specify Mission for which to start a new Digital Twin
  - Provide the date to be used for the alignment
  - Decide whether to apply existing calibrations
  - Run given scenario to validate by executing pre-prepared activity list
- Stop Digital Twin instance
  - Clean resources
- Get status of a Digital Twin instance
  - Display status of the past/current Digital Twin instance
  - Maintain in DB basic characteristics of DT (id, creation/stop times, alignment date, state vector,…)
- List Digital Twin instances
  - Filter by mission and or state (started, aligned, stopped, error)

DT Management



DT Management

Demo

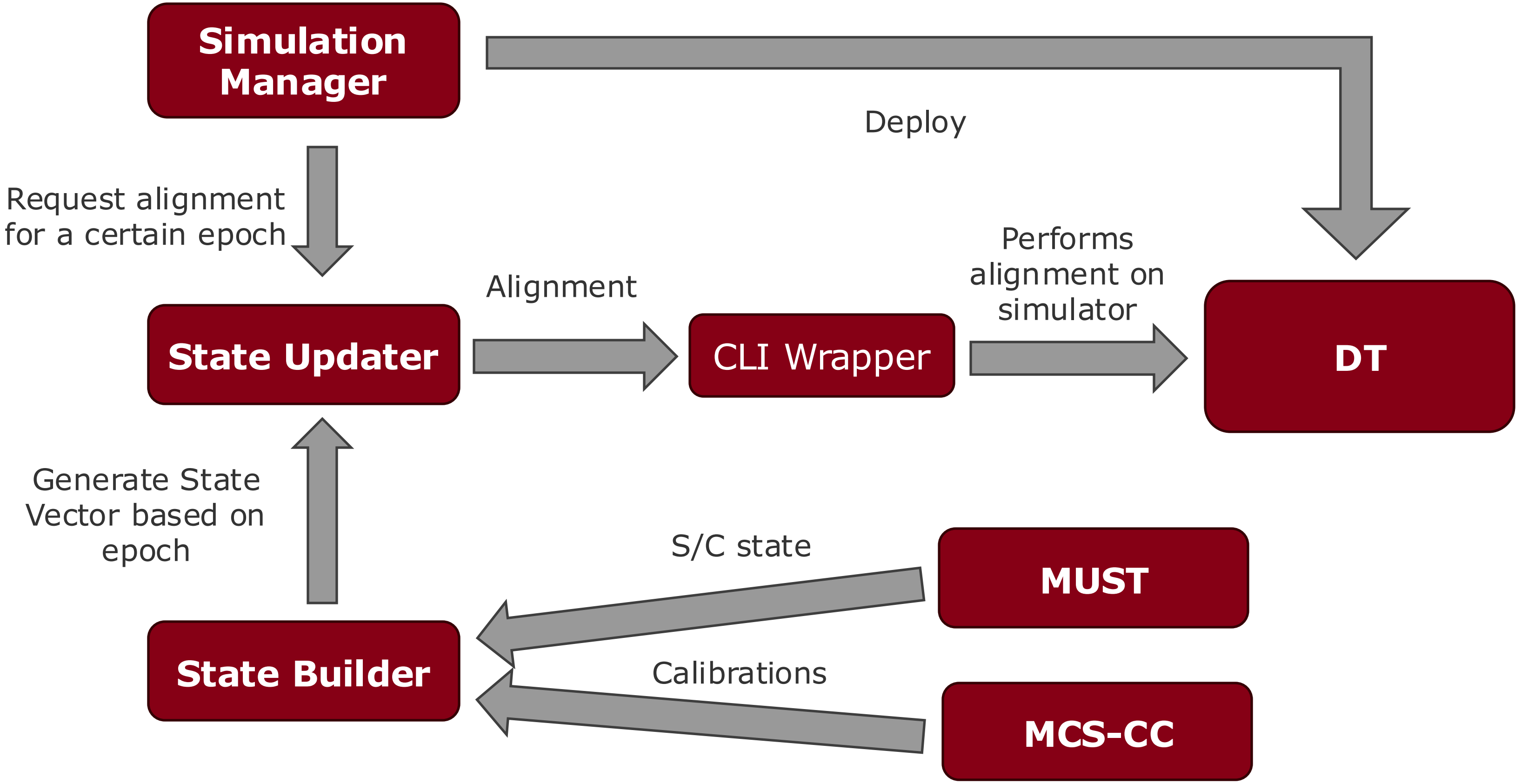
- Create a new instance of DT
- Connect via VNC to that instance
- Create the DT dedicated session in MCS-CC
- List all instances
- Get status of instance by ID

DT Alignment

Automatic DT alignment

State Vector	Formed by S/C telemetry parameters
	Defines the S/C state at a certain moment
	Used as a target state for alignment process
Alignment	Synchronize operational state
	Apply calibration data

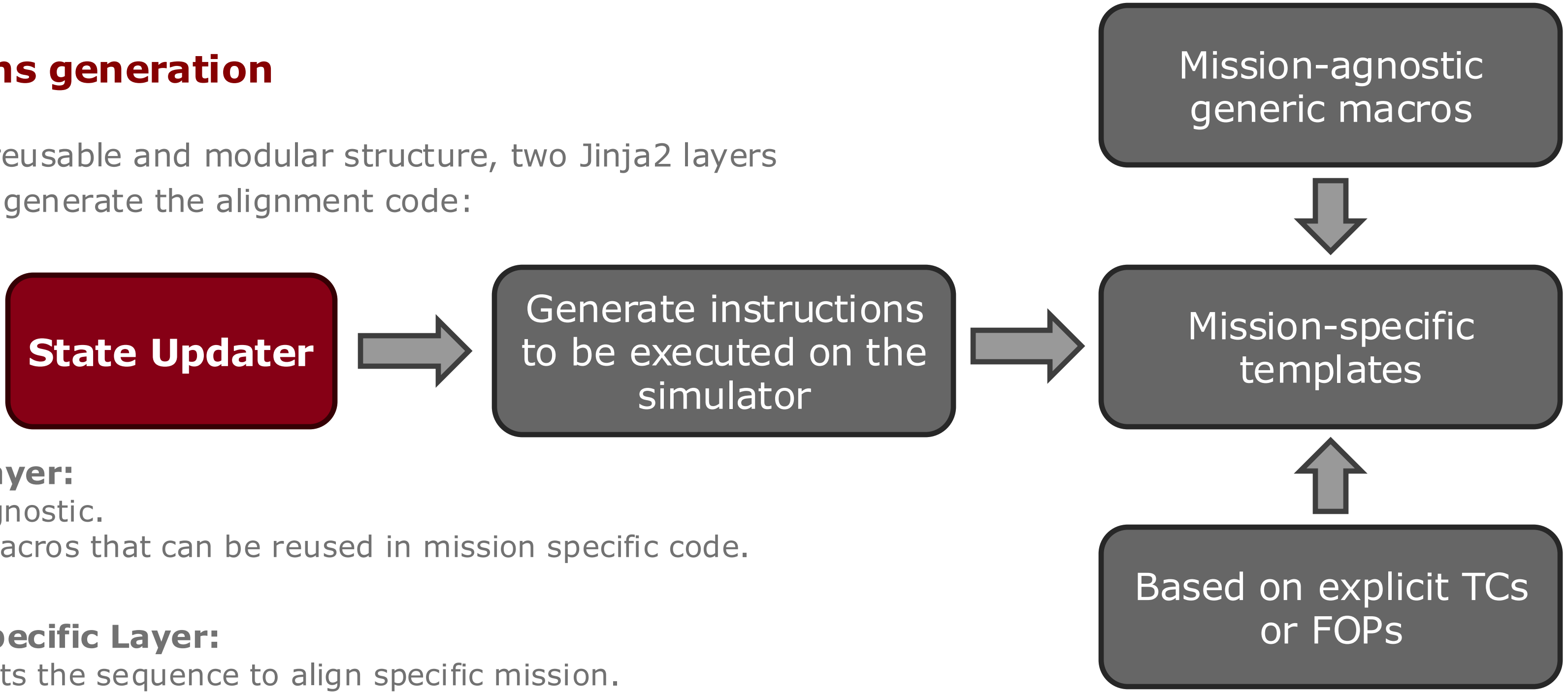
Alignment infrastructure


Simulation Manager
Deploy
Request alignment for a certain epoch
Alignment
Performs alignment on simulator
State Updater
CLI Wrapper
DT
Generate State Vector based on epoch
S/C state
MUST
Calibrations
State Builder
MCS-CC

