POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

EXECUTIVE SUMMARY OF THE THESIS

Design and Implementation of a Decision Support System for Manufacturing Maintenance Optimization

TESI MAGISTRALE IN COMPUTER SCIENCE AND ENGINEERING – INGEGNERIA INFORMATICA

AUTHOR: ROBERTO SANTANGELO

ADVISOR: MARCO BRAMBILLA

ACADEMIC YEAR: 2020-2021

1. Introduction

The aim of the thesis is the development of a Decision Support System which could improve the conditions of maintenance operation inside the production plant. It wants to highlight the that by taking advantage of the historical data and those produced by the machines it is possible to shift from a responsive approach to maintenance towards a preemptive one [1]. It also aims to create a tool which would help the Maintenance Manager to take decisions based on a set of information gathered by the plant, adding a degree of objective analysis to a procedure which is usually experienced based.

2. Theoretical Background

The core of the development system is Decision Support, so it has the task of providing useful insights during the Decision-Making process. In maintenance Decision Making consists in establishing the set of possible actions to perform in order to better tackle the raised problem. It faces the challenge of tackling a very complex environment which produces a wide set of information from different sources [2]:

- The Current health condition of each machine including down, running, idling, being maintained or just about to breakdown;
- The scheduled daily, weekly, and monthly maintenance plan;
- Machine health degradation profile;
- Etc;

Naturally this brings the need analysis of bug and variegated sets of data. Big Data are becoming more and more relevant in basically any field, they are definable as a large amount of data with a very stochastic structure. They are characterized by [3]:

- *Volume;*
- *Variety;*
- *Velocity;*

They are also pushed by the principles of IoT and industry4.0, which take advantage of modern and interconnected equipment which produce information through the sensors monitoring their status, and by exchanging information between themselves [4]. The combination of all these techniques allows a shift to a prescriptive way of performing maintenance.

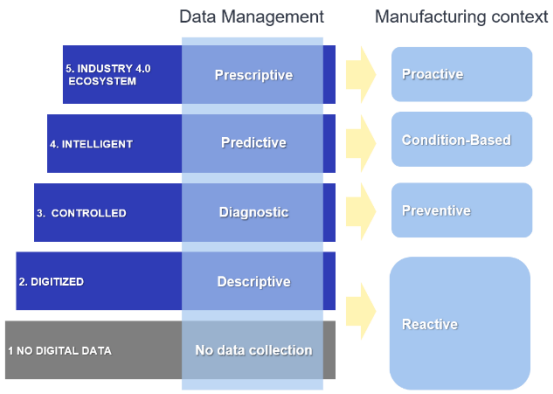


Figure 1: Proactive Decision Context

Decisions are taken directly recommending the best actions to be performed in the context of interest. These actions are decided through the usage of operative research algorithms which dynamically elaborate the best possible solutions for each problem, depending on different factors and then they maximize the gains or minimize the drawbacks. It takes advantage of Predictive maintenance to anticipate possible breakdowns, in fact, The aim of Predictive Maintenance is the assessment of the health status of key components, regardless of their maintenance status, in order to make predictability achievable [5].

3. Decision Support Module Design

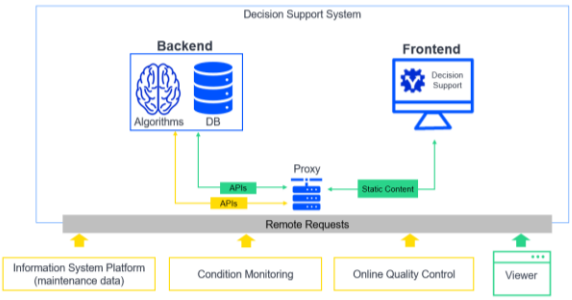


Figure 2: Decision Support Module Architecture

The DSM has been designed with modularity in mind, so it is divided in different independent blocks. The system in Figure 15 is composed by four main blocks which are responsible for different duties inside the DSM:

- Frontend: represents the interface between the user and the system;
- Backend: represents all the system components dedicated to the Decision Support Algorithms and data storage;
- Proxy: it acts as a router for the data between Frontend, Backend, and the Remote Requests;
- Remote Requests: they represent all the possible requests which can come from outside the DSM, they can be either: Data sources or Frontend users. Data sources are represented in yellow and communicate with the backend, Frontend users are represented in green. The following external sources are identified:
  - Information System Platform: which gathers all the general information about the plant;
  - Condition Monitoring: which consists in a predictive maintenance algorithm;
  - Online Quality Control: which gathers information from a Deep Learning based vision system;
  - Viewer: which requires the visualization of the frontend and interacts with its functionalities;

This architecture has been defined by interviewing different figures inside the refenced plant of Novameta, from the Maintenance Manager to the operators, and from researching about decision support looking at similar examples and existing systems. This allowed to understand the needs of the project and resulted in a set of requirements:

DEC.SCH.001

Type: It describes requirement related to the scheduling of periodic works.

Description: The system must be able to represent the suggestions in the best and clearest way possible. Through different views it will allow the Maintenance Manager to organize the execution of the needed maintenance operations. It is important for the Decision Support System to consider in a proper way the maintenance operation timing, identified in paragraph :

- Recurring tasks
  - Daily;
  - Weekly, Bi-weekly, Monthly, Yearly;
- One-time tasks;

DEC.SCH.002

Type: It represents the constrains related to the assignment and scheduling of the works

Description: The Decision Support System must consider in the proper way the tasks characteristics.

In particular, the Decision Support System needs to accurately consider the task assignment:

- *Not assignable*: are all the tasks that do not require user assignment;
- *Assignable*: are all those tasks which needs to be assigned to a worker with a specific qualification;
- *Assigned*: are all those tasks which have a worker already assigned by the Maintenance Manager;

DEC.OP.01
Type: It represents the decision support information that the Maintenance Manger expects to see about the operator.
Description: The Maintenance Manager must receive from the Decision Support System the following information about the operator:

- *General info*: contains all the basic information about the operator;
- *Schedule*: shows the tasks that the operator must perform during its shift on a weekly and daily basis, following the same principles of the whole schedule view;
- *Status of current operation*: shows an indication of the progression of the task;

DEC.MCH.01
Type: It represents the decision support information that the Maintenance Manger expects to see about the machines.
Description: The Maintenance Manager must receive from the Decision Support System the following information about the equipment:

- *Status*: The status represents what is happening to the machine
- *Schedule*: the maintenance activities that should be performed on the machine, when and in which conditions;
- *KPI*: additional indicators about the machine, e.g., the machine availability;

DEC.WH.01
Type: It represents the decision support information that the Maintenance Manger expects to see about the management of the warehouse
Description: To correctly manage the warehouse, the Maintenance Manager must monitor the following information about spare parts and tools:

- *Indication of quantity*: the amount of common spare parts and tools available in the warehouse;
- *Minimum allowed capacity*: the threshold that triggers the reordering of material;
- *Restock suggestion*: a suggested date in which the Maintenance Manager should perform the reordering of the parts and tools, and the needed quantity;
- *KPI*: indicators about the overall warehouse status, the general availability of spare parts and tools and the optimization of their usage;

4. Decision Support Module Implementation

The implemented system is based on the architecture in Figure 3.

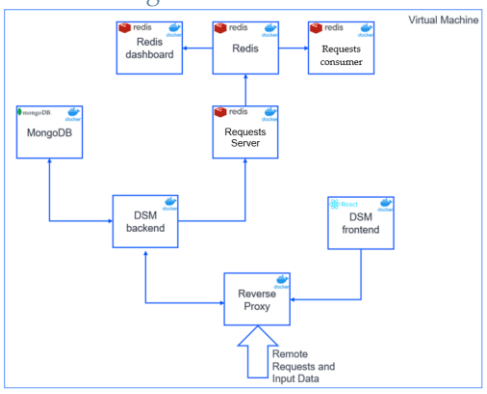


Figure 3: Detailed DSM architecture

The DSM is based on Docker, and it is divided into different containers, one for each component, as follows:

- Reverse Proxy: acts a router for incoming requests;
- DSM Frontend: encapsulates the frontend application;
- DSM Backend: implements the APIs and manages the requests between the Mongo and the Requests Sever;
- MongoDB: encapsulates the MongoDB;
- Redis: encapsulates Redis and all its services;
- Redis Dashboard: allows to see and monitor the status of the queues and the jobs;
- Requests consumer: encapsulates the algorithms made for the computation of the suggestions;
- Requests Server: exposes a private HTTP server to accept the computation requests;

Docker allows to instantiate the DSM onto any system which is capable of running it, making the DSM itself hardware and OS independent.

Modularity is another characteristic which made Docker the better solution for the needs of this project, it allows to quickly update containers without any particular restriction. The DSM expects to be triggered by an external source, then the request is put in a queue managed by the requests server waiting to be executed and consumed by the consumer which triggers the algorithms.

4.1. Frontend

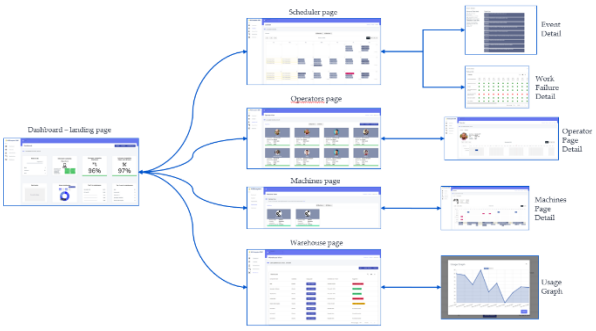


Figure 4: Frontend view navigation

The DSM frontend has been designed to be fast to use and easy to navigate. Once the application is opened the user is presented with the dashboard component, which displays the main KPI and information, giving immediate feedback on the main possible issues present in the plant (not completed works, operators' availability etc.). from the dashboard is possible to navigate to four different views:

- *Scheduler:* which displays the generated schedule and allows to interact with the events and the timeframe to be displayed.
- *Operator view:* which shows the list of all the operators of the plant. From here the user can navigate to a detailed view of the specific selected operator;
- *Machine view:* which behaves in the same way as the operator view, but focused on the machine present inside the plant;
- *Warehouse:* which shows the list of all spare parts and tools and allows to navigate to a graph of the utilization of each element, in the same way as the details of the event on the calendar;

To achieve this objective, it will take advantage of React, a JavaScript library, which allows the creation of a flexible and responsive interface, with broad compatibility with all browsers. React allows the webpage to quickly, and in real time update itself, displaying all the most recent information found in the database. It is important to display all the information in the clearest way possible. On part with speed and responsiveness, the focus should be put on the user experience. It is important to display all the information in the clearest way possible. Consequently, the frontend takes advantage of the routing capabilities of react to display the information into different pages better organizing and displaying all the data provided by the different sources.

