4th Asia Pacific Conference of the Prognostics and Health Management,
Tokyo, Japan, September 11 – 14, 2023 R10-02

Architecting a Digital Twin-Based Predictive Maintenance System for Modelling Cable Joint Degradation

Raymon van Dinter[1], Görkem Ekmekci[2], Sander Rieken[3], Bedir Tekinerdogan[4], and Cagatay Catal[5]

[1,2] *Sioux Technologies, Apeldoorn, The Netherlands*
raymon.van.dinter@sioux.eu
gorkem.ekmekci@sioux.eu

[1,4] *Information Technology Group, Wageningen University & Research, Wageningen, The Netherlands*
raymon.vandinter@wur.nl
bedir.tekinerdogan@wur.nl

[3] *Alliander, Arnhem, The Netherlands*
sander.rieken@alliander.com

[5] *Department of Computer Science and Engineering, Qatar University, Doha, Qatar*
ccatal@qu.edu.qa

Abstract

The large scale adoption of wind turbines and solar panels in the Netherlands places new demands on the medium voltage power grid. For example, highly varying loads can cause failures in certain cables. Cable joints are natural weak spots prone to faults due to varying currents, creating downtime challenges for public utility companies. Predictive maintenance (PdM) practices are necessary to minimize downtime for users. We present a Model-Based System Engineering approach using formal models and UML views to provide a scalable PdM design ontology for modeling cable joint degradation. We aim to monitor cable joint degradation from different manufacturers under varying conditions throughout the Netherlands in real-time using a Digital Twin (DT) approach. Our design provides high-resolution, real-time synchronization between the DT-based PdM system and the cable joints. The proposed architecture is scalable, robust, and flexible, and the software implementation is publicly available in an open-source repository.

1. Introduction

The ongoing energy transition in the Netherlands from a centralized to a decentralized system, driven by the increasing adoption of wind turbines and solar panels, has resulted in significant changes to the Dutch Medium Voltage (MV) electricity grid. As a result of the increasing adoption of sustainable energy sources, MV grids are experiencing increasing variations in load and bi-directionality. Cable joints, the connections between two power cables, are natural weak spots constructed manually on-site and prone to faults due to varying currents. These weak spots create challenges for public utility companies as they aim to minimize user downtime. Therefore, effective predictive maintenance (PdM) or Prognostics and Health Management (PHM) practices are essential to tackle these growing downtime challenges. Partial discharges (PDs) are a symptom of a weak spot in a power cable joint that could lead to a fault. Diagnosing PDs is a proven method to assess the condition of underground power cable joints. A PD induces a small pulse in the conductor(s) and earth screen that propagates through the cable in both directions (Wagenaars, 2010). Alliander, a public utility company in the Netherlands, uses a non-intrusive system to detect PDs in real-time using sensors on two ends of the MV circuit.

Many previous studies have attempted to model cable joint degradation using mathematical, physical, or document-centric engineering approaches. However, using these methods in isolation is a limitation, as they often model a single component or process. The complex ”systems-of-systems” behavior that make up the entire MV grid can be represented by integrating models with different viewpoints into a single system. In this study, we present a novel Model-Based System Engineering (MBSE) approach using formal models for modeling the ontology of the MV grid and for providing a smart and scalable PdM solution for cable joint degradation. To model the views of the different concerns of cable joint degradation, we use UML views by Kruchten (1995) and Mall (2018). The ambition level of our PdM system is to monitor the degradation of various types of cable joints from different manufacturers under varying conditions throughout the Netherlands in real-time. The eventual goal is to provide a Remaining Useful Life module to accurately predict when a fault in a cable joint will occur. We can access high-quality historical weather and PD data, providing information on the joint's condition and environmental influences. As such, we can apply a Digital Twin (DT) approach (Tiddens, 2018). Our MBSE application provides high-resolution, real-time synchronization between the DT-based PdM system and the cable joints, providing a more effective and efficient method of monitoring and predicting the health of cable joints in the MV context. As a first application, we focus on joints that show PDs due to either (1) temperature-related expansion or contraction, (2) water ingress, or (3) both. These defects have a high incidence rate, and when detected early, it is possible to intervene before the fault occurs. Our approach has the potential to significantly improve the PHM of cable joints and reduce the costs associated with PD-related downtime.