Alignment

Instructions generation

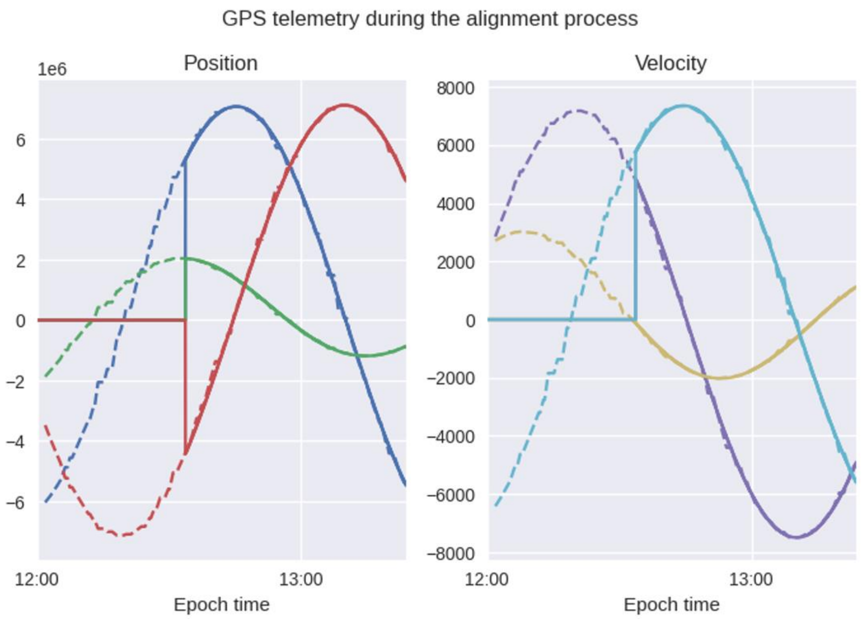
To provide a reusable and modular structure, two Jinja2 layers
Were used to generate the alignment code:

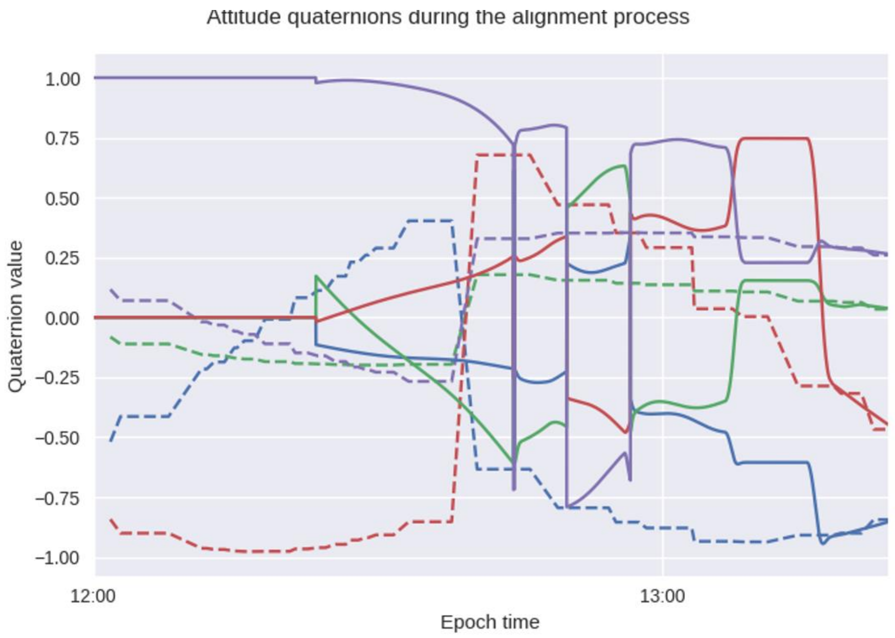

State Updater
Generate instructions to be executed on the simulator
Mission-agnostic generic macros
Mission-specific templates
Based on explicit TCs or FOPs

- **Generic Layer:**
  - Mission agnostic.
  - Defines macros that can be reused in mission specific code.

- **Mission Specific Layer:**
  - Implements the sequence to align specific mission.
  - Based on macros defined in Generic Layer.
  - Specifies the concrete TM parameter and commands, that are transparent to the Generic Layer.

Flying Mission Integration: Alignment





DT Monitoring & Control

MCS-CC

Monitors & Controls DT as if it was a real S/C

Dedicated system sessions provide isolated data spaces for experiments

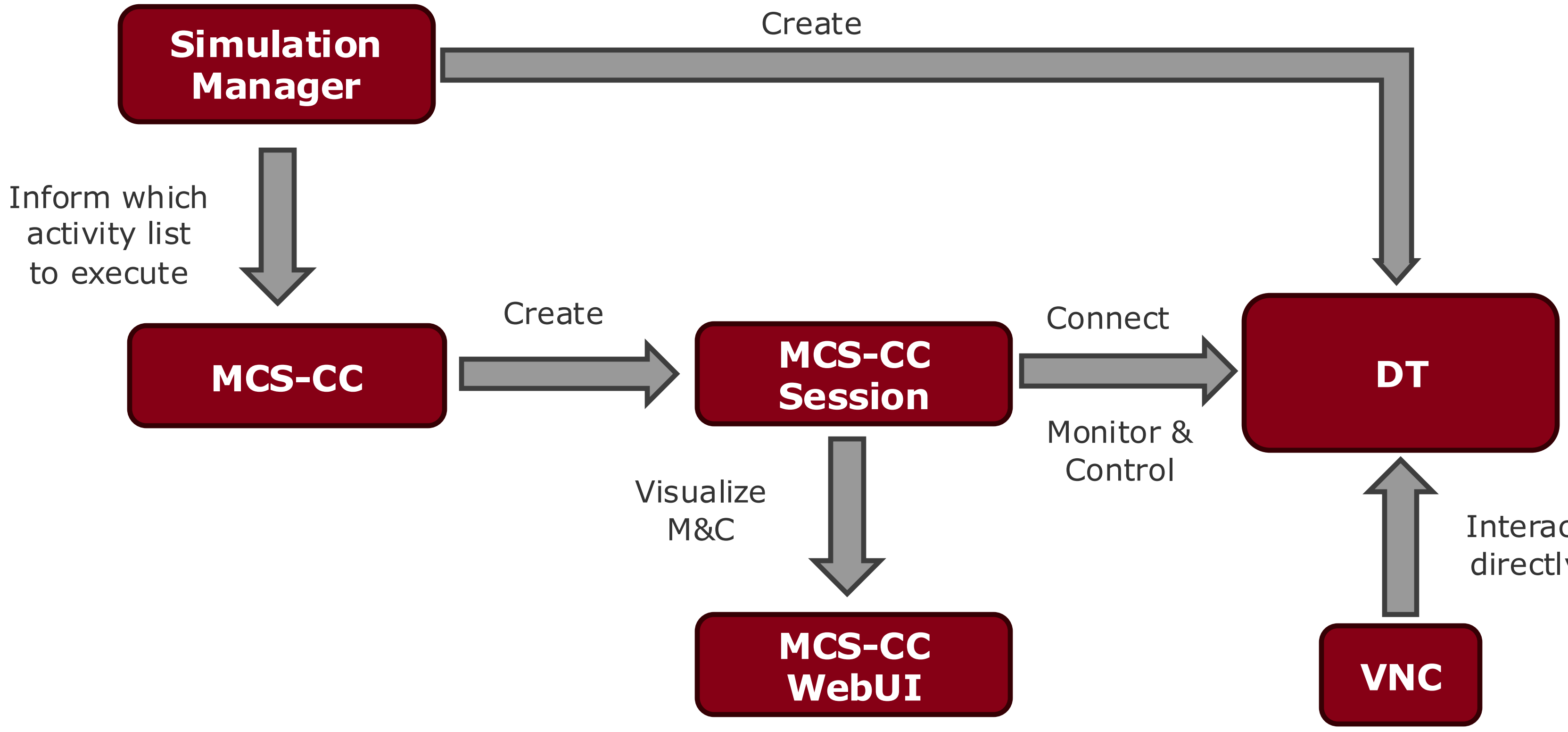
Automatic connection to the new DT instance after alignment