4.2. Algorithms

The decision algorithms are based on python and are structured in different objects, two for each element (works, operators etc.) one class is for the definition of the attributes of the object itself, the other one defines the ways in which the instance of the object interacts with others. The decision process starts by retrieving the Json files of each external source from the database. Then using Pandas the Json files are converted in arrays in order to be analyzed and elaborated. One of the advantages of Pandas is the possibility to execute checks directly using the timestamp as a parameter, which allows fast controls over the possibility of scheduling a work in predetermined time slot. For the DSM are used different algorithms:

- *Scheduling Algorithm*: it generates the maintenance schedule forecast keeping into account all the constraints provided by the different systems;
- *Operator Algorithm*: it elaborates data related to the users to provide insights about their availability and suggestions on their optimization;
- *Equipment Algorithm*: it elaborates data related to the machines to provide insights about their maintenance activities and availability to optimize their efficiency;
- *Warehouse Algorithm*: it elaborates the data related with the warehouse and in particular the expected spare parts and tools transaction to perform a projection of their availability. The output of the algorithm is used to provide suggestions on the management of the warehouse (e.g., reordering of material);

5. Conclusions

The developed Decision Support System is an innovative solution to the problems related to the decision making inside the production plant. It will allow the Maintenance Manager to easily monitor the present and future status of the plant. The DSM has the objective of helping the Maintenance Manager in taking immediate decisions and to easily tackle unexpected events. This is why the system has been developed with the focus on speed, usability, and clarity, developing a fast and intuitive solution which will ease the work of the person in charge of taking decision over matters which, without any support, are mostly experienced based.
The system provides a wide variety of feature:

- Decision Support Module;
- Information System;
- Condition Monitoring;
- Online Quality Control;

At least in theory, in fact due to issues external from our range of influence the last to feature were not developed so they are impossible to be practically integrated. This is a symptom of a multi layered project, composed by different companies, which have to develop their own modules independently, but in the end collaborate, that had an insufficient level of managerial skills from whom had the task to coordinate the work between different partners. This had a snowball effect, because the absence of a constant guide in coordinating the different modules, ended up in delays in the delivery of certain functionalities. From our prospective we ended up waiting and loosing time because of the lack of a certain functionalities that would have been due from another partner. This also conditioned the testing phase, due to the absence of real data we had to create a synthetic set of data which would represent at his best the plant on which this first version of the system is based. Denying the possibilities of applying the DSM to a realistic scenario. Internally the situation was much different, the deliverables have been all updated on time and the project design and development had been constant for what was possible.
The system is still rough, missing some functionalities due to external causes, but also still lacking robustness since the testing had been very limited in scale. A margin of optimization is palpable both in the frontend and in the backend, in fact the API calling process requires a time which is acceptable for the tested scenario, but that will probably be unacceptable for any realistic 6-month schedule, so a work of optimization is required to entitle the system as market ready. The algorithms on the other end lack robustness, because they expect only the planned document formats and lack an optimal management of errors and unexpected situations. These are all problems, though, that do not change the innovative nature of the project in a sector often overlooked and with little literature which covers the same subjects. Is an attempt to coordinate different well-known practices into a unique system which tackles an, often perceived as marginal but actually fundamental, aspect of the production endeavor, maintenance.
Thanks to having modularity in mind since the beginning, the DSM is open to be updated in many different directions, depending mostly on the needs of the specific plant. In fact depending on each infrastructure it could be possible to add more and more features. For example it would be possible, even if very intense, to add a digital twin representation for the monitoring of the equipment. A system which monitors in real time the status of execution of the works, or a more complex predictive system based not only on operative research, but on more advance techniques which can take advantage on the latest trends in AI, Big Data and industry4.0.

6. Bibliografia

[1] R. D. &. P. A. Scarf, "On the impact of optimisation models in maintenance decision making: the state of the art," *Elsevier Science Limited*, vol. 1, no. 1, p. 9, 1998.

[2] N. Jun e J. Xiaoning, «Decision support systems for effective maintenance operations,» p. 4, 2012.

[3] S. SAGIROGLU e D. SINANC, «Big Data: A Review,» p. 6, 2013.

[4] F. Wortmann e K. Flu¨chter, «Internet of Things Technology and Value Added,» p. 4.

[5] Z. Weiting, Y. Dong e W. Hongchao, «Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey,» *IEEE*

SYSTEMS JOURNAL, vol. 13, n. 3, pp. 2213-2227, 2019.

.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Design and Implementation of a Decision Support System for Manufacturing Maintenance Optimization

TESI DI LAUREA MAGISTRALE IN

COMPUTER SCIENCE AND ENGINEERING: BUSINESS AND INNOVATION – INGEGNERIA INFORMATICA: BUSINESS E INNOVAZIONE

Author: **Roberto Santangelo**

Student ID:	927679
Advisor:	Marco Brambilla
Co-advisor:	Andrea Villa
Academic Year:	2020-21

Abstract

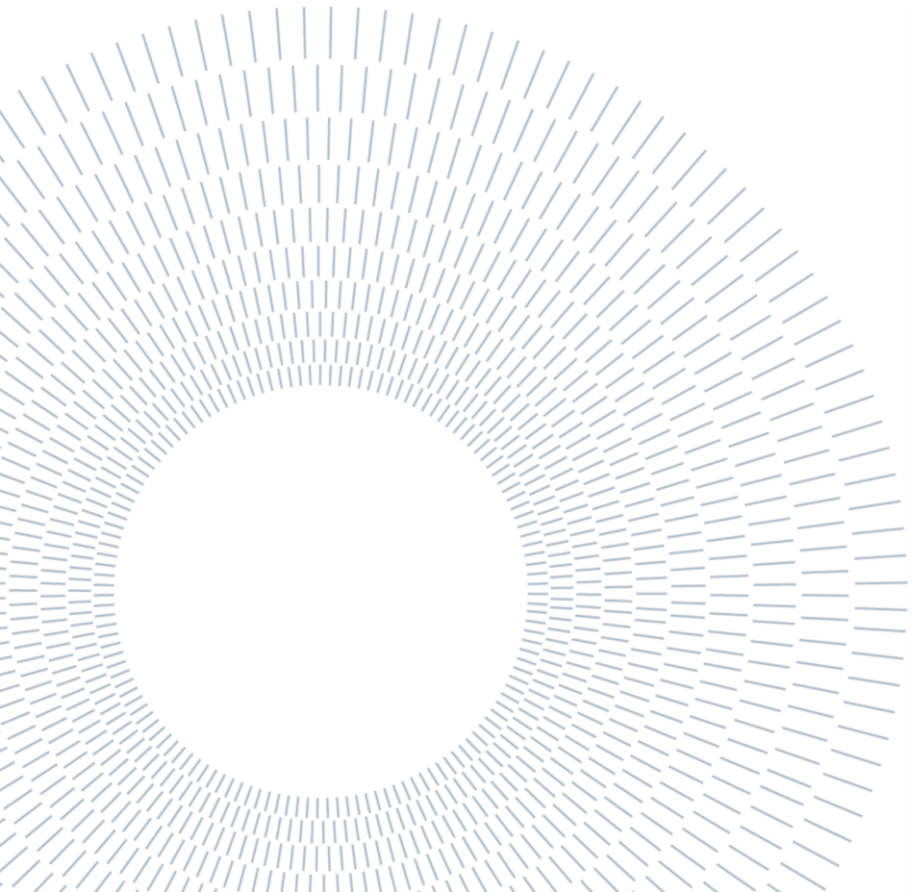
The thesis is about the definition of a Decision Support System for Maintenance Optimization. Since there is not a large bibliography about Decision Support especially in maintenance, so this project encapsulates an innovative spirit within. It is based on the idea that Decision Support is able to speed up and optimize maintenance operations and help the Maintenance Manager to take decision based not only on his own personal experience. The advantages of this Decision Support System, respect to others already existing, it incorporates different techniques to guarantee a higher coverage of the needs of the user. It combines optimized and enhanced scheduling, which provides a dynamic and fast suggestion for the scheduling of maintenance operations, with a Predictive Maintenance module, for a preventive approach, and an Online Quality Control Module to monitor the machine conditions based on a vision system built around 8 cameras, which constantly monitors the state of the pieces produced by the equipment.

Keywords: Decision Support, Maintenance Optimization, Enhanced Scheduling, Predictive Maintenance, Maintenance Operations, Decision Algorithms

Abstract in lingua italiana

La tesi riguarda la definizione di un sistema di supporto alle decisioni per l'ottimizzazione della manutenzione. Poiché non c'è una grande bibliografia sul supporto alle decisioni, specialmente nella manutenzione, questo progetto racchiude uno spirito innovativo al suo interno. Si basa sull'idea che Decision Support sia in grado di velocizzare e ottimizzare le operazioni di manutenzione e aiutare il Maintenance Manager a prendere decisioni basate non solo sulla propria esperienza personale. I vantaggi di questo Decision Support System, rispetto ad altri già esistenti, sono che incorpora diverse tecniche per garantire una maggiore copertura delle esigenze dell'utente. Combina tecniche di enhanced scheduling, che fornisce un suggerimento dinamico e veloce per la programmazione delle operazioni di manutenzione, con un modulo di manutenzione predittiva, per un approccio preventivo, e un modulo di controllo qualità per monitorare le condizioni della macchina basato su un vision system costruito intorno 8 telecamere, che monitora costantemente lo stato dei pezzi prodotti dalle apparecchiature.

Parole chiave: Supporto alle decisioni, ottimizzazione della manutenzione, enhanced scheduling, manutenzione predittiva, operazioni di manutenzione, algoritmi decisionali.



