Predictive maintenance in a fleet management system: the Navarchos case*

Apostolos Giannoulidis[1], Anna-Valentini Michailidou[2], Theodoros Toliopoulos[1,2], Ioannis Constantinou[2], and Anastasios Gounaris[1]

[1] Aristotle University of Thessaloniki, Thessaloniki, Greece
{agiannous,tatoliop,gounaria}@csd.auth.gr
[2] Istognosis Ltd., Nicosia, Cyprus
valentina@navarchos.com,ioannis@istognosis.com

Abstract. This work presents a state-of-the-art predictive maintenance (PdM) framework, which is tailored to demanding cases, where information is dynamic and partial, and non-supervised solutions should be applied. Moreover, it discusses and aims to demonstrate the application of this framework to the Navarchos Fleet Management System (FMS).

1 Introduction

Fleet Management Systems (FMS) play a pivotal role in maintaining the operational safety and efficiency of vehicle fleets, especially in the face of escalating fleet sizes. A critical responsibility within this domain is the meticulous scheduling of vehicle maintenance, aimed at averting potential failures that not only endanger driver safety but also disrupt vehicle uptime. As we navigate the landscape of Industry 4.0, the integration of Predictive Maintenance (PdM) emerges as a crucial paradigm shift for enhancing FMS capabilities.

Traditionally, many FMS rely on Diagnostic Trouble Codes (DTCs) sourced from original equipment manufacturers (OEMs) as a primary means of identifying and addressing vehicle issues. These codes, emanating from the engine control unit (ECU), serve to pinpoint faulty behavior based on predefined rules applied to sensor data. However, a main limitations of this approach is that serious vehicle failures evade detection through DTCs, highlighting a need for a more proactive and comprehensive solution. Furthermore, while DTCs excel at detecting malfunctions, their primary focus contrasts with the broader scope of PdM, which not only identifies existing issues but anticipates potential failures.

In a FMS, PdM constitutes a holistic end-to-end application encompassing data collection, pre-processing, storage, and the generation of predictive alarms within a dynamic streaming environment. Implementing such a solution comprises many challenges, including the dynamic nature of vehicle operations, insufficient information about past services and repairs, and the absence of readily

* The research is funded under the programme of social cohesion "THALIA 2021-2027" co-funded by the European Union, through Research and Innovation Foundation.

available expert guidance. The intricacies of these challenges underscore the complexity involved in establishing a robust PdM framework for FMSs. In this work, we delve into the application of PdM in the FMS of Navarchos proposed in [2], presenting detailed information and insights gained from real-world implementation. While frameworks like COSMO, as presented in [5], exist for PdM in vehicles, we distinguish ourselves in several ways. Firstly, our fleet is heterogeneous, comprising various vehicle types operating in diverse areas, both urban and non-urban. Additionally, we address the challenge of partial information concerning the state of vehicles, derived solely from our limited access to a subset of the vehicles' service logs. Overall, we aim to contribute valuable perspectives to the evolving landscape of working PdM systems in fleet management.

Structure. The next section provides the architecture of Navarchos FMS. Sec. 3 deals with the PdM framework design principles, while Sec. 4 discusses its implementation. We conclude with the demo description.

2 Navarchos Architecture Overview

The NAVARCHOS AI Fleet Management System (FMS) has been developed as a cloud-native solution, deployed on a Kubernetes infrastructure directly on bare metal. This system adopts a microservices architecture, where each Internet of Things (IoT) subsystem is a distinct, independently deployable service. This architecture enables autonomous scaling of each subsystem, significantly enhancing the system's overall scalability, simplifying updates to individual components, and allowing for the selection of the most appropriate technology for each subsystem based on its unique requirements.

Each subsystem, as well as the FMS as a whole, is continuously monitored to quickly identify and rectify failures, ensuring swift service recovery. The majority of subsystems communicate over REST/HTTPS with JSON payloads; JSON is chosen due to its human-readable format. Nevertheless, for critical performance subsystems like the GPS tracking gateway, which utilize inherently binary protocols, a binary communication protocol has been implemented to assure high performance.