Automatic execution of predefined activity lists for scenarios evaluation

Extension to AInabler for accessing the DT data for analytical purposes

WebUI for visualizing Monitoring & Control data

DT Monitoring & Control



DT Monitoring & Control

Demo

- Creation of MCS-CC session
- Connecting to the session
- Checking TM and TC links status before and after alignment
- Observing that the activity stack specified during DT creation was executed
- Creating UDDs for visualizing selected TM parameters, e.g. the ones affected by the ML model incorporation

MCS-CC

DT Data access from AInabler

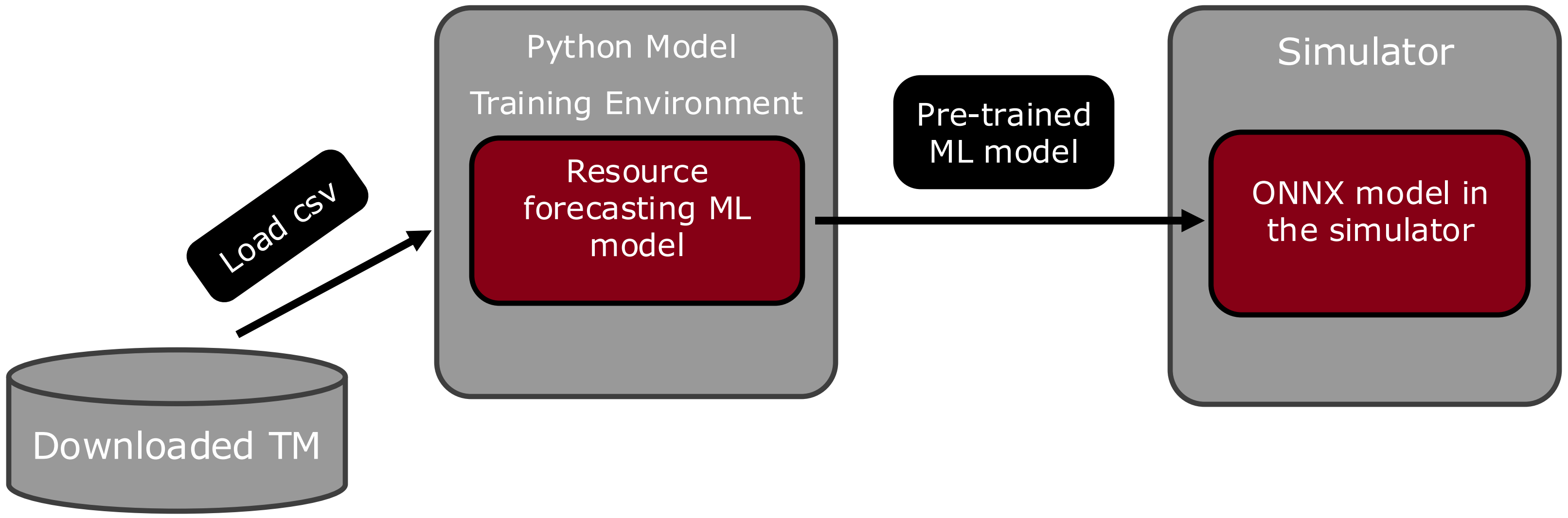
- Accessing real S/C data from MUST
- Accessing DT data from MCS-CC

AI/Optimization Models

Resource Usage Forecasting

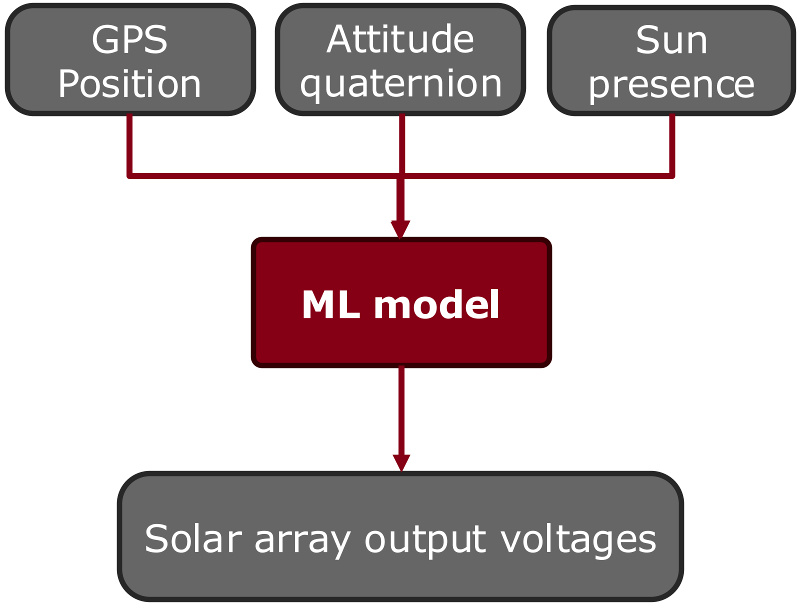
Resource Usage Forecasting

Goal: Resource forecasting of the solar array output voltages



Resource Usage Forecasting

Input data and pre-processing steps: one year of data



1. **Data Interpolation:** Interpolate attitude quaternion (q1, q2, q3, q4) and sun presence data using linear methods to match the GPS timestamp values.
2. **Coordinate Conversion:** Convert GPS coordinates into radius and unit vectors.
3. **Data Standardization:** Standardize the radius data.
4. **Error Removal:** Remove GPS data with position values of 0 or infinity

Resource Usage Forecasting

Comparison of Regression Models and their performances

Model	Description	Key Feature
OLSR (Ordinary Least Squares Regression)	Minimizes the sum of squared differences between observed and predicted values.	Basic linear regression without regularization
LASSO (Least Absolute Shrinkage and Selection Operator)	Uses L1 regularization, leading to some coefficients becoming zero.	Effective for variable selection.
Ridge Regression	Similar to OLS but includes L2 regularization, penalizing large coefficients.	Helps prevent overfitting
ElasticNet	Combines L1 and L2 regularization, useful for correlated variables.	Hybrid of Lasso and Ridge regression.