Contents

1. Introduction

1.1 Context

In the modern world where the concept of automation and optimization are becoming more and more relevant due to the increase in competitiveness and the faster changes happening in the industry world. This very dynamic environment led to the creation of new ideas and tools to keep the pace with change. A lot of importance is attributed, nowadays, to Big Data which, thanks to the latest technologies, can be used to perform analysis of both the internal and external context of a company, supporting the creation of more focused and optimized strategies. Crucial is also the concept of Internet of Things (IoT) which stands for the possibility of common object, machinery, cars etc. to be connected to the internet, and through it to send and receive information in real time, creating an interconnected net of object with the ability of communicating between them and with other entities, enlarging the spectrum of possible remote interactions with them. Maintenance is becoming more and more crucial and expensive in recent years, despite the better technologies available. This is linked to the constant increase for quality demanding in services, with better services is consequently less acceptable to delay maintenance and slow down the solving of the problem [1]. With new technologies maintenance becomes more complex and it increases the need for specialized technicians, consequently the outsourcing of maintenance operation has become a common practice throughout all kind of industries.

Maintenance and quality management has a high impact on sustainability for medium and large sized discrete manufacturing companies. These aspects are common to different sectors like Automotive, Aeronautics, Railway, Serial equipment, or Food & Beverage. We can notice that the involved scenarios represent two very different markets that have in common the following characteristics:

- *Complex maintenance management:* the target manufacturing context is characterized by a large variety of maintenance activities performed by different technicians with specific skills.
- *Decision Support System for the Maintenance Manager:* in this kind of context the Maintenance Manager plays an important role since it has to manage all the maintenance activities and try to optimize all the aspects related to the maintenance. This role expresses the need of having a decision support system that helps in managing the maintenance activities and suggests overall improvements.
- *Strict quality control requirements:* In all the mentioned sectors nearly, zero-defect manufacturing is compulsorily required by contract or even legal conditions. Therefore, heavy quality control walls are implemented at production plant to prevent defective units reaching the market.
- *Correlation of quality measurements with maintenance aspects:* in the target manufacturing context the maintenance has a strong impact on the final quality of the product and not only on the overall production efficiency. The quality measurements need to be correlated with maintenance activities to be performed in short time in order to correct the problem and proceed with the production. The low final quality generates out of standards product are often not fixable and that may cause product waste.

1.2 Problem Statement

The problem on which the thesis is going to focus is the *optimization of maintenance activities inside the production plant*, and more broadly the decision support for the Maintenance Manager inside the plant. This problem has as consequence some subproblems:

- *Absence of Information*: since most industrial machines are quite old, they do not provide much information about their internal state, mor often than not the understanding of the problem relies on the experience of the maintenance technician and the knowledge that he has of that particular machine. Nowadays though, due to the complexity of its management and the cost of maintaining one, do not have an internal and specialized team of technician, they tend to assign routine operation to generic technicians and to outsource the more complex operation. This leads to the problem of exchanging

information between the internal and the external technician. The internal technician it is usually not able to provide reliable and complete information about the specific problem, leaving the outsourced technician into a state in which it has to perform further explorations to understand and solve the problem increasing the time needed to complete the operation and stalling production in case the operation has to be performed on an essential machine [1].

- *Lack of scheduling optimization*: nowadays it is common practice to perform condition-based maintenance, in which decisions are based on the evaluation of the machine status through inspection and measurement [2]. This approach does not have a long-term objective in mind and reacts to events on a day-to-day basis, not allowing to plan strategies and solutions due to the absence of information.

A report by plant services in 2016 revealed that most of the plant managers are not satisfied with their organization's predictive maintenance program, and therefore are increasing their investment in areas such as control systems, predictive analytics, and mobility.

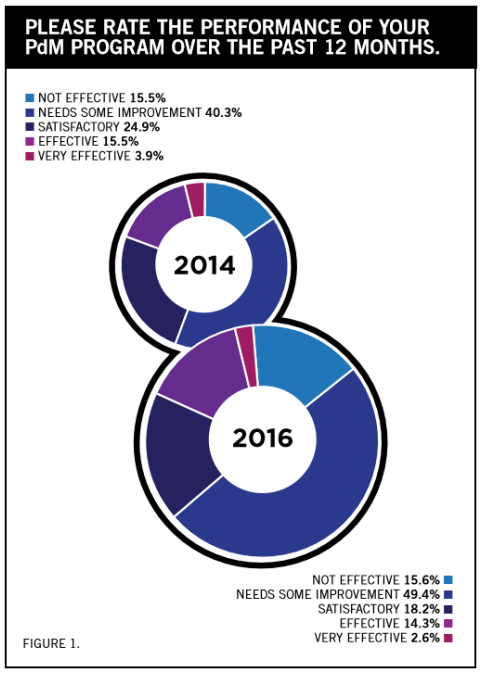


Figure 1.1: PdM satisfaction questionnaire results

At the same time, most of the respondents stated their intention to increase their budget for predictive maintenance related technologies. On a survey and market research from PWC they found that most of the companies trying to apply predictive maintenance (PdM) are still on an early stage of the development being only the 11% the companies that have a mature PdM system. Moreover, half of the 280 companies on the survey show their interest on increasing the maturity of their PdM systems and use predictive maintenance extensively using a continuous real-time monitoring of assets (PdM 4.0).

Railway sector is ahead of any others and PdM 4.0 is being used by a 42% of their companies while the average of PdM 4.0 usage is 11% across all sectors. Railway sector seems to be a good benchmark for the predictive maintenance market as their innovative attitude on this field has created a good environment for the development of a knowledge base related to fleet monitoring and advanced maintenance techniques.

1.3 Proposed Solution

The possible solution to tackle the upper mentioned problems is the creation of a system which would allow to monitor the status of the plant and to quickly gather information which can help the Maintenance Manager to better react to possible problems. A solution is possible, nowadays, thanks to the possibility of fully utilize the concepts of:

- *Big Data*
- *Internet of things*

These concepts are interconnected inside a production plant, in fact the presence of sensors inside the machinery produces a lot of information which, if correctly analyzed and elaborated could produce the basis on which the maintenance strategy can be built, shifting the attention from a condition based to a predictive approach [2]. So, the solution must incorporate all the aspects mentioned in the previous paragraph, to do so it is split in different modules:

- *Information System*: which is a personalized solution tailored around the specific production plant and it combines all the information produced by: the plant, the Condition Monitoring, and the Online Quality Control. It then feeds them to the DSM which will perform the final elaborations on the data.

- *Condition Monitoring:* which incorporates the predictive aspect of the solution, it estimates the Remaining Useful Life of a machine using AI to perform the prediction.
- *Online Quality Control:* which is a vision system based on image recognition, it monitors the pieces produced by the machines and, through a trained AI, recognizes possible defects in the produced pieces from the acquired images.
- *Decision Support Module:* It elaborates on the information provided by the other modules creating the final form on the information, the one that will be displayed to the end used and which will provide the necessary support. The core of the DSM is the implementation of an optimized schedule, which will provide the best possible schedule depending on the condition imposed by the specific plant.

The state of the solution is still not complete since two of the four modules (Condition Monitoring and Online Quality Control) are not completer. Still the innovative nature of the idea is independent from its realization, in any case the future integration has been already implemented in the other two modules, so the ideation and design part are completed and fully analyzable.

2. Chapter 2

Before diving into the actual idea and implementation, in this chapter is reported the scientific background of the Proposed Solution.

2.1 Big Data

Big Data represents one of the two bases on which the solution is based. There is not a general definition of Big Data, they represent all those information assets, technologies, processes, and social phenomena related to the creation and analysis of a great rage of information [3].

Figure 2.1: Static tag cloud visualization of key words appearing in the abstract of paper related to Big Data

The definition and identification of what exactly is big data it is quite confusing and chaotic, in fact as seen from the picture above, they do not refer only to dataset, but to a variety of different concept. For the sake of this thesis, we will focus on the most common definition regarding Big Data to clarify the concept.

2.1.1 Characteristics of Big Data

Even in the enormous variety of definitions and concepts behind Big Data it there are some key characteristics which are:

- *Volume*: as the name suggests it represents the amount of data which are generated by the user from his interactions with the web or all those information produced by sensors, bank transactions etc. The *IDC*[1] (International Data Corporation) estimated that in 2020 the amount of data created and consumed in one year would have reached 40 zettabytes[2] [4]. According to Statista (Figure 3.1) the actual reached amount was 64,2 zettabytes.

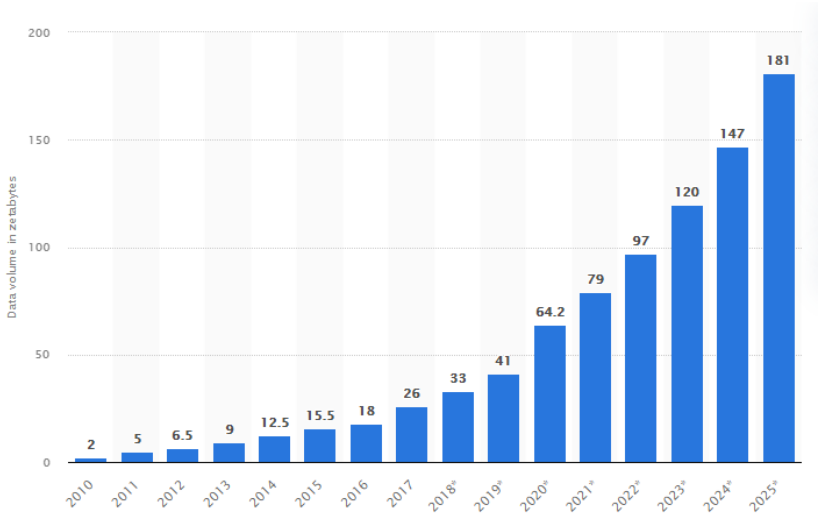


Figure 3.1: Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025(in zettabytes)

- *Variety:* Big data comes from a great variety of sources and generally has in three types: structured, semi structured, and unstructured. Structured data inserts a data warehouse already tagged and easily sorted but unstructured data is random and difficult to analyze. Semi-structured data does not conform to fixed fields but contains tags to separate data elements [5].

[1] IDC is a company which performs market analysis, consultants, and analysis specialized in Information Technology

[2] Zettabyte= 10^{21}byte

- *Velocity:* represents the rate at which new data become available. For companies the challenge is to be able to keep pace with this fast-changing environment. The faster they can analyze the data the faster they will get the information and result required to fulfil their strategy [4].

This nature of Big Data brought to the development of various techniques suited to their analysis and elaboration. Google's Map Reduce is a good example as a framework for the analysis of Data, it uses a divide and conquer method which separates complex Big Data problems into smaller instances to be analyzed in distributed platform and in parallel. Another example is Hadoop, which is a Java based framework and heterogeneous open-source platform. It is not a replacement for database, warehouse or ETL (Extract, Transform, Load) strategy. Hadoop includes a distributed file system, analytics and data storage platforms and a layer that manages parallel computation, workflow, and configuration administration. It is not designed for real time complex event processing like streams.