Subsystems are deployed in isolated environments, such as containers, taking full advantage of Kubernetes' dynamic workload management and advanced scalability features. This configuration permits the system to dynamically adjust resources (either scaling up or down) in near real-time in response to evolving business needs. Additionally, the system embraces Continuous Integration (CI) and Continuous Delivery (CD) practices, enabling frequent updates and maintaining the system's currency with minimal downtime. The NAVARCHOS FMS architecture has been meticulously developed, showcasing a comprehensive, multi-layered approach to fleet management. This system seamlessly integrates a broad array of functionalities, specifically designed to meet the advanced needs of modern vehicle tracking, data processing, and analysis. Below is a detailed overview of the developed subsystems, also presented in Figure 1:

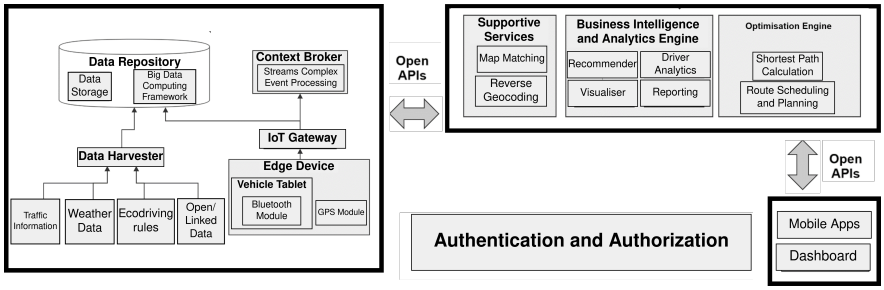

Fig. 1. NAVARCHOS FMS architecture

Edge Devices: Installed across the fleet, these devices capture GPS locations and vehicle diagnostics, ensuring secure data exchange with the cloud.

Vehicle Data Gateway/Hub: Acts as a secure conduit for GPS locations and vehicle diagnostics data, compatible with a diverse range of vehicle tracking protocols.

Data Harvester: This subsystem interfaces with various external data sources (e.g., via APIs) to aggregate, process, and store both real-time and historical data, such as weather and traffic conditions, within the NAVARCHOS FMS infrastructure. This capability is vital for generating timely notifications, alerts, and underpinning business intelligence and analytics.

Context Broker: Serves as the intermediary for sharing context information between input subsystems (such as the data gateway and data aggregator, known as context producers) and other subsystems or users (context consumers) who require this information. It accommodates dynamic roles, allowing entities to act as either producers, consumers, or both, and manages updates as subscribable events.

Stream Processor: Incorporated within the context broker or as an independent subsystem, it processes data streams and integrates them with business processes, enabling real-time responses to recognized events with notifications and alerts.

Routing Optimizer Microservice: Processes a sequence of GPS locations to determine the most efficient route, providing detailed turn-by-turn navigation instructions to optimize travel paths.

Scheduling Optimizer Microservice: Solves the Vehicle Routing Problem (VRP) by calculating the optimal delivery routes for a fleet of vehicles, considering various constraints and objectives, and then providing detailed navigation guidelines.

Map-Matching Microservice: Matches recorded geographic coordinates to a logical representation of the real world, minimizing storage needs for trip data, facilitating quicker trip visualization, and standardizing geodata for subsequent analysis.

Reverse Geocoding Microservice: Transforms geographic coordinates back into readable addresses or place names, simplifying the visualization and understanding of trip data for users.

Authentication/Authorization Gateway (AAG): Oversees user authentication, determining access privileges to the NAVARCHOS FMS and identifying which microservices each user is authorized to use.

Business Intelligence and Analytics Machine: Delivers a holistic view of collected data, enabling informed decision-making and promoting operational efficiency and cost savings for fleet-based operations.

Data Storage: Guarantees data availability and resilience, featuring automatic failover and maintenance operations without causing system downtime.

Visualizer: Provides visualization tools for data, analytics, and insights, especially regarding driver behavior, and supports the management of vehicles, devices, and user interfaces.