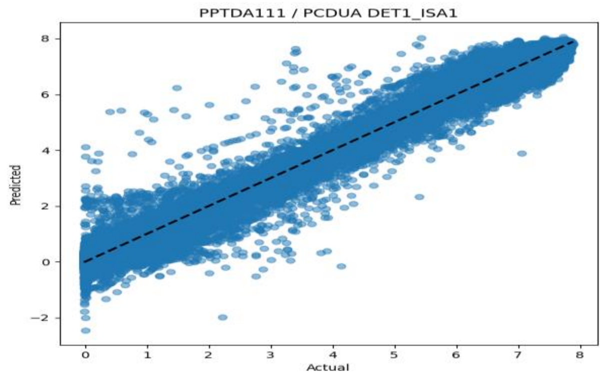
R-squared (r2) score value of all four models:

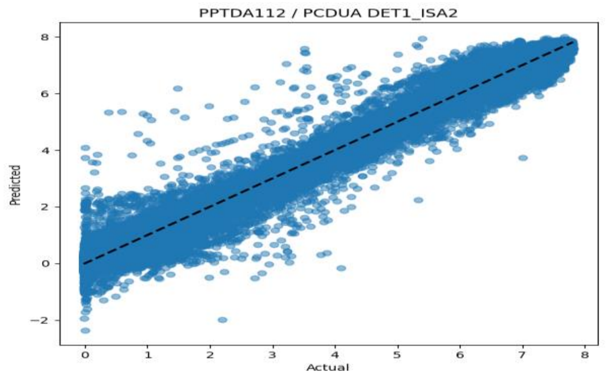
- `OLSR = 0.9926702477416942`
- `LASS = 0.7787996545678473`
- `ENET = 0.807793425468875`
- `RIDG = 0.992433476417404`

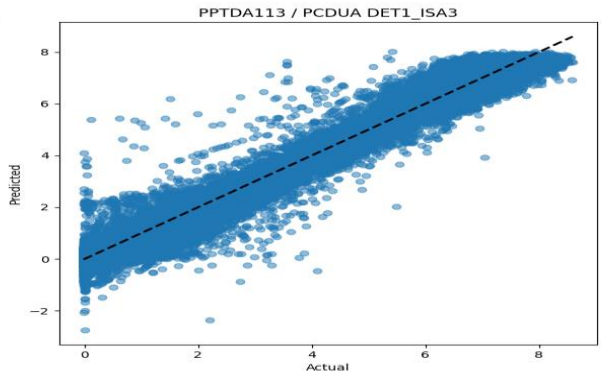
Ridge Regression showed better performance out of all other models

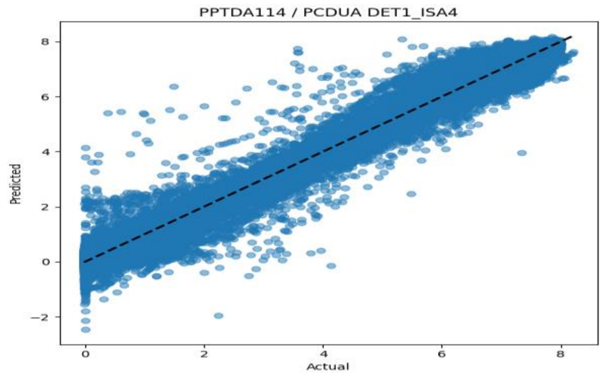
Resource Usage Forecasting

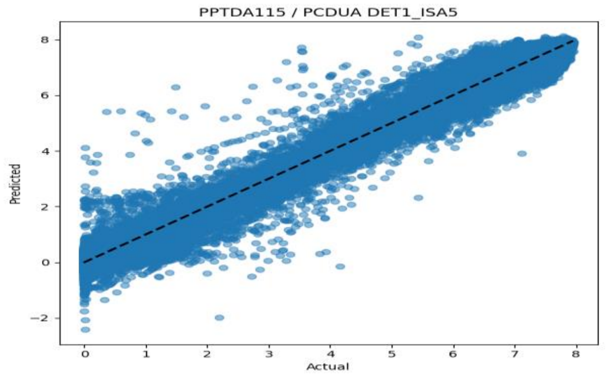
Ridge Regression Model: Predicted vs. Actual Values

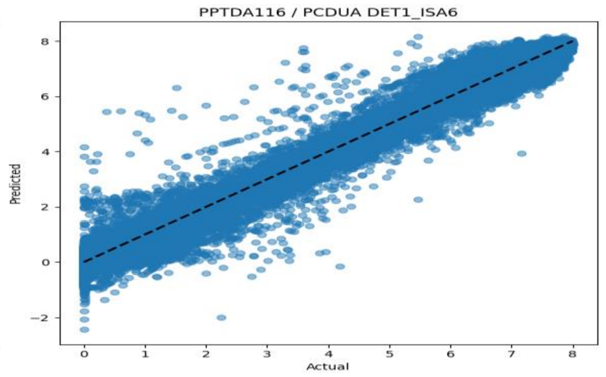






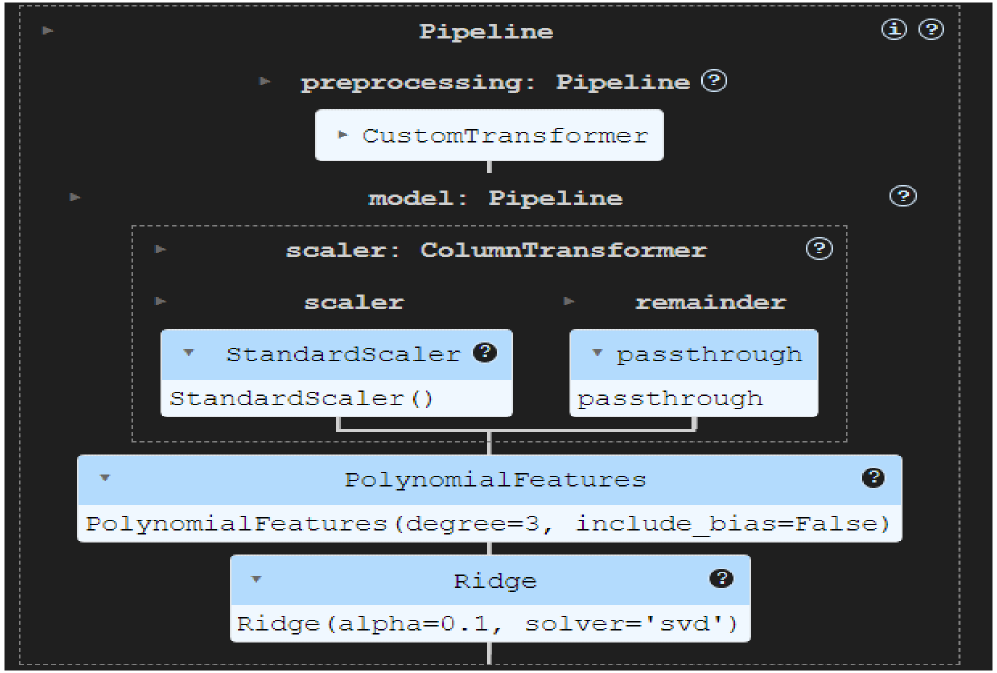






Resource Usage Forecasting

Model pipeline and Integration



```
Pipeline
  preprocessing: Pipeline
    CustomTransformer
  model: Pipeline
    scaler: ColumnTransformer
      scaler                 remainder
      StandardScaler         passthrough
      StandardScaler()       passthrough
    PolynomialFeatures
    PolynomialFeatures(degree=3, include_bias=False)
    Ridge
    Ridge(alpha=0.1, solver='svd')
```

Sklearn Pipelines:

- **class sklearn.pipeline. Pipeline(*steps*, *, *memory*=*None*, *verbose*=*False*)**
- **A sequence of data transformers** with an optional final **predictor**.
- **Purpose: Sequentially apply transformers** to preprocess data, concluding with a final predictor for modeling.

- **Model Training Pipeline:**
  - Develop a custom transformer for preprocessing the training data within the sklearn pipeline.
  - Construct and train the final model pipeline, saving the whole pipeline as a pickle file.
- **Inference Pipeline:**
  - Implement a custom transformer for preprocessing the test data.
  - Integrate the test data preprocessing step into the saved pipeline, Utilizing the previously stored model to process and analyze the test data.