2.1.2 Data Awareness

It is a crucial factor in the interaction with Big Data, it states the level of comprehension that people interacting with data have of their importance. In fact, even if Big Data and their usage has been at the center of the attention for a while by now. People still tend to underestimate their importance, even in situation in which they could be seen as superfluous, if well utilized they could provide a significant advantage. Big Data give the possibility of obtaining information which would have never been available with older technologies and methodologies. The training of companies and employees in the usage and understanding of this phenomenon can significantly improve the performance in nearly every field of application. To sustain the importance of *Data Awareness* is possible to look at the Figure 4.1, which show the improvement in the supply chain due to an increase in the awareness [6].

Figure 4.1: Areas improved when using SC Analytics (Gii Finance Network, 2016)

2.2 Internet of Things

It is a term used to define the interconnectivity between different entities inside a network, which communicate via internet. The International Telecommunication Union (ITU) for instance defines the Internet of Things as ''a global infrastructure for the Information Society, enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies'' [7]. There is not a single globally shared definition, but the concept of IoT is deeply connected to the concept of industry4.0, in which machines and components gain an "intelligence" allowing them to communicate and exchange information. So, at its core, "IoT solutions typically combine physical things with IT in the form of hardware and software" [7], the value creation potential is retained by the possibility of each component to communicate as a single entity and, more importantly, as part of a complex net of different components. Therefore, a great amount of Data is generated every second, due to all the sensors present inside a machine linkable to the industry4.0. that is why the concept of IoT, and Big Data are so deeply interconnected, tackling only one of the two aspects would never result in the best possible outcome.

2.3 Decision Making

In this section are analyzed all the aspects related to the Decision-Making process, which represent the core concept on which the Proposed Solution is based.

2.3.1 What is the Decision-Making process?

During the production endeavor it is necessary to make decisions regarding different matters. We will focus on the decisions consequential to the breakdown of machinery

and its effects on production. These types of decisions are usually experienced based, meaning that they rely on the expertise of the person in charge, without any support from IT system, which could help to analyze Data and derive predictions and information helpful to the Manager, the absence of it results in an extremely aleatory decision, which is directly connected with the knowledge of the type of problem or of the context held by the decision maker. Another challenge of Decision Making is the wide variety of information sources which have to be analyzed:

- The Current health condition of each machine including down, running, idling, being maintained or just about to breakdown;
- The scheduled daily, weekly, and monthly maintenance plan;
- Machine health degradation profile;
- Throughput target and production rate;
- Costs of maintenance resource, i.e., labor, spare parts, tools, etc.;
- The system configuration and decision alternatives [8];

This delineates a very complex environment to be analyzed just with the Maintenance Manager's personal knowledge.

2.3.2 Decision Support

The concept of decision support represents the possibility, through the usage of IoT and the analysis of Big Data, to help the Maintenance Manager to take better decisions, based on a set of objective information obtained by the status of the elements of the plant. A Survey was conducted by a CIRP [3]Working Group on ''Flexible Automation-Assessment and Future'', stating a general dissatisfaction in maintenance costs [8]. Const and challenges in maintenance derive both from regular operations and extraordinary operations, these are both hard challenges to face: regular maintenance requires high scheduling effort and great attention, because a good regular maintenance lowers the risk of unexpected failures, which are challenges themselves due to their natural unpredictability.

2.3.2.1 Condition Based Maintenance

One of the main advantages that Decision support brings in parallel with Big Data and IoT is Condition Based Maintenance. CBM is *"the standard term employed to describe maintenance strategies determined on the basis of the actual condition of assets"* [9],

[3] THE INTERNATIONAL ACADEMY FOR PRODUCTION ENGINEERING

it involves the performance of maintenance tasks derived by the analysis and interpretation of hardware parameters. It allows real time diagnostics of impending failures and prognosis of future equipment health. Even if the arise of intelligent computers helped the creation of new systems for the prediction of failures [9]. The prognostics of the possible failure can be performed trough different approaches:

- *Model-based*: Centered on detailed knowledge of a system and its interlinkages, its use is limited due to the inherent complexity of modern industrial systems;
- *Data-driven*: Requires historical parameter collection from monitored assets; requires pattern recognition and machine intelligence techniques to realize actionable decision-making outcomes;
- *Hybrid*: Is a combination of the two aforementioned approaches requiring a joint analysis of both known information about a system in combination with sensed data points [9];

These three approaches all serve for the delineation of conditions a limitation to take into account when planning maintenance or taking decision. The best possible practice to follow is the combined usage of the data gathered from the machines, which defines a determined and objective set of predictions and conditions, with the opinion of the expert (Maintenance Manager, operators etc.), which can give a more realistic approach to the problem since is based on personal and real live events and not only over statistical analysis [10]. it allows to define a set of condition based both on objective and agnostic information, and information gathered inside the plant allowing to personalize the conditions depending on the specifics of the plant, these conditions could be:

- *Preferred periods*: period in which is preferred to perform the maintenance activities, due to a lower impact on production, a more forgiving timeframe etc.;
- *Periodicity of maintenance*: regular maintenance is normally executed in a plant, it is important to consider the periodicity, so it can be transferred in the personalized solution;
- *Limitation of the machine*: which are the operators allowed to perform operations, the importance of the machine in the production etc.;
- *Other special condition;*

The combination of all these aspects allows the creation of a system which can Fully represent the complexity of the production plant.

2.4 Decision Making in Maintenance

There are different approaches to maintenance, as already mentioned in the previous paragraphs. In this section all the different approaches to Decision Making in Maintenance are analyzed in detail.

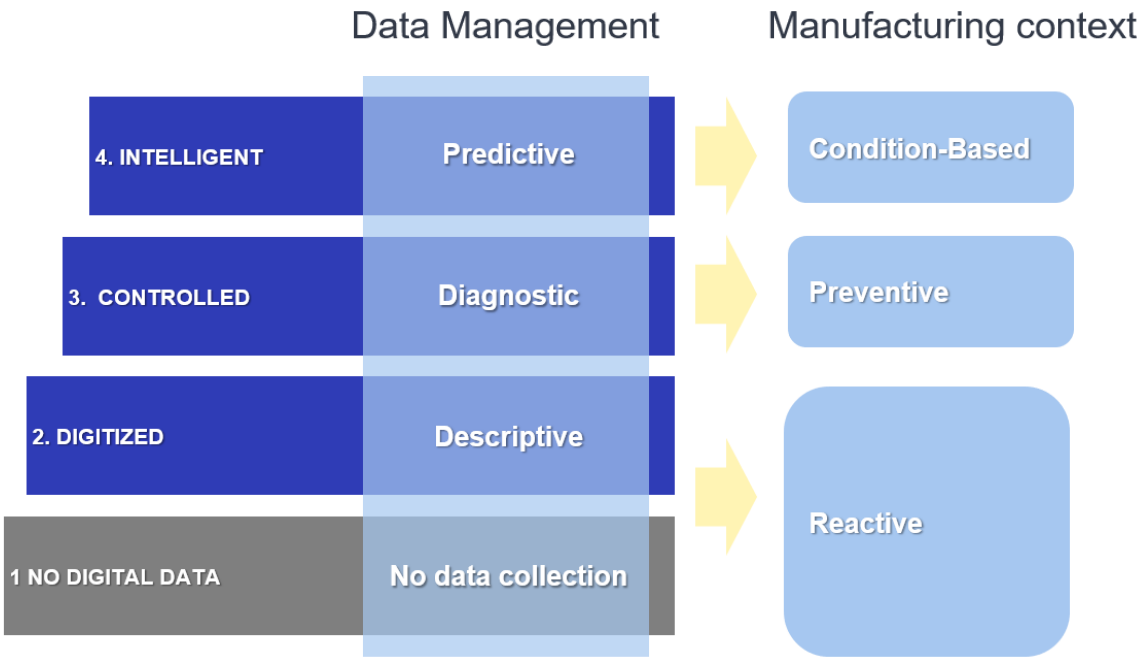


Figure 5.1: Data management in Maintenance Context

In the picture above it is possible to see the 4 main levels of data integration inside the maintenance operations, which derive in three different contexts.

2.4.1 Reactive Decisioning

In this context decisions are taken after the event occurred, so the maintenance operation is a reaction to the random event. Both non-digital and digital data can be used in the decision making, but it is mainly used for retrospective and descriptive purposes, so they do not serve as a form of monitoring or preemptive behavior.

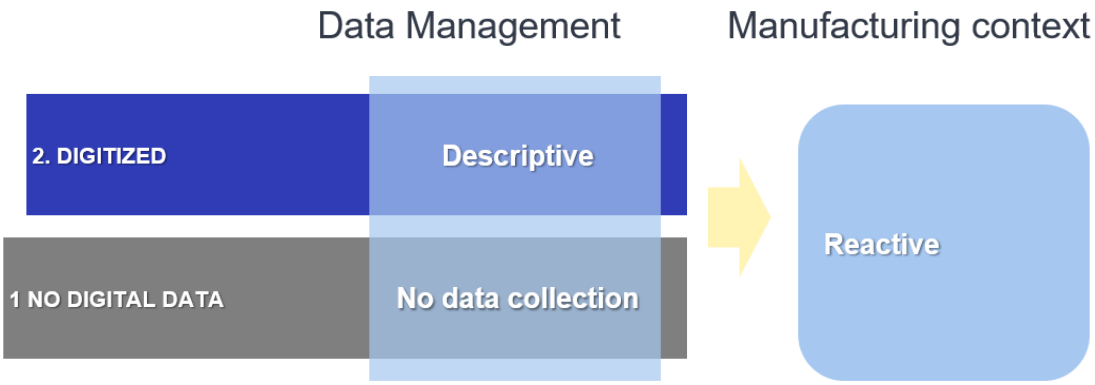


Figure 6.1: Reactive Decisioning Context

Pros:

Suitable for random and unpredictable events, which can only be handled after they have occurred.

Cons:

Does not take advantage of the use of Data to optimize the regular maintenance operations, losing the advantages of preemptive actions.

Example:

Quality checks can be performed on the production lines in order to monitor the presence of damage and the compliance with standards level of production. When a certain number of products fail the checks then maintenance activities are executed on the production equipment to find malfunctions or issues.