2. Related Work

This study is related to numerous other studies. Tiddens (2018) defines a framework for defining a business case and selecting the optimal PdM approach. Cocheteux, Voisin, Levrat, and Iung (2009) provide a methodology for selecting failure modes and prognostic models for the design process. Li, Verhagen, and Curran (2018) developed a functional architecture for PHM using a system engineering approach (Li et al., 2018). Aizpurua and Catterson (2015, 2016) showcase a methodology for designing PdM systems. Vogl, Weiss, and Donmez (2014) provide an overview of standards for PHM systems. Previous work on math-based computations to predict power outages due to PD and weather has been conducted by van Osch (2021).

3. Methodology

We adopted a MBSE approach to design the DT-based PdM system, which allowed for systematic and efficient system development. The MBSE approach involved using UML to create a detailed software design for the system using four views: User view, Implementation view, Behavioral view, and Environmental view (Kruchten, 1995; Mall, 2018). The Structural View was omitted, as it provides too much detail for this paper. The User view included a context diagram and use case diagram to give a high-level understanding of the system from the user's perspective. The Implementation view included a component diagram, which specified the system's structural organization and the interrelationships between its components. The Behavioral view included an activity diagram describing the system's dynamic behavior and the interactions between its components. Finally, the Environmental view included a deployment diagram illustrating the physical hardware components that make up the system and their connections. The MBSE approach began with defining the business case and requirements of the DT-based PdM system in collaboration with all stakeholders, including Alliander, the primary stakeholder. These requirements were then transformed into a software design specification linked to the Reference Architecture's feature model by van Dinter, Tekinerdogan, and Catal (2023) to determine the necessary components. The UML models were created based on the software design specification, providing a detailed representation of the system's structure and behavior. In addition to the UML models, code and unit tests were developed and made publicly available by van Dinter, Ekmekci, et al. (2023).

4. Results

4.1. User View

The PdM software can be executed locally and can, for instance, forecast PDs or predict the Remaining Useful Life (RUL) of cable joints. The Object-Oriented software uses input arguments from a technical user. The context diagram is shown in Figure 1, and the use case diagram is shown in Figure 2. The following paragraphs will provide more detail on the design of both diagrams. The technical user, typically a data scientist, can provide a train or test command, which uses a set of failed joints and joints with a seasonal PD pattern, or a predict command, together with a list of circuit IDs and joint locations, to get predictions of the specific joints. The train command always calls the test command after training the model. After providing a command, the software launches the *PredictiveMaintenanceJob*, and loads (1) information on failed joints, (2) joints with seasonal PD patterns, and (3) circuit coordinates. We load each circuit's coordinates, request circuit-level weather data from Alliander's weather API (Alliander, n.d.), and request circuit configurations (i.e., configs) and circuit-level PD data from local storage or Cloud Environment. The circuit config provides information to build a *Joint* object with location-specific PD. The models use PD and circuit weather data to train, and their results are returned to the user. The model parameters are saved to local storage when the train command is called. The latest stored model is loaded from storage when the test or predict command is called. When the user passes the visualization flag, analysis results are shown.

4.2. Implementation View

The component diagram in Figure 3 shows a layered design following the 5C architecture for cyber-physical systems (Lee, Bagheri, & Kao, 2015). Note that our architecture has not implemented components for the *Configuration* layer, as we do not handle asset maintenance optimization in this case