Resource Usage Forecasting

Conclusions

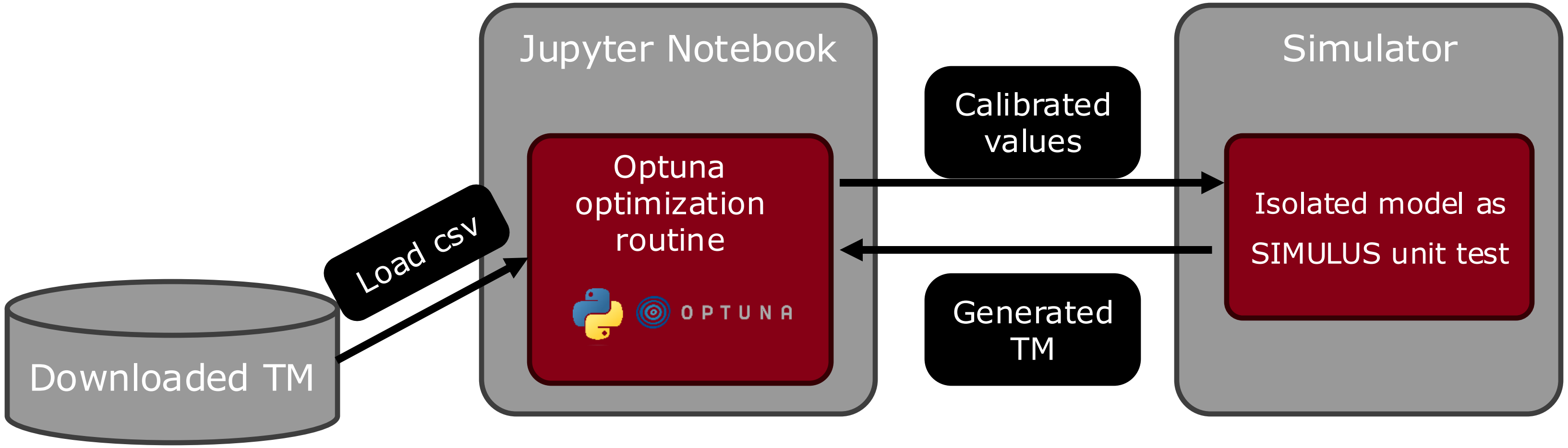
- **Model performance:** The Ridge Regression model achieved a high score of 0.99, demonstrating strong predictive ability for solar array output voltages.
- **Negative predictions**: The model sometimes produced negative voltage values
- **Recommendations for Improvement**:
  - Model constraints: Introduce constraints or transformations to prevent negative values.
  - Expanded Data: Use multi-year datasets to improve model generalization and account for seasonality.
  - Model refinement: Fine-tune the regularization parameter.
  - Implement advanced feature selection techniques.
  - Use ensemble methods to combine Ridge Regression with other models.
  - Explore alternative models, such as Support Vector Regression, Decision Tree Regression, Random Forest, and Neural Networks.
  - Include Physics Informed AI/ML models

Calibration

Isolated model for calibration

Goal of the calibration use case

Find the constants that, when set in the simulator models, lead to the most accurate TM



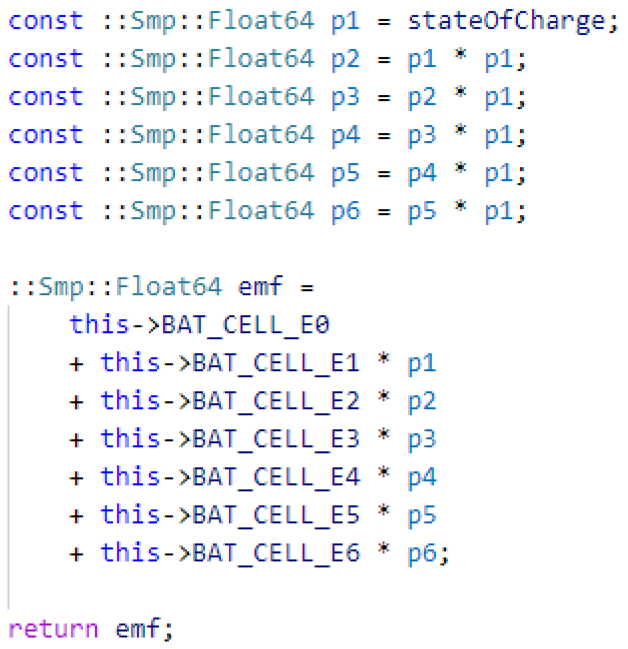
Python library Optuna used for optimizing hyperparameters.

Isolated model for calibration

Contents of the isolated model

1. Instantiate the battery
2. Set the selected values of the coefficients being calibrated
3. Simulate the battery for the same period of time as the downloaded TM
   - Set the intensity of charge/discharge to simulate same power load
   - Simulate for a ΔT
   - Send the new voltage to the optimization routine

Calculation of EMF in the simulator



```
const ::Smp::Float64 p1 = stateOfCharge;
const ::Smp::Float64 p2 = p1 * p1;
const ::Smp::Float64 p3 = p2 * p1;
const ::Smp::Float64 p4 = p3 * p1;
const ::Smp::Float64 p5 = p4 * p1;
const ::Smp::Float64 p6 = p5 * p1;

::Smp::Float64 emf =
    this->BAT_CELL_E0
    + this->BAT_CELL_E1 * p1
    + this->BAT_CELL_E2 * p2
    + this->BAT_CELL_E3 * p3
    + this->BAT_CELL_E4 * p4
    + this->BAT_CELL_E5 * p5
    + this->BAT_CELL_E6 * p6;

return emf;
```

Calibration

Optimization Algorithms

- **<u>Bayesian optimization:</u>**
  - ➢ Probabilistic model-based approach for optimizing black-box functions that are expensive to evaluate.
  - ➢ It uses a surrogate model (commonly a Gaussian Process) to approximate the objective function and an acquisition function to decide where to sample next.

- **<u>Genetic algorithm:</u>**
  - ➢ Inspired by the process of natural selection and are used for optimization and search problems.
  - ➢ They work by evolving a population of candidate solutions through selection, crossover (recombination), and mutation.

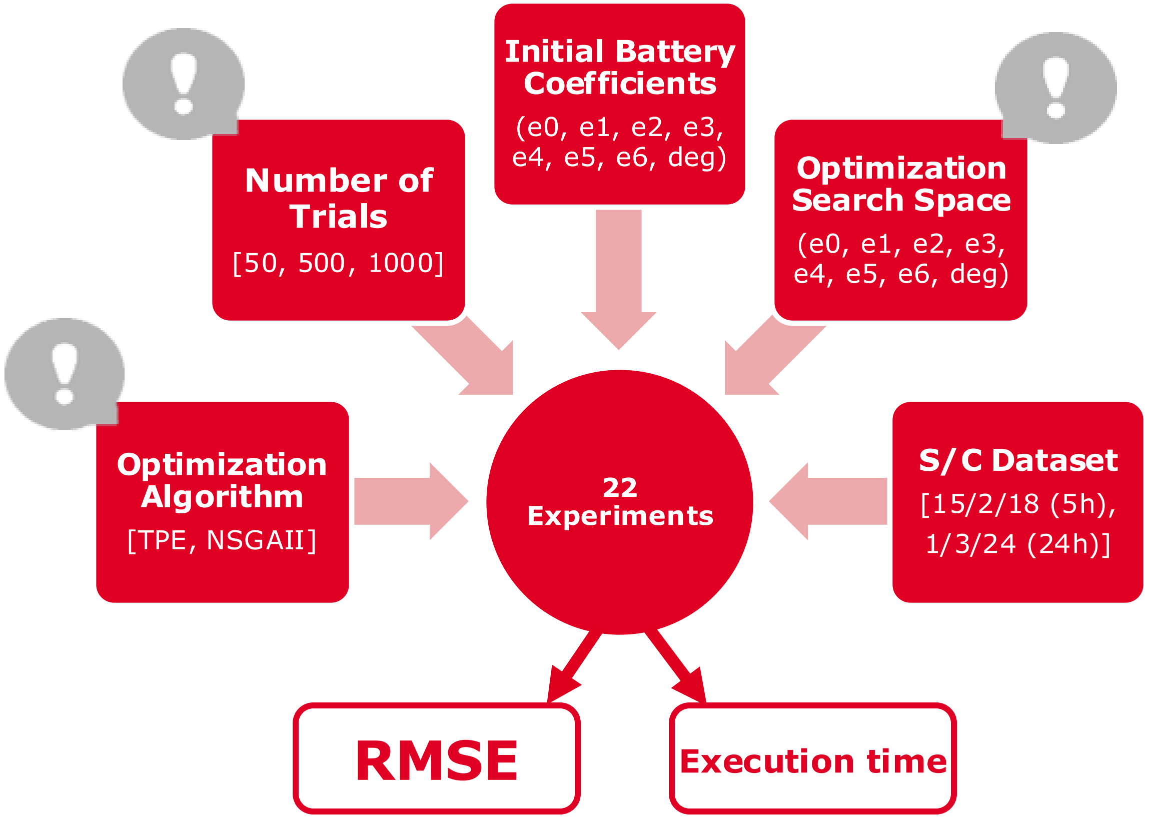
Feature	Bayesian Optimization	Genetic Algorithm
Model Type	Probabilistic model-based (e.g., Gaussian Process)	Population-based evolutionary algorithm
Function Evaluations	Efficient, fewer evaluations needed	Can require many evaluations
Search Strategy	Uses a surrogate model and acquisition function to balance exploration and exploitation	Uses evolutionary operators to explore and exploit the search space
Handling of Noise	Can handle noisy evaluations	Can handle noisy evaluations, though less explicitly
Dimensionality	Computationally intensive for high dimensions	Handles high-dimensional spaces better
Global vs Local Optima	Tends to find global optima	Good at avoiding local optima, but convergence can be slow
Parameter Tuning	Requires careful tuning of the surrogate model and acquisition function	Requires tuning of evolutionary parameters (e.g., population size, mutation rate)
Application Suitabilty	Expensive-to-evaluate, smooth functions	Combinatorial, complex, multi-modal functions