2.4.2 Preventive Decisioning

Decisions are taken regularly with a fixed frequency over time of usage. So periodic checks are performed to ensure a lower window of uncertainty over the condition of the machines. There can be a lot of different criteria to establish the frequency (monthly, every 100 items, once every 100km etc.). In this context is crucial to find the right balance decision made too often, which bare a higher cost, and too sporadically, which could cause a higher idle time of the machines.

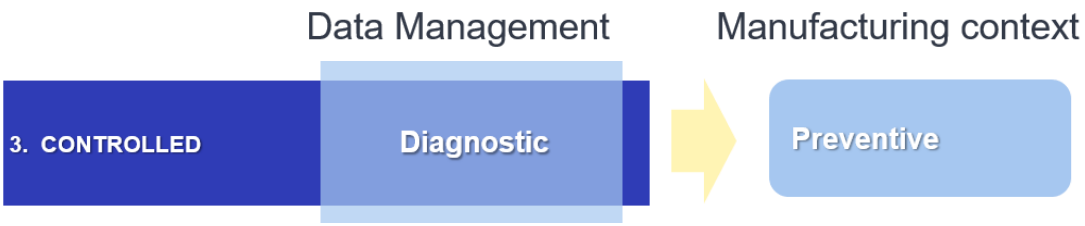


Figure 7.1: Preventive Decisioning Context

Pros:

Appropriate strategy for tasks with a constant workload that should be continuously handled in the same way.

Cons:

Just like Reactive Decisioning it doesn't take advantage of the possibilities of using Data to dynamically adapt and optimize the strategy.

Example:

The maintenance manager decides when schedule the ordinary maintenance activities, based on:

- the knowledge of the machinery behavior
- the general needs of the production plan
- the average availability of the operators

2.4.3 Condition Based Decisioning

Decisions are taken the actual condition of an asset with a certain criterion. The criterion could be the Exceeding of a predefined threshold established by domain experts; the outcome of a predictive model that have forecast capabilities learned from historic data. Artificial Intelligence and Machine Learning techniques are used to find strong relationships between input and output signals.

Figure 8.1: Condition Based Context

Pros:

Appropriate strategy for critical events that need to be anticipated as soon as possible in order prevent them from happening.

Cons:

Requires a lot of effort in and spending in the development of the predictive systems

2.4.4 Proactive Decisioning

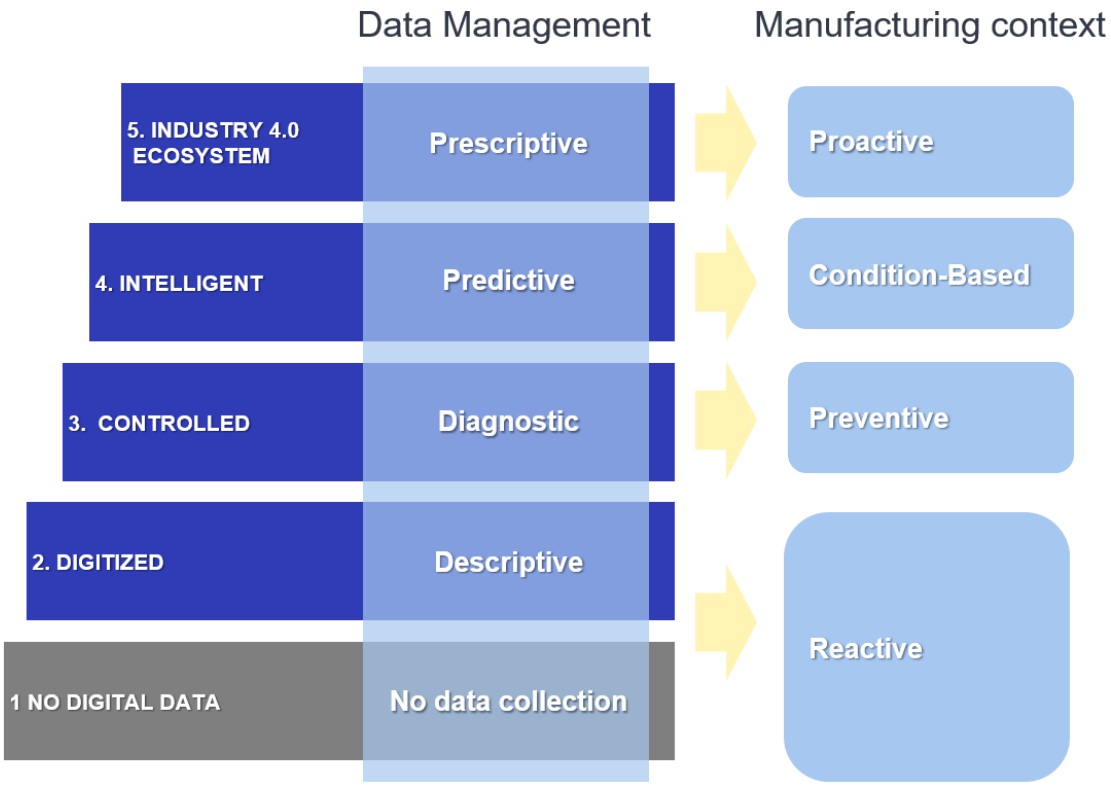


Figure 5.2: Proactive Decisioning Context

Decisions are taken directly recommending the best actions to be performed in the context of interest. These actions are decided through the usage of operative research algorithms which dynamically elaborate the best possible solutions for each problem, depending on different factors and then they maximize the gains or minimize the drawbacks. Some of the possible factors could be:

- Minimize the machine downtime
- Maximize the operators' worktime schedule

- Maximize production
- Etc.

The decision could be fully automated executing the best suggested action, but usually the final decision is left to a person that makes the conclusive choice. Enabling an optimized collaboration between the technical knowhow of the Maintenance Manager and the optimized and agnostic calculations of the algorithms.

Pros:

Appropriate strategy for reduce the complexity of repetitive time-consuming tasks that involve the analysis of huge amount of heterogenous data.

Cons:

Long and complex development process which has to combine a lot of different components to gather and analyze the data, which is time consuming and expensive.

2.5 Predictive Maintenance

Predictive Maintenance is one of the latest policies adopted for maintenance, especially in those cases in which reliability is a key factor for the correct execution of the production processes. It consists in the forecast of faults and failures for a specific piece of equipment in order to optimize maintenance efforts. This process is performed through the usage of historical data, which are analyzed to produce the forecast, and they provide a source of diagnostic and prognostic information [11].

The aim of Predictive Maintenance is the assessment of the health status of key components, regardless of their maintenance status, in order to make predictability achievable [12]. This approach has the possibility to minimize maintenance cost and also to extend the machinery useful life.

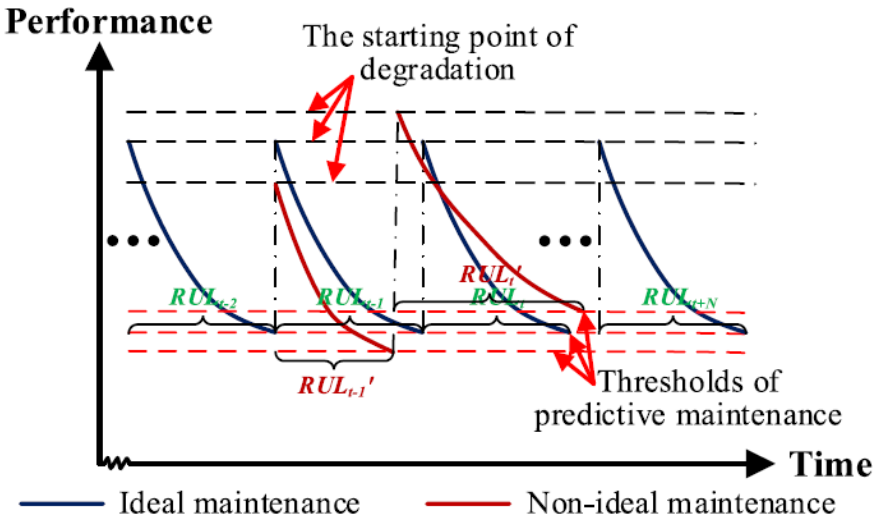


Figure 9.1: Predictive Maintenance of degraded equipment

Looking at **Errore. L'origine riferimento non è stata trovata.**1 and assuming that the equipment can be repaired in the appropriate timespan. The curves in the picture show the performance trend of the monitored object. The blue curve indicates the ideal maintenance where the RUL (remaining useful life) remains basically identical. The red curve represents an early or late maintenance, and its visible how the RUL is different from the ideal state, the dotted lines represent the performance thresholds in each maintenance cycle. **Errore. L'origine riferimento non è stata trovata.**.1 represents how the world is shifting to a paradigm in which performing maintenance in the most precise way possible is becoming more and more important [12].

3. Chapter 3

Most of the solutions dedicated to the Maintenance Management are categorized as Computerized Maintenance Management System (CMMS). These solutions often integrate more features extending their scope in the more generic Assets Management area. In addition, what characterize these solutions is that they are vertical solutions where all the components are part of a unique system. In other words, all the offered features are integrated in a single platform with strong limitations in terms of customization.