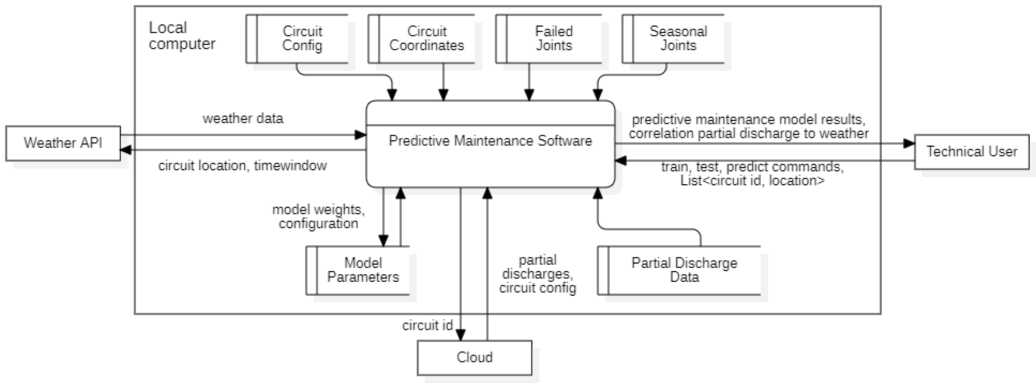
Local computer
Circuit Config
Circuit Coordinates
Failed Joints
Seasonal Joints
Weather API
weather data
circuit location, timewindow
Predictive Maintenance Software
predictive maintenance model results, correlation partial discharge to weather
Technical User
train, test, predict commands, List<circuit id, location>
model weights, configuration
Model Parameters
partial discharges, circuit config
Partial Discharge Data
circuit id
Cloud

Figure 1. Context diagram of the predictive maintenance software

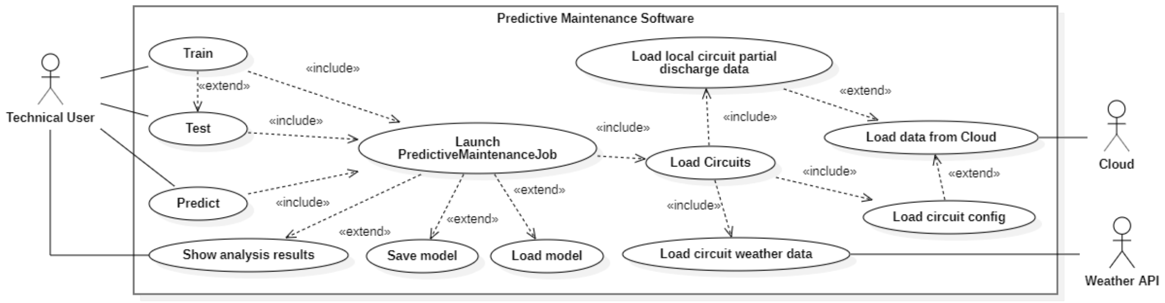

Figure 2. Use case diagram of the predictive maintenance software

study. The PdM software communicates with two external entities for loading (1) weather data and (2) partial discharge and circuit configs. Data loaders are designed using interfaces to ensure flexible conversion when another data type is delivered. The *PredictiveMaintenanceJob* component is responsible for interpreting the user commands, calling the appropriate backend components, and returning the results of those components to the user. The *Visualizer* component is responsible for printing the results to the command line in a human-readable format and plotting various graphs. The *SimulationModel* represents the physical object from various viewpoints, for example, a *PartialDischargeForecaster*. The *PartialDischargeCorrelator* is responsible for finding weather features that correlate with partial discharges to provide more insight into the joint's behavior (e.g., correlation with wind might highlight the influence from a wind farm). The *PredictiveMaintenance* component is responsible for models like anomaly detection or RUL estimation. The *FailedJointsReader* and *JointsWithPartialDischargeReader* components are responsible for loading corresponding data files. The *CircuitWeatherRetriever* is responsible for requesting weather data through Alliander's weather API using coordinates as input, generally the coordinate of the primary partial discharge sensor. The *S3MergedCircuitStorage* is responsible for loading circuit partial discharge data and configs from local storage or via *scg_analytics*. The *Readers.Circuit* component is responsible for loading circuit-specific data and building objects from the *DataTypes* component. The *DataTypes* component is responsible for objects that are generated by the *Readers.Circuit* component or ingested by the *PredictiveMaintenance* and *SimulationModel* components. *Readers.Abstraction* is responsible for providing abstract classes or shared interfaces.