Calibration

Experiments

- Data: TM parameters downloaded from MUST
- *2 datasets: shorter and longer timespan*
- Complemented with information from
    - tdbed
    - SC mib
- Dataset Variables: Voltage, Batt Charge/Discharge, Depth of Discharge
- Calculated variables: Intensity (Charge-Discharge)
- *Simulator data*:
    - 7 battery coefficients
    - Battery degradation

coeff	init_value	mult_min	mult_max
e0	2.65852	0.25	1.25
e1	0.0575422	0.85	1.15
e2	-0.0008937	0.9945	1.0055
e3	5.3843E-06	0.985	1.015
e4	-1.651E-08	0.985	1.015
e5	2.9032E-10	0.985	1.015
e6	-1.92E-12	0.98	1.02
deg	rand(0, 10)	0	10

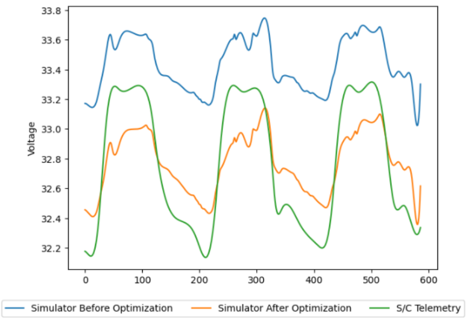

Initial Battery Coefficients
(e0, e1, e2, e3, e4, e5, e6, deg)
Number of Trials
[50, 500, 1000]
Optimization Search Space
(e0, e1, e2, e3, e4, e5, e6, deg)
Optimization Algorithm
[TPE, NSGAII]
22 Experiments
S/C Dataset
[15/2/18 (5h), 1/3/24 (24h)]
RMSE
Execution time

Calibration

Results (S/C Dataset 1)

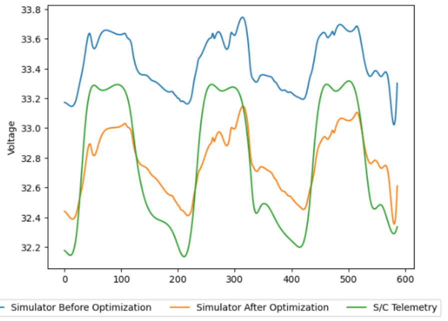
TPE (n_trials=1000)

After optimizing EMF formula (MAE = 0.24, RMSE = 0.26)



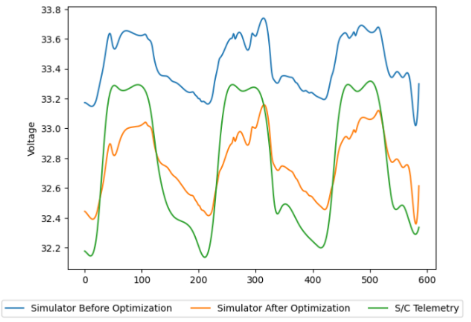
TPE (n_trials=500)

After optimizing EMF formula (MAE = 0.24, RMSE = 0.26)



NSGAII (n_trials=1000)

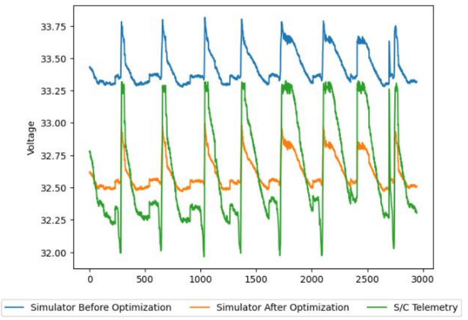
After optimizing EMF formula (MAE = 0.24, RMSE = 0.26)



Results (S/C Dataset 2)

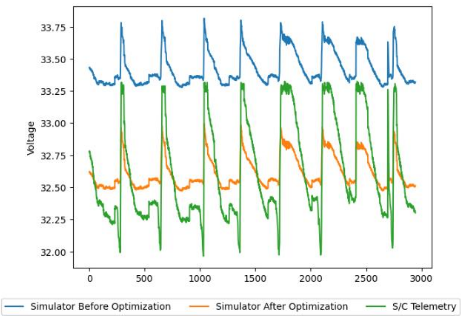
TPE (n_trials=1000)

After optimizing EMF formula (MAE = 0.22, RMSE = 0.26)



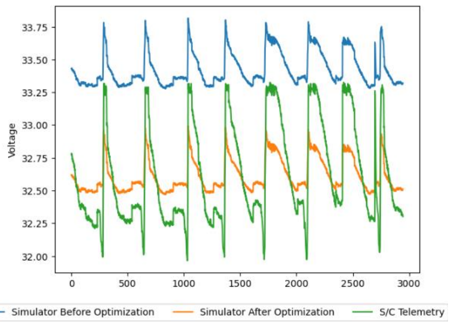
TPE (n_trials=500)

After optimizing EMF formula (MAE = 0.22, RMSE = 0.26)



NSGAII (n_trials=1000)

After optimizing EMF formula (MAE = 0.22, RMSE = 0.26)



Calibration

Conclusions

- **Simulation results are better** with the new **calibration** procedure.

- **Trade-off** between **number of trials** and **execution time:**
  - **Optimization process seems to converge, but it takes time/trials.**

- **TPE sampler** algorithm gives the best results for optimization, although genetic sampler gives very similar results.