3.1 Decision Support System for Predictive Maintenance

The aim of this solution is to produce a system which would determine the time in which the next Predictive Maintenance stem must be taken. It is based on the following assumptions:

- Predictive Maintenance frequency established by the original manufacturer;
- Both historical and forecasted data are used;
- The system must be independent from the operator, no usage of manually recorded data;
- The existing relationship between company parameters and processes parameters must be taken into account;

This system was designed on the Hessel company in Slovakia and uses the following company information and parameters. The historical data came from existing company reports/documents:

- *BOM*: The bill of material of finished goods because the company took track of the amount of raw material used. This data has been combined with the amount of scraps, because they can significantly change this number
- *Working Instructions:* they are used as second input. WI describe the number of production steps and their details for the operator.
- *Standard Time of Operation:* this data was applied to the calculation, and it appear as a cycle, showing the frequency to achieve a standard time for operations. It can be represented either by the number of pieces produced by minute/hour/shift, or as a the time used to complete a single operation

- *Equipment List:* which comprises information on how each operation should be performed and its working place, work equipment, the usage of each piece of equipment, and the processing of defined number of materials (based on BOM)

So, the proposed system is based on the above parameters. EL and WI ensure the process flow, in the assessment of the equipment and tools for PM we gather information from these two sources. Each operation takes different time, and a different number of cycles is needed to produce the final product. It is noticeable that, even with parallelization in the execution of the operation, the real usage time of the single tool doesn't change. Preventive maintenance, though, should not be based only on historical data, the innovative side of this solution is the usage of forecasted information. It is used a Material Request Planning (MRP) calculator. Material usage (consumption) applied as a cumulative parameter and MRP applied as a variable forecast parameter allow us to determine the material frequency at the production line very precisely [13].

$$TTPM = \frac{n(F-(ICC-(PM_R \times PM_F)))}{\sum_1^n MRP_i} \quad (1)$$

$$ICC = \sum_1^N M_C \times EL_T \quad (2)$$

To test this solution one operation step using one tool family was selected. A tool family is composed by 85 tools used to process 85 different raw materials at the same stage of production process. For understanding of usage reference equation, this tool family is characteristic by cycle counting – this means that one cycle refers to one material unit. As it was mentioned above, it started with assigning of all 85 tools to raw material which are processed by means of them. To do this, the equipment list or other available company documents such as a control plan were used. Table 1 describes an example of EL: it expects to obtain the information about the number of processed materials from the inspection of records of finished goods at the end of the line. To transfer and evaluate such data the BOM of each product is needed, in order to multiply the structure of particular materials at the final product with the time or cycles used (Table 2). Other input data such as material consumption, MRP forecasted material volumes came from company databases. The information on service interval or several intervals given by tool producer is the last input necessary for defining the final parameter – time – for preventive maintenance (TTPM).

Project	Tool family	Tool number	Assigned raw material overworked	Cycle time in min	Cycle count in pc
Automotive	Press	335-1597	41221597	0.00	1
Automotive	Press	333-1597	41221597	0.00	1
Automotive	Press	334-1597	41221597	0.00	1
Automotive	Press	130-0497	41221597	0.00	1
Automotive	Press	336-0497	41221597	0.00	1
Automotive	Press	274-1600	41221597	0.00	1
Automotive	Press	341-1648	41221597	0.00	1
Automotive	Press	342-1648	41221597	0.00	1
Automotive	Press	337-9088	41221597	0.00	1

Table 1: Example of equipment list [13]

Project	PN	Material number	Structure
Automotive	16-B-185-1692	xxxx	1
Automotive	16-B-185-1692	41221597	6
Automotive	16-B-185-1692	xxxx	1
Automotive	16-B-185-1692	xxxx	1
Automotive	16-B-185-1692	41221597	12
Automotive	16-B-185-1692	xxxx	1
Automotive	16-B-185-1692	xxxx	1
Automotive	16-B-185-1692	xxxx	1
Automotive	16-B-185-1692	41219088	21

Table 2: Example of the BOM [13]

Designed algorithm was applied to case study data. Tool family used in the example comprises three service intervals with different tasks; these intervals are called intervals ’A, B, C‘ by the software. Thanks to this simple navigation, it is easy to find a To-do list in the maintenance plan for each tool. The case study of possible implementation is shown in Table 3. Let us focus our attention on the example marked with yellow color. Selected tool with number 341-1648 belongs to examined press tool family. As it was mentioned above, there should be carried out a different set of preventive maintenance of the presses at all three levels of use. Task ’A‘ is performed after 150,000 cycles, task ’B‘ after 500,000 cycles and task ’C‘ after 1 × 106

cycles. Using the information from Table 1, we already know that raw material assigned to this particular tool number is 41221648. In order to calculate the number of cycles and fill the data in column Cycles/Time, hrs (Table 3), the number of materials used in production up to present time is uploaded.

Tool number	Tool family	Cycles/Timed control			Cumulative	Dispo	Cycles Control			TTMP			
		A	B	C	Cycles/time in hrs	Cycles/time in hrs	A	B	C		A	B	C
335-1597	Press	150,000	500,000	1,000,000	4,370,520	187,000	30	9	4	Weeks	1.5	3.4	3.4
335-1597	Press	150,000	500,000	1,000,000	5,827,360	236,000	39	12	6	Days	-2.3	2.9	5.0
341-1684	Press	150,000	500,000	1,000,000	1,456,840	109,000	10	3	1	Weeks	1.8	5.0	5.0

Table 3: Examples of DSS output - calculation of time to preventive maintenance (TTPM) [13]

This is valid for material which reached the form of final products. MRP provides more information: it describes the expected use of the press machine for the period. Once the PM records are stored, the conversion of the data leads to new timeframes for condition-based checks. The new plan is produced as soon as data are downloaded, and input data are uploaded. To prove the advantages of this system the results have been compared with the old system as shown in Table 4.

Tool number	Annual number of cycles	Cycle control 'A'	Number of condition-based controls	Number of time-based controls if weekly
341-1648	5,262,000	150,000	35	48
326-1577	1,032,539	150,000	7	48
50-0664	504,851	150,000	3	48
8-0888	210,152	150,000	1	48
334-1597	8,778,667	125,000	70	48
342-1648	3,545,600	125,000	28	48
			145	288

Table 4: Comparison of time-based controls and condition-based controls using the number of checks [13]

It is evident that the application of designed system in our case study decreases the number of needed maintenance controls by 75%. Moreover, this system provides the possibility to follow tasks per time priorities, which is not possible when using time-based management, as in spite of its literal meaning it does not allow any impartial priority sorting.

3.2 The Serena Project

The scope of this project is the creation of a truly interconnected system in which all machine data are available, allowing easier maintenance in case of unexpected events, and minimization of the cost of production downtime. This is made available through [14]:

- Advanced gateways for collecting sensor data and extracting meaningful information for the condition of the equipment;
- AI methods for equipment health assessment;
- Hybrid predictive analytics;
- Predictive Maintenance and cloud-based communication framework;

The system is composed by the following key elements:

- *Work center*: defining the physical place where a maintenance or production activity will take place, e.g., an assembly station. A work center is linked to certain properties, such as geometrical characteristics of its physical layout, I/O streams, etc. The work center element is closely coupled to its physical assets, such as equipment/machines/tools. The equipment of a work center is used by resources, e.g., human operators, to execute a specific job.
- *Job*: is the superset of the activities that take place in a certain period in the workstation. A job is comprised of tasks. Each job has a set of constraints, such as completion time, assembly sequence, bill of material, required tools, etc.
- *Resource*: responsible for executing a task in a specific workstation. In this study, as resources are considered operators, robotic manipulators and mobile robots for logistic operations interacting with the work center.

A separate element is considered for capturing and representing operator's feedback. Purpose of integrating this input is to enhance the effectiveness of the generated schedule from the operator's perspective, either related to a manufacturing or maintenance operation [14].

The approach of this project is based on a decision-making framework for selecting the most appropriate time for scheduling maintenance operations. The system receives in input the list of productions and maintenance tasks alongside the constrains that may arise from each production facility. The constrains vary from facility to facility and can be: operation time, arrival, and due date of the job, etcetera.

Then the input format is evaluated by a set of weighted criteria towards the creation of different schedules. The criteria are based on the production targets of each facility

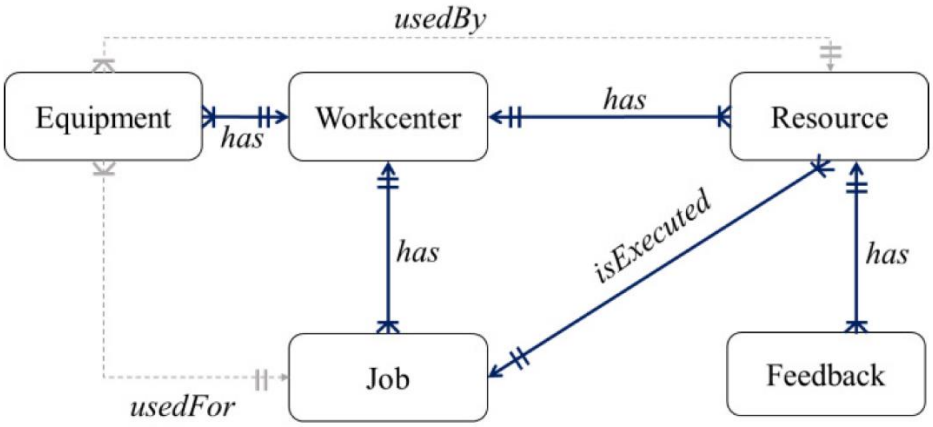


Figure 10.1: Main scheduling entities relationship [14]

Some of the implemented criteria are:

1. The total time for the tasks' execution, ΔR, is estimated as the sum of the completion time of the assignment tasks both to the human maintenance operator and the monitored equipment. This criterion is based on the following equation

$$\Delta_R = \sum_{i=1}^{n} \Delta_E^i \quad (3)$$

Where:

- Δ_E^i: execution time for a task i that is assigned to the human maintenance operator and the robot/rolling mill;
- n: the total number of tasks that have been assigned to the selected resources;

2. The overall cost of the selected resources, K_{total}, is estimated as the sum of the operating cost of each resource for each task. The equation describing the correlation is:

$$K_{total} = \sum_{j=1}^{N} K_R^j \quad (4)$$

Where:

- K_R^j: operation cost of resource j that is responsible to perform each task;
- N: the total number of resources;

generated schedule includes both maintenance tasks best fitted to the existing production schedule. Depending on the criteria selected for the evaluation and their weights it is possible either to change the production schedule or keep it intact [14].