4.3. Behavioral View

The activity diagram of the PdM software is shown in Figure 4. It elaborates on the behavior of the software. First, the config file is loaded. The config file contains the absolute paths of the data files. The data reader objects are created and load common data. After loading the data, the *CircuitFactory* object is created. Based on the train, test, and predict arguments, circuit-specific data is loaded. We use the given data for the train argument and calculate correlations per set. The *Visualizer* activity diagram contains the if statement for visualizing the model results. Afterward, the *PartialDischargeForecaster* and *RemainingUsefulLifeEstimator* models are trained, their parameters are saved, and the models are tested. The results of each step are visualized if the flag is given in the argu-

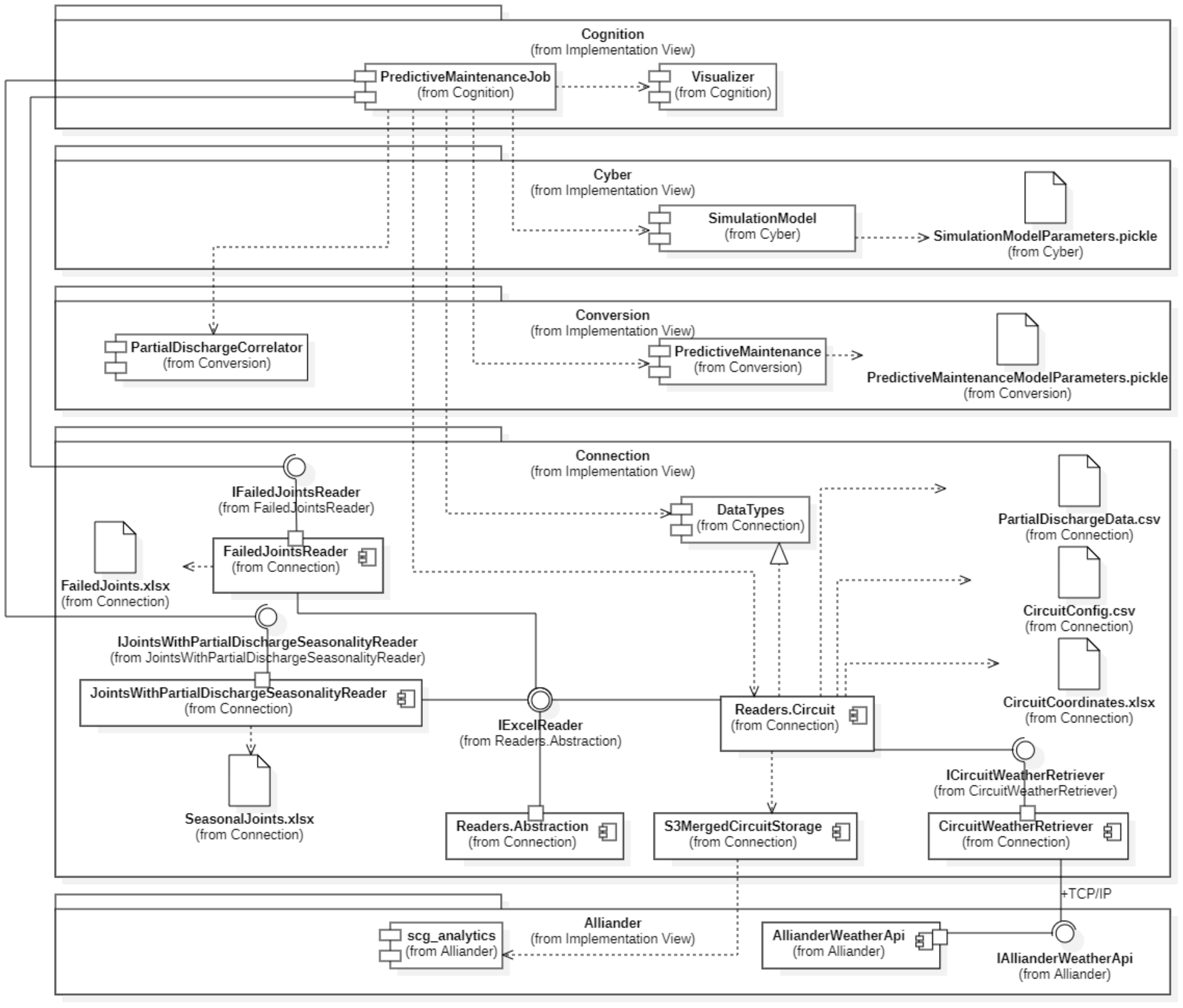

Figure 3. Component diagram of the predictive maintenance software architecture

ments. For the test and predict arguments, the saved model parameters are loaded. Then, the model uses a test set or a list of given joint locations to predict.