REST API: Makes system data accessible and supports the integration of NAVARCHOS FMS with other applications, thereby enhancing both interoperability and extensibility.

3 Our PdM Solution

Navarchos's fleet comprises a variety of vehicles operating in different conditions in terms of driving behavior, route types, weather conditions and so on. The objective of PdM is this context is to detect which vehicles should perform maintenance tasks (i.e., unscheduled, non-periodic service) to avoid serious damage, based on their operational state derived solely from sensor data and (partial) past maintenance event recordings. The absence of appropriate and/or adequate labels and historical data led us to a non-supervised solution upon Parameter IDs (PIDs) signals [3]; DTCs (Diagnostic Trouble Codes)[4] are also monitored, but, after investigation, they cannot serve our purpose. The main rationale is to detect deviations from the normal operating condition in vehicles; such deviations are interpreted as early signs of a forthcoming damage. In conclusion, the primary challenge lies in the requirement to offer precise indications for maintenance requirements while safeguarding the trust of drivers and mechanics. This underscores the need for enhanced precision in our anomaly score-based alarms. Such alerts serve to notify both drivers and maintenance managers of abnormal vehicle operation, prompting the necessary inspections.

To address all the aforementioned challenges, we build a PdM framework, which does not rely on domain expert active involvement, handles the dynamicity of the operating conditions, and maintains high precision in predicting failures. This framework consists of three main modules forming a 3-stage pipeline: 1) transformation of data in a form that highlights behavioral changes, 2) the construction of a normal reference state of a vehicle, and 3) use of a non-supervised model to produce an anomaly score. Full details and evaluation results are in [2], and here we present only a summary.

The first module's objective is to transform the data into a form where changes related to the failure state of a vehicle are highlighted. The final choice

[3] https://en.wikipedia.org/wiki/OBD-II PIDs
[4] https://github.com/mytrile/obd-trouble-codes/blob/master/obd-trouble-codes.csv

of these steps depends on the variety and volume of the data that our system collects. Key alternatives include delta transformation, correlation between signals, frequency-domain transformation, histograms, and so on.

The second module deals with constructing a reference state of vehicle's normal operating condition and is closely related to the support of non-supervised solutions. The idea behind this stage is that, based on the normal operation state of a vehicle despite the existence of noise, we can calculate a deviation of that vehicle in real-time by comparing its current state to the reference one. The extraction of such knowledge (i.e., the normal reference profile), depends on the actual case and characteristics of the problem. In most PdM cases, we expect that, in general, a failure state is rarer than healthy/normal operation. Different approaches exist in the literature regarding this step, for example, in [3], the authors selected the data of an asset in its starting state as healthy, while in [1], the authors based the selection of healthy data on the concept of the wisdom of the crowd, i.e., in any given time point, the majority of a bus fleet performing the same routes is in a good condition.

Finally, the third module concerns the actual detection of abnormal operational behavior and thus the need for maintenance. Here, a non-supervised model that leverages the transformed data and the knowledge regarding the reference data produces a deviation level (a.k.a, an anomaly score) and alarms for each vehicle in the fleet in a streaming fashion. The choice for such a model can vary, where any non-supervised model is applicable without excluding supervised ones if enough trustworthy data have been gathered. Navarchos currently operates in a non-supervised manner and three well-known representative types of techniques employed are similarity-based, reconstruction (deep-learning) models, and regression (or forecasting) ones. We note that the performance of a model depends on many factors, such as the size of the reference data, the underlying hypothesis used by the model to predict anomalies (e.g., proximity-based techniques use distance, statistic-based techniques use some kind of distribution, where many different approaches exist in literature).

Framework Instantiation in our FMS. We provide more details about how the framework above is instantiated within Navarchos FMS to become operational and effective. As previously, more details are in [2]. Regarding the first stage, the correlation between the raw features is selected, which lets us observe behavioral changes between two different periods of vehicle operation. Correlation transformation refers to the calculation of the cross-correlation between the different data features. Using a sliding window over raw data, we calculate the correlation between the collected signal data. In more detail, if we consider that the number of the initial data features are f_n, after cross-correlation, we result in a symmetric $f_n \times f_n$ matrix, which can be considered as a vector with $\frac{f_n * (f_n - 1)}{2}$ features. The idea behind this transformation is that different usages of vehicles may produce similar correlations. For example, we expect that speed and rpm are positively correlated regardless of whether the vehicle performs urban or regional rides under different weather conditions, while a difference in the correlation of two signals may refer to a failure state.