3.2.1 Implementation

The scheduling tools implements a generic multi-criteria algorithm to produce a series of tasks/resource assignment. The design of the components that realize the scheduling application follows a client-server approach where most of the functionality resides on the server side and the client applications mainly support user interaction [14]. The data handling mechanism is based on a microservices architecture, the application is deployed as a set of small services, which opens possibilities for modularity and keeps every service independent. The usage of the Microservice architecture has been implemented in the scheduler data consuming Application Programming Interface (API) that stands between the scheduler and the data and information providers such as:

- Shop floor Systems
- Supervision Systems
- Safety Systems
- Legacy Systems

The various abovementioned systems and any other that could provide data can connect to the API through well - defined communication protocols, such as Representational State Transfer (REST), Message Queuing Telemetry Transport (MQTT) or other. The API due to its lightweight nature that is provided by the Microservices Architecture can be rapidly deployed in Internet of Things (IoT) devices and containers. The implementation of the API consists of the following components:

- Communication Traffic Handler is a component that manages the incoming traffic from the various sources and distributes them to the next component. Acts as a dispatcher server.
- Data and Information Normalization Handler is a component that normalizes all the various types of user info into JavaScript Object Notation (JSON) objects for the next component to manage the data
- Data Verification Handler is a component that verifies that the data that was transformed into JSON objects is in an appropriate format to be sent to the next component
- Entity Mapping Handler is a component that creates a mapping between JSON objects and Entities that are corresponding to the Database schema of the scheduler

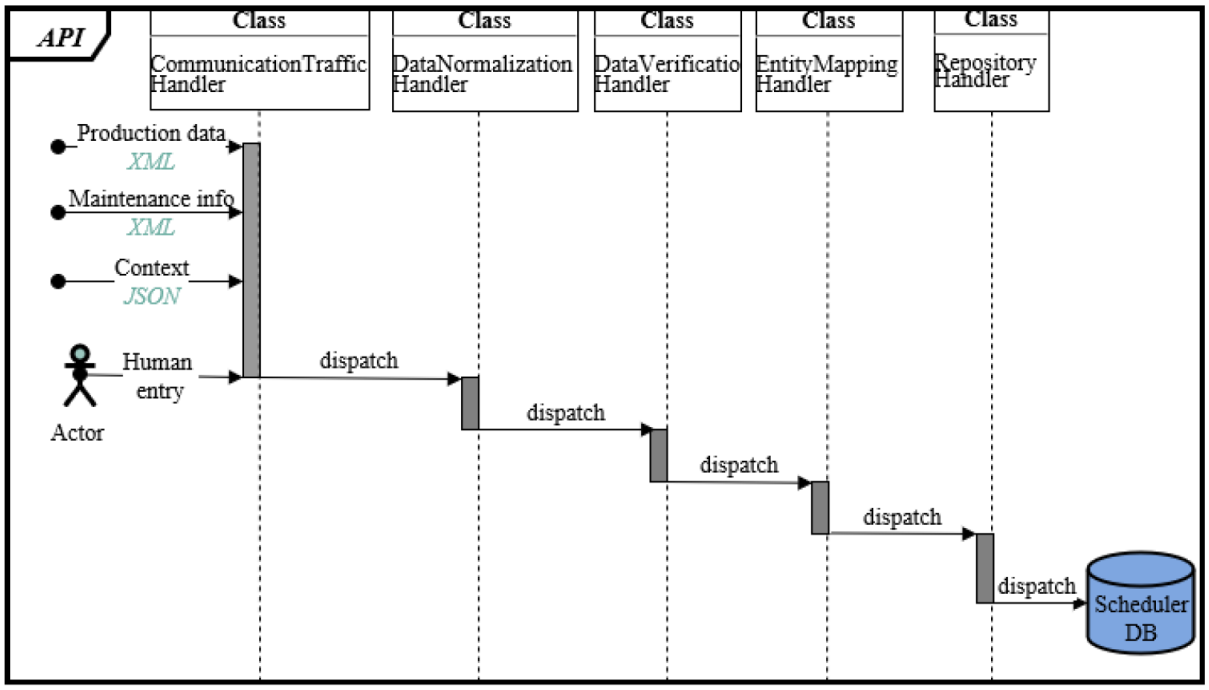


Figure 11.1: Data consuming API sequence diagram [14]

- Repository Handler is the component that creates the connection between the API and the Database of the scheduler and passes the info to the Database. Additionally, it can manage different schemas depending on the required schedule.

The technology behind the API is Java with Spring Boot, Spring Data Rest and Hibernate ORM. The communication between the components of the API is achieved

with REST calls using JSON objects. The input values are retrieved from a MySQL database through an Object Relational Mapping framework. It is used to map the values coming from the database to the classes present in the client application. The common language used is XML and the result file contains a list of tasks with the assigned resource. The architecture can be seen in Figure 12.1. The system allows the creation of a schedule where a specialized team is needed. The classification is based on the authentication of the user.

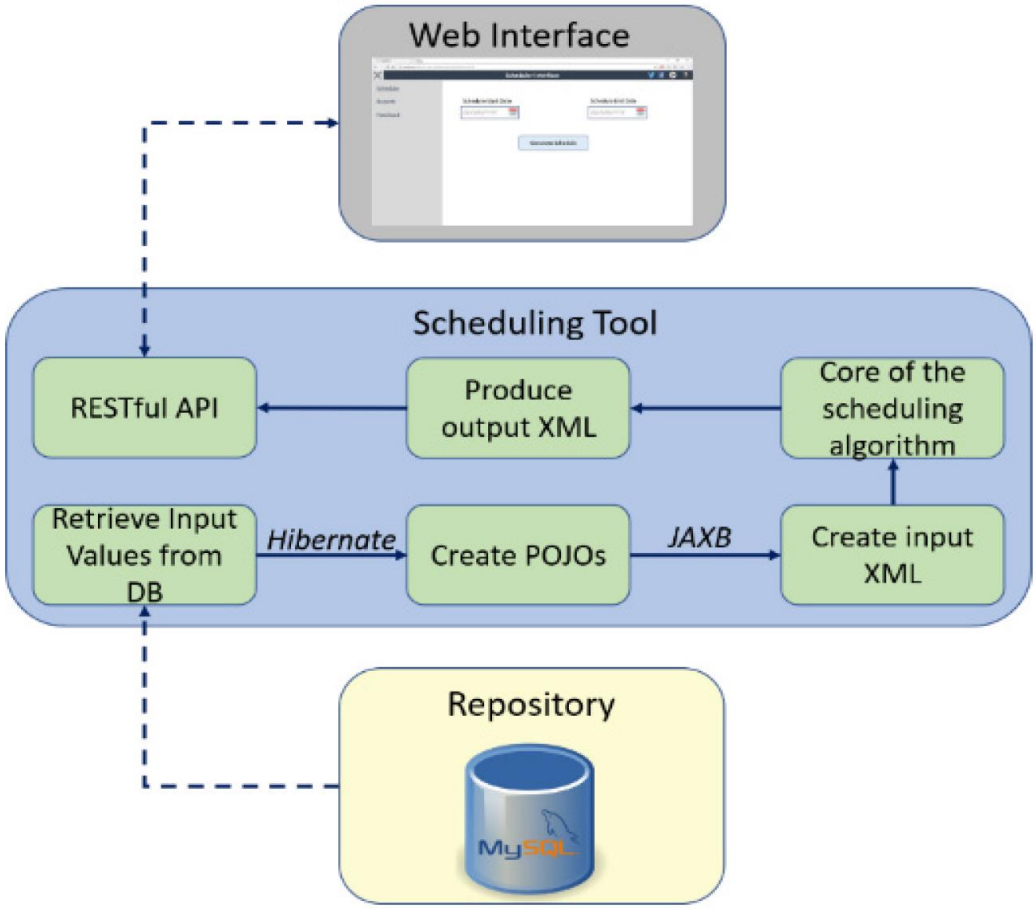


Figure 12.1: High level SERENA architecture [14]

4. Chapter 4

In this chapter it is presented the design thinking behind the proposed solution. It will cover the general architecture, followed by the interactions between components.

4.1 I4.0 Ecosystem DSM description

Properly coordinated decisions can improve production related processes when they are made on the basis of information that is reliably transferred through the hierarchical levels of control and management. Decision support systems aim to improve efficiency and speed of decision-making activities related to any organizational business process in order to achieve high productivity and cost effectiveness. These systems usually do not necessarily make a final decision themselves, but they help to automate the most repetitive and time-consuming parts of the managerial processes in order to facilitate the decision maker's task.

Figure 13.1: Ecosystem DSM logo

Decision support derives from a large variety of multiple heterogenous sources of information. So, the first action to perform during the design was the mapping of all the different data sources and the needs of the stakeholders. After the mapping the decision *problem* is described as a set of constrains, the decision *solution* is composed by the actions that are able to satisfy those constrains. Since there could be multiple solutions, a set of *optimization metrics* is defined in order to identify the best possible solution. Finally, in order to fully support the decision maker in taking decisions, the system provides a set of *frontend dashboards* that allow to switch between summarized KPIs and detailed information about of the current and expected situations. A Decision Support System can be used to generate a suggested schedule for the Maintenance Manager taking into consideration a set of metrics to be optimized and constraints to be respected. The typical metrics evaluated in the optimization of the manufacturing maintenance activities are:

- Reduction of the impact on production activities and their priority (from production plan);
- Minimization of late maintenance, this means the reduction of the time between the expected start of the maintenance activity (e.g. because of a predictive maintenance algorithm expected failure) and the maintenance execution;
- Minimization of machines downtime, in other words reduction of the timespan used to perform all the maintenance operations;
- Reduction of workforce utilization, for example by using the available resources in a more effective way;
- Reduction of spare parts and tools utilization, by optimizing the number of parts or tools required during the maintenance activities;

During the optimization process of maintenance activities, the following constraints are typically taken into account:

- Workforce availability (generic, per role, per day/hour, shifts);
- Spare parts and tools availability;
- Periods when machines cannot be stopped (e.g., from production plan);
- Plant information (e.g., interconnection between machines);
- Already scheduled maintenance activities;

To schedule the maintenance activities, an important source of data is represented by the **predictive maintenance** models. These systems are able to predict the health status of the operating machinery through a real time monitoring of the machinery. The result is an expected remaining useful life (RUL) of the machine that is converted in a time-window of opportunity where the maintenance operator can fix the detected problem before that it actually happens. The final big picture involves all the key elements that can be used by a Decision Support System: constraints, optimization metrics and the time window.

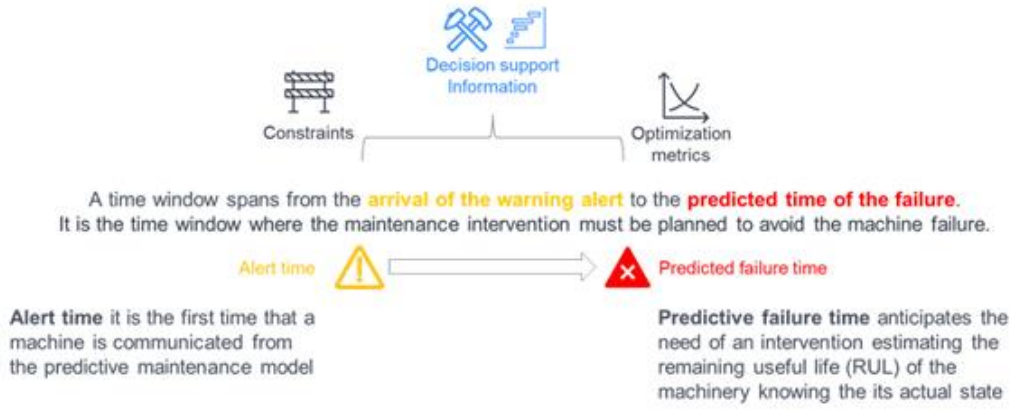


Figure 14.1: Decision Support System decision time-window

At the same time the online quality controls can provide an instant picture of the production runtime. This information can be used by the Decision Support System to validate its prediction and eventually suggest corrective actions in case of unexpected situation.