4.4. Environmental View

The data files on the local disk are shown in 5. The diagram shows two subfolders: *ModelParameters* and *Data*. The *Data* folder resides all data files that Alliander's systems have provided. The partial discharge and circuit config CSV files are separate files for each loaded circuit. Currently, we expect to use less than five hundred circuits for modeling, so storing the data on disk is possible to reduce data transfer compared to always loading circuit data from the cloud. The *ModelParameters* folder contains a subfolder for each model type with a pickle file of the model weights to reload the model. The deployment diagram in 6 shows the external entities that interact with the PdM software. The *CircuitWeatherRetriever* communicates with Alliander's Weather API. The *S3MergedCircuitStorage* uses an implementation by Alliander's private software, which communicates with their private AWS S3 data storage. Both components communicate with the interfaces using TCP/IP.

5. Conclusion and Discussion

With the Model-based Systems Engineering (MBSE) approach and a previously designed Reference Architecture (van Dinter, Tekinerdogan, & Catal, 2023), we provided several architectural viewpoints for the different perspectives of

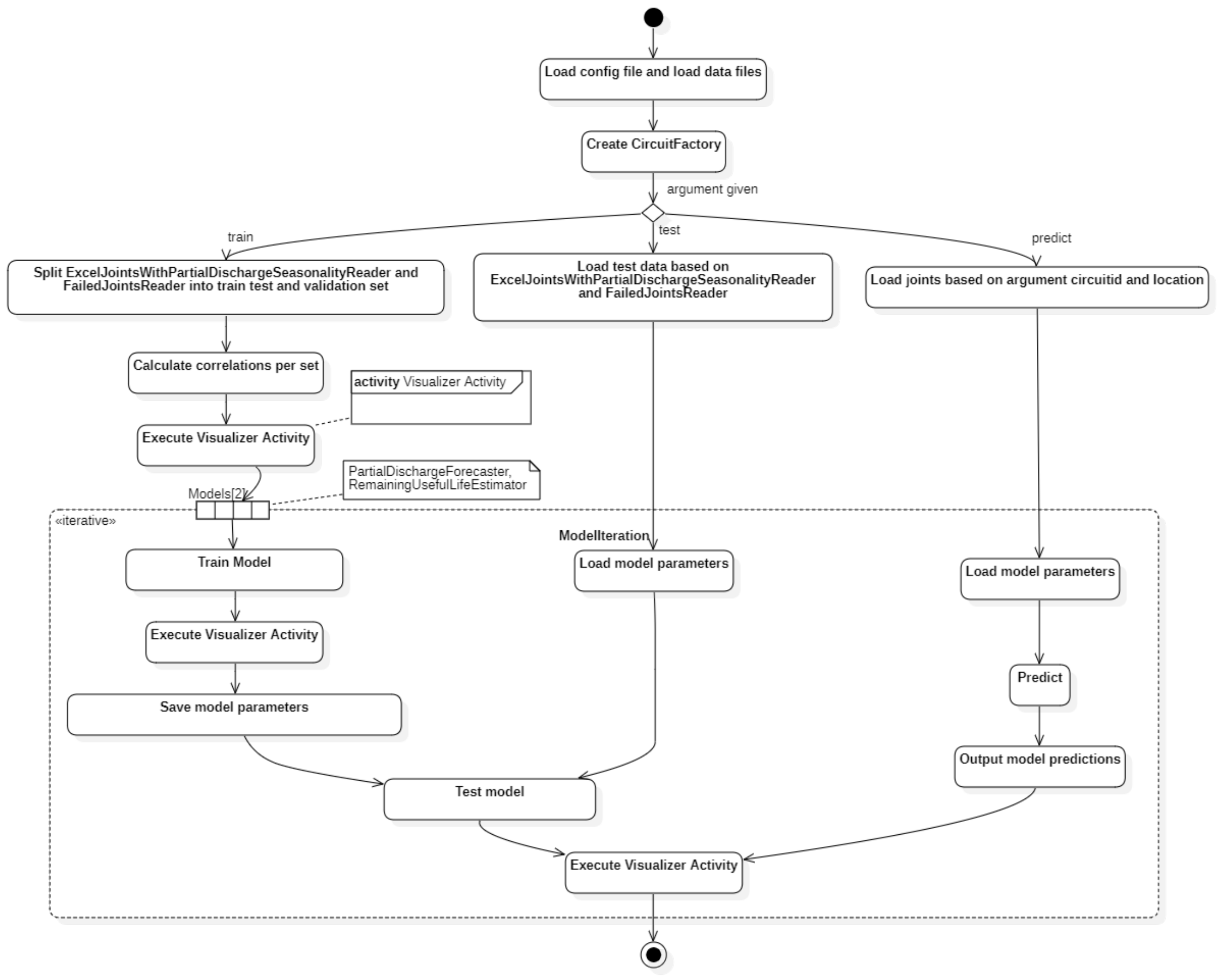
Load config file and load data files
Create CircuitFactory
argument given
train
test
predict
Split ExcelJointsWithPartialDischargeSeasonalityReader and FailedJointsReader into train test and validation set
Load test data based on ExcelJointsWithPartialDischargeSeasonalityReader and FailedJointsReader
Load joints based on argument circuitid and location
Calculate correlations per set
activity Visualizer Activity
Execute Visualizer Activity
PartialDischargeForecaster, RemainingUsefulLifeEstimator
Models[2]
«iterative»
ModelIteration
Train Model
Load model parameters
Load model parameters
Execute Visualizer Activity
Predict
Save model parameters
Output model predictions
Test model
Execute Visualizer Activity