For the second stage, to define a reference state, we use a period of vehicle operation after maintenance (or standard service), assuming that the vehicle operates normally after such events without seeking more guarantees. Note that building the reference state based on such events yields a dynamic solution, where the reference vehicle profile is updated upon known completion of maintenance tasks. Moreover, towards being more flexible, the framework allows the recollection of the reference data upon request, where data from the current period can be used for reference.

To instantiate the final stage of our framework, we use the *Closest Pair Detection* technique, inspired by solutions such as those in [4, 3]. The technique calculates the deviation of upcoming data by leveraging a healthy reference. However, instead of producing a single score for each vehicle, the technique calculates the distance of upcoming data from the reference data, in each feature separately. So, if the dimensionality of our data, or the number of features is f_n, then the technique produces f_n anomaly scores, by computing the minimum difference between the value of its feature from the values of the same feature in the reference data. Alarms are produced from violation of the threshold in any of f_n anomaly scores and are accompanied by a description with the feature that triggered it.

The above methodology relies on appropriate tuning. For setting the threshold for each score, we decided to use the self-tuning thresholding from [3], which is useful in dynamic environments and, using the same parameters, a different threshold is calculated for each vehicle.

4 Navarchos PdM Service Implementation

In this section, we present the pipeline and data flow through the system. The process begins with data collection from GPS tracking devices installed on vehicles. These data undergo parsing and transformation to become suitable input for the PdM module. This module then processes the input to generate predictions and trigger alarms when necessary. Figure 2 provides an overview of the pipeline, presenting the technologies employed at each stage and the type of data being transmitted. In the following sections, we will detail each step of this process, starting with data parsing, followed by data processing, and concluding with an overview of the PdM module.

Data Parsing. Vehicles are equipped with edge devices (GPS tracking devices) that monitor and report the status of various vehicle sub-systems in real-time. Each tracking device utilizes a specific communication protocol to periodically exchange data with a server in the form of data packets. These packets contain information such as device ID, timestamp, location, speed, temperature, and so on. The tracking device and communication protocol used in this work is manufactured by Sinocastel[5]. The transmitted packets are represented in a hexadecimal format and need to be decoded to extract the information. This

[5] https://www.sinocastel.com/

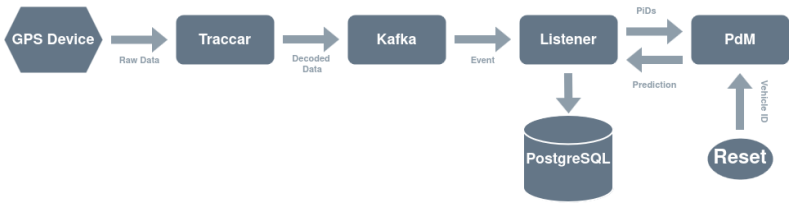

Fig. 2. Data flow of NAVARCHOS PdM

decoding is performed by Traccar[6], an open-source GPS tracking server that supports multiple models of GPS tracking devices and protocols, by providing real-time tracking in fleet management scenarios. Upon receiving a data packet from a tracking device, Traccar identifies the protocol used, decodes the incoming data packet, and extracts information such as device ID, timestamp, location, and several PID signals. These data are then transformed into a human-readable format, specifically to JSON, and are then sent to an Apache Kafka topic for further analysis.

