1.Service description

PdM2 service is designed to monitor the health status of different SEW production modules, where the evolution of the frequency of historical failures is used as the health indicator. Such monitoring is carried out automatically (performed by an external MODAPTO component) with a frequency that is specific to each module, and it results in inspection or replacement suggestions for the most critical sub elements that compose a certain module (for more contextual details please refer to Deliverable 5.2).

The service is structured as a python class called PdM2, which can be invoked. See below:

PdM2Service( input\_data, winds\_count\_component\_replac=3, output\_path="maintenance\_recommendations.json")

The PdM2Service arguments necessary to instantiate the class are detailed below.

2.Inputs

“Input data” requires a JSON object that should contain historical failure data and parameters for module monitoring. An example of a JSON object is provided below while a description of its content is provided hereafter.

{

"events": [

{

"Stage": "LIGNE DE MONTAGE MOTG04",

"Cell": 100000,

"Module": 100019,

"Component": 100020,

"Failure type (electrical/mechanical)": "Mechanic",

"Failure description": "ascenseur ne monte pas retournement",

"Maintenance action performed": "Réglage capteur sorti indexeur\n----- Mickael KLEIN 16/06/2023 08:01 -----",

"Component replacement (yes/no)": "no",

"Name of worker": "Carlos M",

"TS Intervention started": "16/06/2023 06:38:18"

},

{

"Stage": "LIGNE DE MONTAGE IMS MOTG",

"Cell": 100000,

"Module": 100019,

"Component": 100312,

"Failure type (electrical/mechanical)": "Mechanic",

"Failure description": "Donner suite à la panne sur les rippeurs. Remise en état !",

"Maintenance action performed": "Intervention réalisée le 22 novembre 2023\r\n----- Damien Kuhn 23/11/2023 08:34 -----",

"Component replacement (yes/no)": "no",

"Name of worker": "Carlos M",

"TS Intervention started": "08/11/2023 10:13:56"

}

],

"parameters":{

"module\_ID": 100019,

"components\_ID": [100020, 100021, 100022, 100023, 100024],

"window\_size": 90,

"inspection\_threshold": 3,

"replacement\_threshold": 7

}

}

Description of the content of the JSON object

**Events:** up-to-date failure records (drift table). It contains details about failure and maintenance events such as: “Stage”, “Cell”, “Module”, “Component”; “Failure type”, “Failure description”, “Maintenance action performed”, “Component replacement (yes/no)”, “Name of worker”, “TS Intervention started”.

**Parameters:** parameters for monitoring a module. It contains: the “module ID”, which defines the module to be monitored; “components ID”, which defines an array of components that constitute the module and for which maintenance suggestions are desired; the “window\_size”, which defines the amount of data (failure period in days) used to measure the health of the module; “inspection\_threshold”, which is the number of failures that triggers an inspection; “replacement\_threshold”, which is the number of failures that triggers a replacement. Note that all these parameters are fixed but they are module-specific, as so, the API should enable the user to configure these values for each production module within the MODAPTO environment and add new production modules to be monitored with the respective parameters if necessary. Such values are defined based on expert-knowledge or exploratory analysis of failure data and are they do not change dynamically.

While the JSON input was provided beforehand, an input sample is also provided as .json file in the sharepoint repository under the name “CORIM\_tool\_test.json”.

The second PdM2Service argument called “winds\_count\_component\_replac” is the number of time windows to measure the health state of the sub elements and used to define the most critical sub element to be replaced. It has a fixed value and is independent of the module or component. Consequently, it does not need to be provided when invoking the class, i.e., the default value is enough.

The third PdM2Service argument “output\_path="maintenance\_recommendations.json" corresponds to the path where the maintenance suggestions will be stored, with “maintenance\_recommendations.json” as the default path/file name.

3.Output

The output of the algorithm is a .json file with default name “maintenance\_recommendations.json”. The file follows the following format when no recommendation is given.

{

"recommendation": "non",

"details": "The system is under control"

}

The file follows the following format when a recommendation is given.

{

"recommendation": "schedule a replacement of sub element 100019",

"details": "It failed 16 times in the last 270 days"

}

* “recommendation” corresponds to the maintenance actions to be scheduled in the next available maintenance slot.
* “details” corresponds to contextual information that justify the maintenance recommendation.

Note that the last output scenario (when a recommendation is given) should be converted to notifications, which can be used to schedule maintenance actions and to update production schedules accordingly. As agreed between UL and ATC: “These notifications should be incorporated as part of the input for Optimization Service. It can be done manually by creating assignments from the PdM2 notifications to re-schedule and add the details of the notification to the Opt.Service input”.

4.Requirements for API

The service should contain a configuration button/area that configure PdM2 monitoring for different production modules and potential new modules. The user (potentially in charge of maintaining the code/service) can input a set of modules of interest to be monitored and its specific monitoring frequencies. This input will also contain “window\_size”, “components ID”, “inspection\_threshold”, and “replacement\_threshold”.

An external scheduler triggers the service automatically given the configuration of PdM2, i.e., the set of modules to be monitored and the respective frequency of each module. The scheduler then passes an input to the PdM2 algorithm as JSON object, containing the information defined in the JSON example provided in section 2 and tailored to the specific production module to be monitored, i.e., the most up-to-date failure data (historical drift table) and the parameters for monitoring the module.

Given that the logic for generalization (different production modules) was already stablished, the initial set of modules to be monitored for this version could contain just the module press (100019) with the following parameter values:

{

"100019": {

"monitoring\_frequency": 5,

"components\_ID": [100020, 100021, 100022, 100023, 100024],

"window\_size": 90,

"inspection\_threshold": 3,

"replacement\_threshold": 7

}

}

This set can be extended by the user if a new module is added to SEW production or if other modules are considered for monitoring. For the moment more modules are not added since the monitoring parameters have been only validated for the press.

**Invoking PdM2**

An example for invoking PdM2 is provided below using the JSON example provided in section 2.

# Example of usage:

from pdm2\_service import PdM2Service

# This is where your API receives input\_data (e.g., via POST request)

input\_data = {

"events": [...],

"parameters": {...}

}

service = PdM2Service(input\_data=input\_data)

output = service.run()