Figure 4. Activity diagram of predictive maintenance software

the stakeholders. We leveraged knowledge from a systematic literature review (van Dinter, Tekinerdogan, & Catal, 2022) to gather required system components. We could model the ”system-of-systems” nature of modeling cable joint degradation in the medium-voltage grid. With the design ontology, we developed a First-Time-Right software solution for monitoring the degradation of cable joints. The design is robust to changes in data formats, as we used interfaces for data input. Changes in data formats or Application Programming Interfaces (APIs) can be quickly introduced with a new interface implementation. A threat to the validity of this work is that we focused on one public utility company in the Netherlands. Our study might not apply to other public utility companies in- our outside the Netherlands.

This paper aims to show the Model-based Systems Engineering (MBSE) design process and to develop a Digital Twin-based Predictive Maintenance system for modeling cable joint degradation. We have designed a scalable, robust, and flexible software stack with the proposed architecture to monitor cable joint degradation. The software implementation of the design is publicly available in an open-source repository. The MBSE approach and the use of UML models allowed for the comprehensive and structured development of the Digital Twin-based Predictive Maintenance system, providing a clear understanding of the system’s structure and behavior. The resulting UML models and software design can be used to guide future development and maintenance of the system. This use case can be used to present the open-source software and as a guide for other Digital Twin-based Predictive Maintenance case studies.

For future work, we will focus on (1) evaluating the performance of the architecture in different case studies globally,