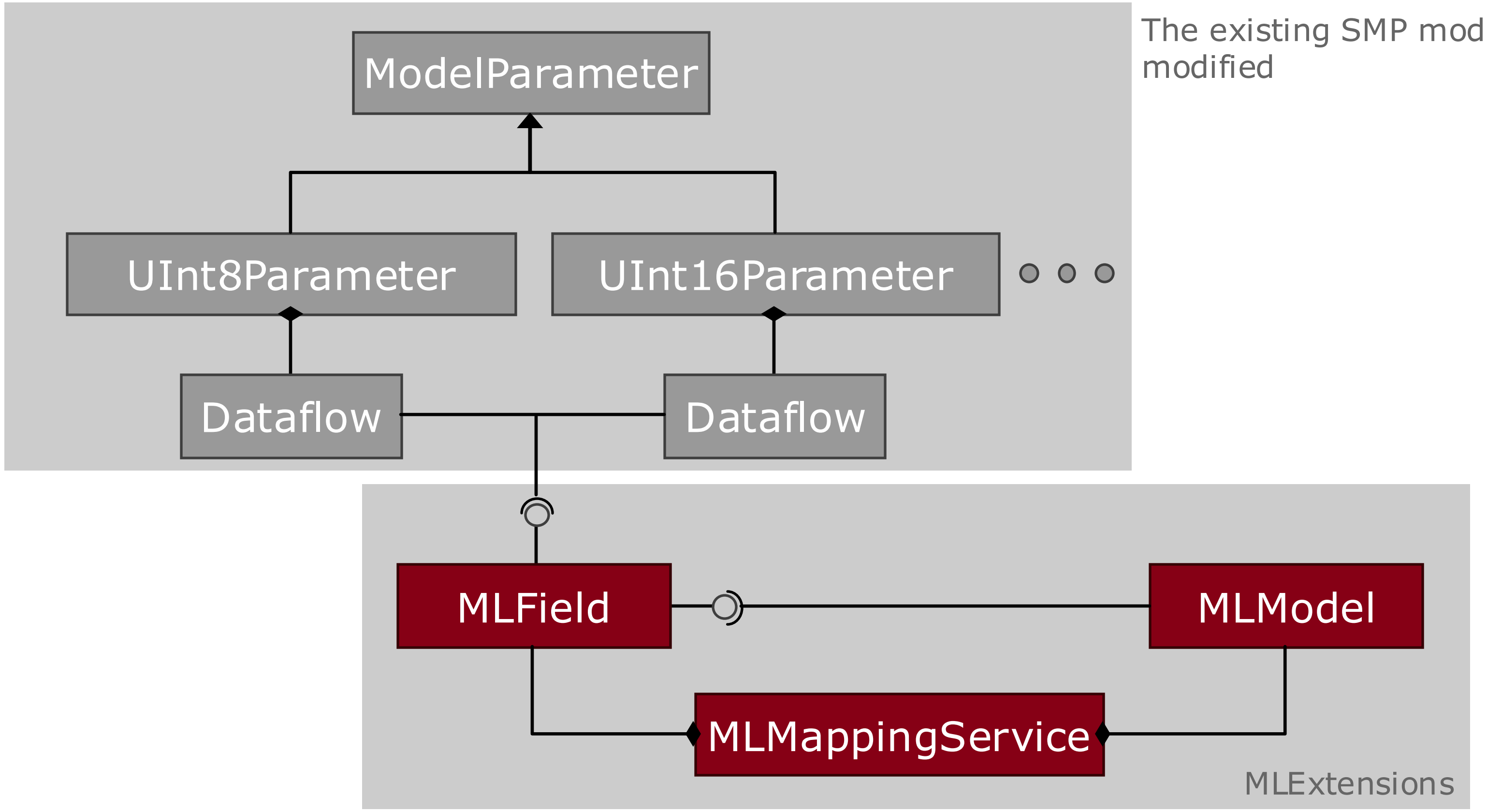
- With **5h of data**, the optimization process achieves **good results that are similar to 24h** dataset experiments.

- Establish a **good search space** is important, and also the **coefficients' first initialization**.

- The **most important coefficients** can be **identified** (*e1*, *e0* and *deg* in this case).