Data Processing. To continuously consume vehicle data from the Kafka topic to which Traccar sends data, a Spring application was developed. More specifically, a Kafka listener that consumes sensor data related to pressure, speed, and temperature of the vehicle engine from this topic was implemented. When a new message arrives, the listener receives it and stores the important information in a database table; the database we utilized is PostgreSQL with provisions for the cases where data volumes grow very large. If the message contains all six of the required PID features without any outliers, based on specified thresholds, it is also stored in a buffer table. This buffer is used to implement the first stage of the PdM framework (Section 3), which is the data correlation transformation. More specifically, the Pearson correlation coefficient between the PID features is calculated using the last m (m=300) stored signals. This calculation of the correlation between features takes place every 100 new samples, and the results are submitted along with the timestamp and vehicle ID to the PdM Module through an API. In return, the PdM Module for each submitted message returns a response with the anomaly scores for all features and corresponding thresholds, an indication for alarms, and a description.

PdM Module. The PdM module essentially implements the second and third stages of the proposed PdM framework. It is based on a simple communication scheme, where two functionalities are implemented, namely 1) data sample collection, and 2) event collection. In both cases, data and events arrive with two tag values, a vehicle ID, which indicates the vehicle to which the data or the event is related, and a timestamp. Examples of events are services, DTCs, and user-defined events. The data sample represents a record with the available features. When a new event or data sample from a vehicle (or source) arrives, the system checks if corresponding models exist for that vehicle; if not, it creates

[6] https://www.traccar.org/

one. After that, in the case of data samples, the corresponding model before calculating anomaly scores, checks if reference data exist for that source. When no reference data are available, the PdM module takes care of creating the reference set from the upcoming data; this will stop when criteria regarding the size of the reference set are met. When a reference set exists, the method produces the anomaly scores for all features and returns the decision (along with scores and thresholds). Finally, the events are used to trigger the reset of the models i(i.e., the recalculation of reference data with the new upcoming data) on the fly. To this end, the models are saved on a separate database, so that, in case of system failure, to start from their last state.

5 Demo description and Conclusions

The demo will showcase a successful scenario of detecting failures in operating a vehicle using NAVARCHOS FMS bringing all the aforementioned functionalities together. The scenario involves a stream of raw data produced by a specific vehicle, emphasizing crucial stages of the data flow, encompassing the computation of the anomaly score in a streaming fashion, illustrating how this aids in mitigating upcoming failures. The demo commences with the presentation of upcoming vehicular raw data while the vehicle is in operation. Subsequently, the next stage of the data flow reveals the transformation of raw data into a readable format (JSON), which also encapsulates the signals used for failure prediction. Moving forward, the results of data collection in batches and the extraction of correlation features are displayed. Lastly, in the concluding part of the scenario, the anomaly score is computed and presented for each feature as described in Section 3. The analytical results, encompassing raw anomaly scores and a user-friendly indicator for the vehicle, will be visible as part of NAVARCHOS FMS.

Overall, we present a state-of-the-art PdM framework and an exemplary instantiation of it to serve the needs of a real-world FMS.

References

1. Fan, Y., Nowaczyk, S., Rögnvaldsson, T.: Evaluation of self-organized approach for predicting compressor faults in a city bus fleet. INNS Conference on Big Data pp. 447–456 (2015). https://doi.org/https://doi.org/10.1016/j.procs.2015.07.322
2. Giannoulidis, A., Gounaris, A., Constantinou, I.: Exploring unsupervised anomaly detection for vehicle predictive maintenance with partial information. In: EDBT. pp. 753–761
3. Giannoulidis, A., Gounaris, A., Nikolaidis, N., Naskos, A., Caljouw, D.: Investigating thresholding techniques in a real predictive maintenance scenario. SIGKDD Explor. Newsl. **24**(2), 86–95 (dec 2022), https://doi.org/10.1145/3575637.3575651
4. Linardi, M., Zhu, Y., Palpanas, T., Keogh, E.J.: Matrix profile goes MAD: variable-length motif and discord discovery in data series. Data Min. Knowl. Discov. **34**(4), 1022–1071 (2020)
5. Rögnvaldsson, T., Nowaczyk, S., Byttner, S., Prytz, R., Svensson, M.: Self-monitoring for maintenance of vehicle fleets. Data Mining and Knowledge Discovery **32**(2), 344–384 (2018). https://doi.org/10.1007/s10618-017-0538-6