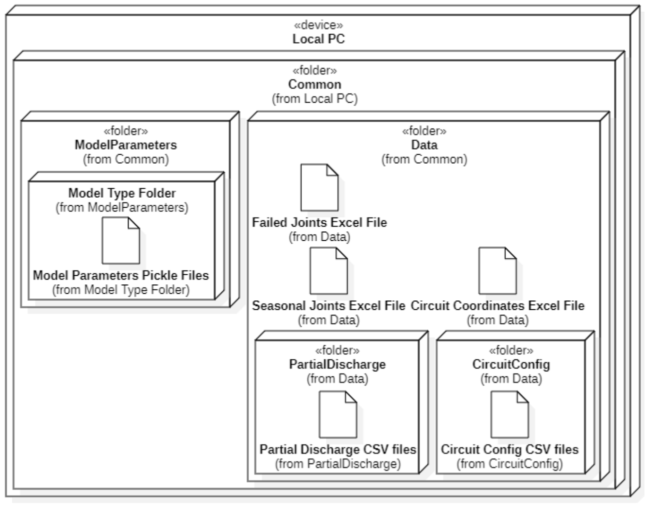

Figure 5. Deployment diagram of data files on local disk

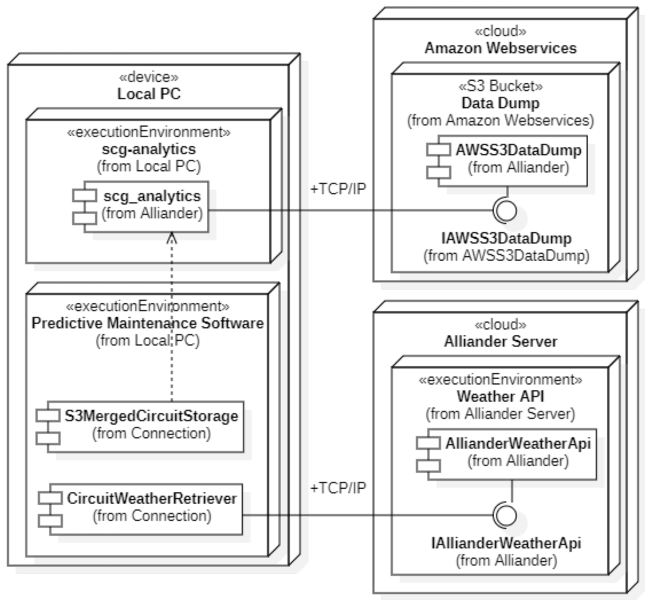

Figure 6. Deployment diagram of interaction with external entities

(2) deploying the software, (3) adding components in the Configuration layer, (4) using other available data, such as joint age, manufacturer, connecting cable types, soil type, or load, (5) assessing the design's performance in practice and (6) optimizing the predictive maintenance models.

REFERENCES

Aizpurua, J. I., & Catterson, V. M. (2015). Towards a methodology for design of prognostic systems [Conference Proceedings]. In *Annual conference of the phm society* (Vol. 7).

Aizpurua, J. I., & Catterson, V. M. (2016). Adeps: a methodology for designing prognostic applications [Conference Proceedings]. In *Phm society european conference* (Vol. 3).

Alliander. (n.d.). *Weather api* (Vol. 2023) (Web Page No. 7 March). Retrieved from `https://weather.appx.cloud/api/v2/docs#`

Cocheteux, P., Voisin, A., Levrat, E., & Iung, B. (2009). Prognostic design: requirements and tools. In *11th international conference on the modern information technology in the innovation processes of the industrial enterprises,, mitip 2009* (p. CDROM).

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, *12*(6), 42–50.

Lee, J., Bagheri, B., & Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems [Journal Article]. *Manufacturing letters*, *3*, 18-23.

Li, R., Verhagen, W. J., & Curran, R. (2018). A functional architecture of prognostics and health management using a systems engineering approach [Conference Proceedings]. In *Proc. eur. conf. phm soc* (p. 1-10).

Mall, R. (2018). *Fundamentals of software engineering* [Book]. PHI Learning Pvt. Ltd.

Tiddens, W. W. (2018). Setting sail towards predictive maintenance: developing tools to conquer difficulties in the implementation of maintenance analytics [Journal Article].

van Dinter, R., Ekmekci, G., Netten, G., Rieken, S., Tekinderdogan, B., & Catal, C. (2023, April). *A code repository for predictive maintenance on cable joints.* doi: 10.5281/zenodo.7863853

van Dinter, R., Tekinerdogan, B., & Catal, C. (2022). Predictive maintenance using digital twins: A systematic literature review. *Information and Software Technology*, 107008.

van Dinter, R., Tekinerdogan, B., & Catal, C. (2023). Reference architecture for digital twin-based predictive maintenance systems [Journal Article]. *Computers Industrial Engineering*, *177*, 109099.

van Osch, M. (2021). *Analyzing data of the medium voltage grid and the weather to predict power outages* (Thesis, Radboud University). Retrieved from `https://www.math.ru.nl/~bosma/Students/MeesvanOschMSc.pdf`

Vogl, G. W., Weiss, B. A., & Donmez, M. A. (2014). *Standards for prognostics and health management (phm) techniques within manufacturing operations* (Report). National Institute of Standards and Technology Gaithersburg United States.

Wagenaars, P. (2010). *Integration of online partial discharge monitoring and defect location in medium-voltage cable networks* (Doctoral dissertation, Technische Universiteit Eindhoven). doi: 10.6100/IR656994