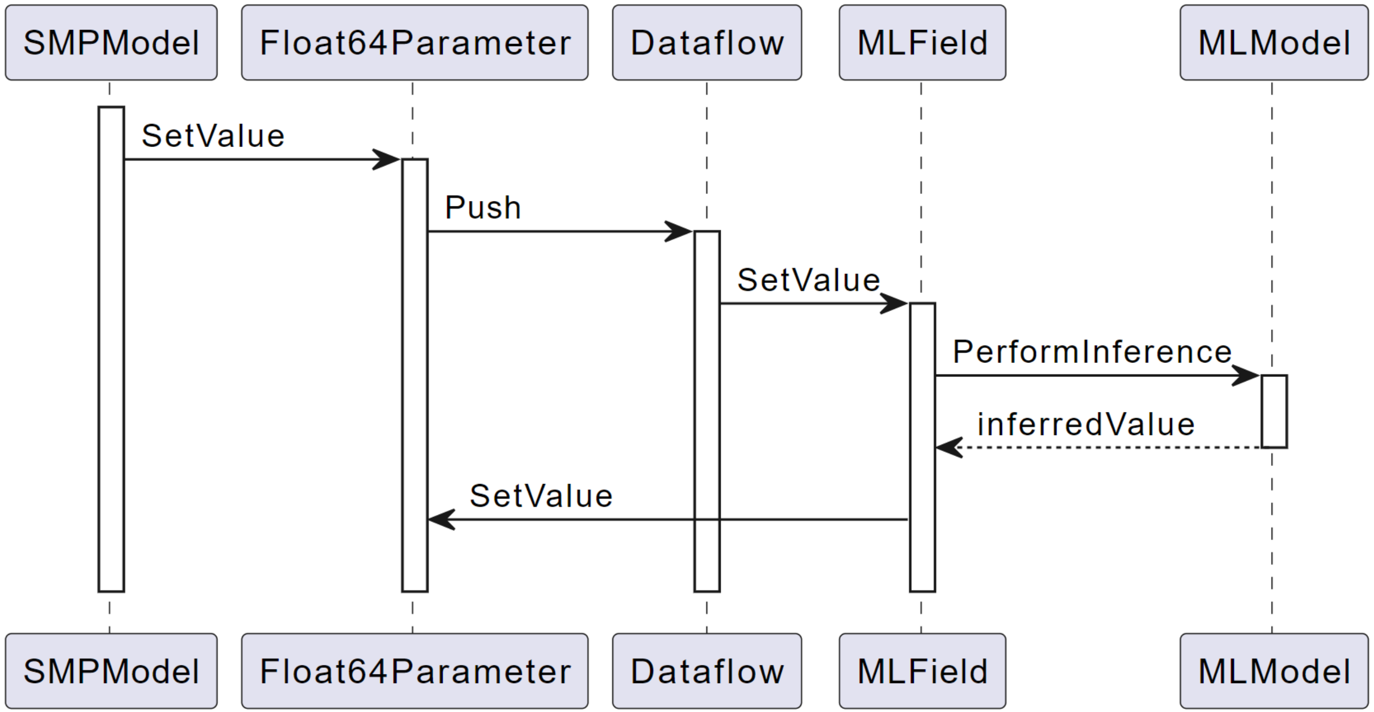
AI models integration into DT simulator

Design



The existing SMP model is not modified

Inference process

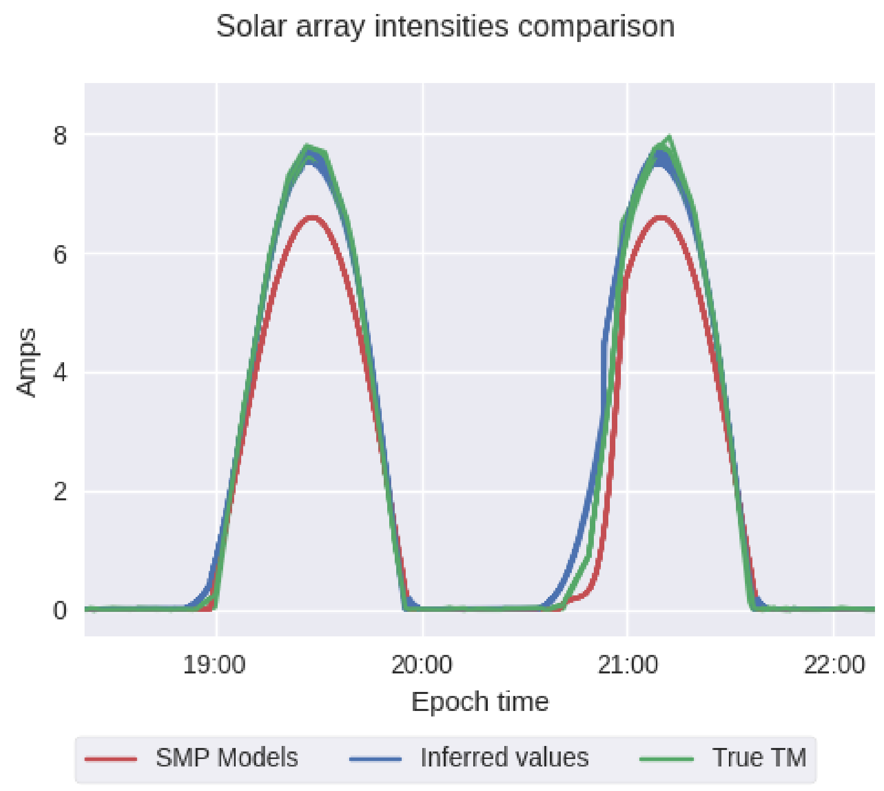


ML Model Configuration

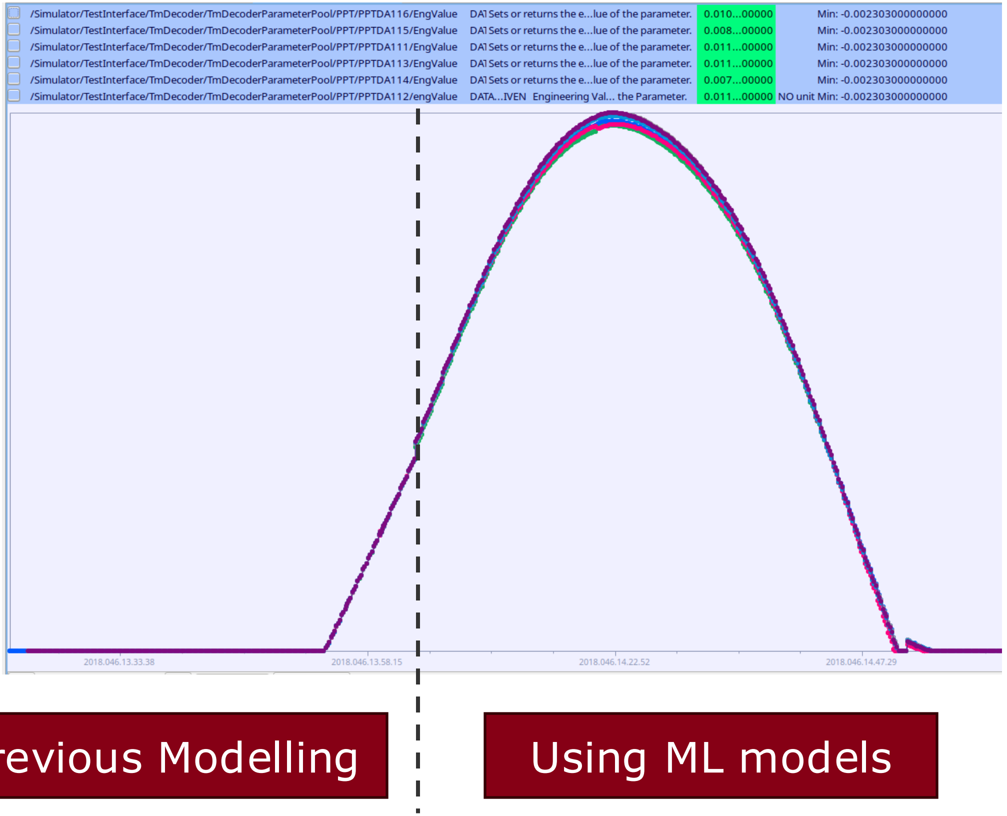
- MLModels/MLFields created from XML
- No assemblies
- No need to recreate breakpoints
- Any field in the simulator can be an input
- Perform inference on value updates and Schedule.
- Enable/Disable inferred value propagation
- Upper & lower limit

```xml
<MLMapping …>
    <Model Path="share/data/mlmapping/solar_array.onnx" Name="SolarArrayIntensity" ValidityPeriod="0">
        <Input Path="TestInterface/TmDecoder/TmDecoderParameterPool/<path>/engValue"/>
        <Input Path="TestInterface/TmDecoder/TmDecoderParameterPool/<path>/engValue"/>
        …
        <Output Path="Spacecraft/EPS/Pcdu/SaPC/S3R1" Parameter="inputCurrentTm" Interval="1" Lower="0.01"/>
        <Output Path="Spacecraft/EPS/Pcdu/SaPC/S3R2" Parameter="inputCurrentTm" Interval="1" Lower="0.01"/>
        …
    </Model>
</MLMapping>
```

Results


Solar array intensities comparison
Amps
0
2
4
6
8
19:00
20:00
21:00
22:00
Epoch time
SMP Models
Inferred values
True TM

Results



/Simulator/TestInterface/TmDecoder/TmDecoderParameterPool/PPT/PPTDA116/EngValue	DAI	Sets or returns the e...lue of the parameter.	0.010...00000		Min: -0.002303000000000
/Simulator/TestInterface/TmDecoder/TmDecoderParameterPool/PPT/PPTDA115/EngValue	DAI	Sets or returns the e...lue of the parameter.	0.008...00000		Min: -0.002303000000000
/Simulator/TestInterface/TmDecoder/TmDecoderParameterPool/PPT/PPTDA111/EngValue	DAI	Sets or returns the e...lue of the parameter.	0.011...00000		Min: -0.002303000000000
/Simulator/TestInterface/TmDecoder/TmDecoderParameterPool/PPT/PPTDA113/EngValue	DAI	Sets or returns the e...lue of the parameter.	0.011...00000		Min: -0.002303000000000
/Simulator/TestInterface/TmDecoder/TmDecoderParameterPool/PPT/PPTDA114/EngValue	DAI	Sets or returns the e...lue of the parameter.	0.007...00000		Min: -0.002303000000000
/Simulator/TestInterface/TmDecoder/TmDecoderParameterPool/PPT/PPTDA112/engValue	DATA...IVEN	Engineering Val... the Parameter.	0.011...00000	NO unit	Min: -0.002303000000000

Storing DT internal state

INNOVATING SOLUTIONS

MCS-CC

Flying mission tailoring extended to contain DT internal state parameters

SIMSAT Remote Connector to publish internal parameter values

MCS-CC extended to read and inject SIM internal parameters

Internal parameters accessible as any others (GUI, REST API)

Tailoring

Demo

- Flying mission tailoring extension to contain internal SIM parameters
- Publishing of DT internal parameters
- Injection of DT internal parameters
- Visualization of DT internal parameters on UDD

Conclusions & future steps

Conclusions

Achievements:

- Creation of the DT infrastructure
- Reuse of ESA systems
- DT Alignment
- AI/ML models improving SIM fidelity
- Validation against real mission

● ● ●

Future steps

Future tasks to achieve a better system:

- Mature the prototype into a production system
- Add support of different missions
- Alignment of all S/C systems, including OBC memory
- Continuous synchronization and calibration
- Implementation of analytic UCs

● ● ●

Thank you

Questions?

Maciej Prokopczyk	GMV PL
Maciej Szreter	GMV PL
Anna Banachowicz	GMV PL
David del Val	GMV ES
Leandro Garcia	GMV ES
Yusra Al-Khazraji	GMV DE
Inmaculada Perea	GMV ES
Vanessa Navarro	GMV ES
Sepideh Rahimian	GMV DE
Daniele Segneri	ESA/ESOC
Federico Antonello	ESA/ESOC

INNOVATING SOLUTIONS

Icon attribution: Flaticon.com