Decision support systems can be used not only to schedule the maintenance activities, but they can also promote many types of analysis about the data they integrate, i.e., personnel, maintenance activities, warehouse, and many others. Collecting historical data about the execution of the maintenance activities, the system can compare the planned work and resources information with the actual ones. These analyses allow to better plan future occurrences of those activities taking care of time-variant aspects like:

- Estimate the quantity of tools and spare parts needed for a specific work;
- Propose a new restock periodicity of spare parts to be kept in the warehouse;
- Correct the maintenance work duration;
- Discover potential low performance of the maintenance operators, in order to suggest specific training or coaching;
- Calculate useful KPIs about the availability of the operators, the downtime of different types of machines and the frequency of ordinary and non-ordinary maintenance activities;

4.2 DSM Design

In this section are presented all the steps made for the design of Ecosystem DSM, from the requirement analysis to the interactions between the different components, and the architecture. The information presented are related to the scenario in which the application has been tested, so on the information coming from the plant of Novameta in collaboration with InTechCentras.

4.2.1 Application Architecture

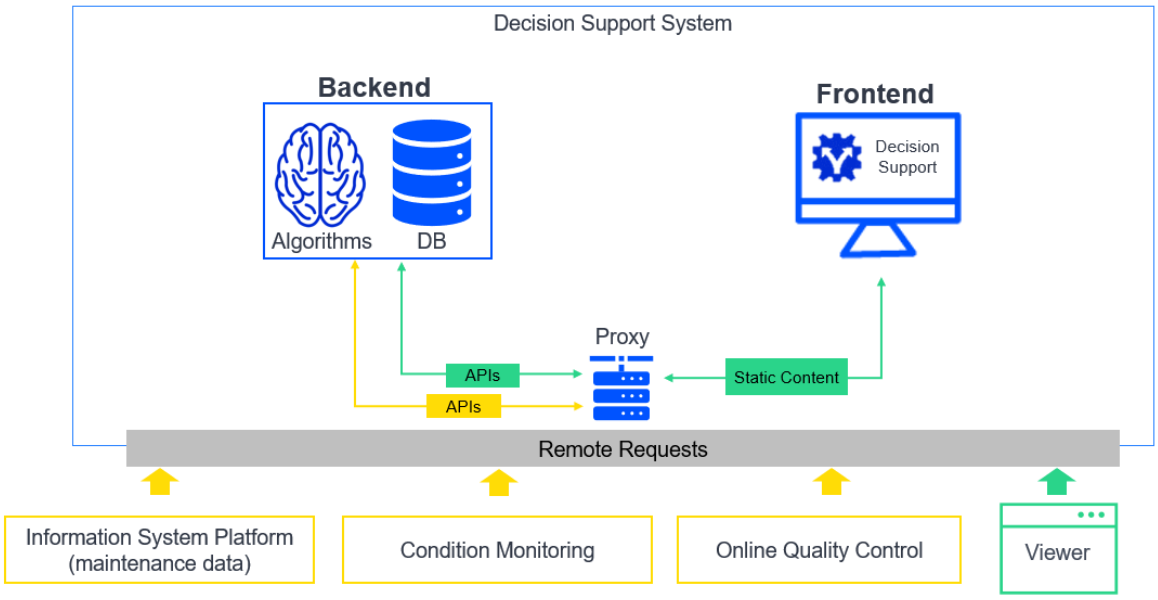


Figure 15.1: Decision Support Module Architecture

The system in Figure 15.1 is composed by four main blocks which are responsible for different duties inside the DSM:

- *Frontend*: represents the interface between the user and the system, allowing him to interact with all the functionalities exposed by the other components and assisting the Maintenance Manager in the decision-making process;
- *Backend*: represents all the system components dedicated to the Decision Support Algorithms execution (algorithms) and the storage of received data (database);
- *Proxy*: it acts as a router for the data between Frontend, Backend, and the Remote Requests;
- *Remote Requests*: they represent all the possible requests which can come from outside the DSM, they can be either: Data sources or Frontend users. Data sources are represented in yellow and communicate with the backend, Frontend users are represented in green, and they interact with the backend after getting the Frontend static content. In this activity the following external sources are identified:
    - *Information System Platform*: which gathers all the general information about the plant and how it manages the maintenance operations;

- *Condition Monitoring*: which gathers information on the remaining useful life of the machine through a predictive maintenance algorithm;
- *Online Quality Control*: which gathers information from a Deep Learning based vision system, that is able to identify defects on the product at the end of the production line;
- *Viewer*: which requires the visualization of the frontend and interacts with its functionalities;

4.2.2 Requirement Analysis

In this section we will analyze all the requirements raised by the inspection of the plant. Due to the Covid-19 pandemic it hasn't been possible to visit the plant in person, so in order to collect all the information needed a series of interviews has been done with different roles inside the maintenance processes of the plant: from the Maintenance Manager of Novameta to the operators who operates on the machines.

4.2.2.1 Maintenance Actors

This paragraph shows the main actors that are involved in the maintenance process are described. These actors are defined considering the requirements collected in the Novameta plant, but they can be present in different plants of similar size. The following roles are identified:

- *Maintenance Manager*: manages the whole maintenance process, from the assignment of the tasks to the final decisions on how and who should perform the operation. He is also in charge of supervising all the aspects linked to the maintenance organization (warehouse, training etc.);
- *Internal Maintenance Technician*: is an experienced employee with peculiar technical qualifications, able to perform more complex maintenance activities;
- *Operator*: is a generic operator that is dedicated to the production and can perform basic maintenance activities;
- *External Maintenance Technician*: is a highly specialized technician who is called whenever an issue cannot be solved by the plant's Internal Maintenance Technician;

Depending on the type of maintenance, different qualification may be needed, therefore different roles are assigned by the Maintenance Manager to the specific maintenance activity.

As regards the tasks dispatching, the Maintenance Manager often communicates directly to the operator via voice or phone. Anyway, the Maintenance Manager is

equipped with a fixed desktop device while Maintenance Technicians are equipped with mobile devices that can be used to perform actions on the warehouse and to receive specific instructions about the maintenance procedures.

4.2.2.2 Maintenance Works

This paragraph contains a description of the tasks (or activities) that are performed during the maintenance process. The tasks can be divided into two main categories:

- *Ordinary maintenance*: composed by known actions to be performed regularly to prevent damage or just to allow the regular production. It is addressed through a recurring task;
- *Extraordinary maintenance*: composed by actions to be performed in case of extraordinary events, like machine breakage. The detection of problems and breakdowns is mainly done through visual inspection during the production or maintenance process. The identification process is structured as follows:
    1. The operator notices a problem during the production phases;
    2. The operator fills a non-conformity form and sends notification to the Maintenance Manager;
    3. The Maintenance Manager evaluates the entity of the problem and decides if that maintenance activity can be performed by the Internal Maintenance Technician or if an External Maintenance Technician is required. A one-time work is scheduled to address the problem and associated to the specific user;

Depending on their recurrency the tasks can be:

- *Recurring tasks*: these kind of tasks repeats in time with different frequency. These activities are ordinary maintenance activities, and they can be classified into:
    - *Daily*: low effort activities performed daily. They generally do not need to be assigned since they are performed by the operator at the beginning of the shift;
    - *Weekly, Bi-weekly, Monthly, Yearly*: high effort activities (e.g., longer clean operations, machine checks, generic repairing, etc.) performed with a lower frequency. The preferred time slot to perform all complex operations, which are those who implies shutting down the machine, is the weekend due to the lower intensity of the production (monthly and bi-weekly tasks are usually performed in this slot). For simplicity, the recurrency of these activities is expressed in weeks (e.g., a monthly task has a recurrency of 4 weeks);

- *One-time tasks*: are those kinds of activities that are performed only once and so do not present any kind of recurrency. These activities are usually extraordinary maintenance activities (e.g., machine breaks);

Depending on the type of assignment to an operator, tasks can be divided into:

- *Not assignable*: are all the tasks that do not require user assignment. Usually, they do not require specific skills and can be scheduled without caring about the operator qualification;
- *Assignable*: are all those tasks which needs to be assigned to a worker with a specific qualification. The scheduling algorithm needs to consider the availability of the required qualification in the scheduling process;
- *Assigned*: are all those tasks which have a worker already assigned by the Maintenance Manager. These tasks need to be scheduled considering the availability of the specific assigned operator;

4.2.2.3 Collected Requirements

This represents the requirements related to the general procedure inside the plant. They reflect the status of maintenance before the usage of any Decision Support System.

DEC.SCH.001

Type: It describes requirement related to the scheduling of periodic works. The scheduler must consider this aspect in order to organize the works distribution, based on their periodicity during the year.

Description: The system must be able to represent the suggestions in the best and clearest way possible. Through different views it will allow the Maintenance Manager to organize the execution of the needed maintenance operations. It is important for the Decision Support System to consider in a proper way the maintenance operation timing, identified in paragraph :

- *Recurring tasks*
  - *Daily*;
  - *Weekly, Bi-weekly, Monthly, Yearly*;
- *One-time tasks*;

Maintenance Manager needs to easily recollect the information related with the task's periodicity.

Frontend features: The scheduler comprises all the information about the task, separated in different views. All the views should be easily accessible, navigable, to represent the schedule calendar in the best possible way. The different tasks typologies must be highlighted to be identified in a clear and immediate way. The

preferred time for the execution of more complex operations (performed during the weekend) as well as the one-time operations, should be shown and highlighted in the schedule (different color).

A different view implies different characteristics:

- *Monthly*: allows a level of detail, due to the quantity of information this view can become crowded fast, so less information can be displayed;
- *Weekly*: this view allows a higher level of detail, so it allows a more complete display of information;
- *Daily*: this view has the highest level of detail due to the higher number of tasks to be displayed;

DEC.SCH.002

Type: It represents the constrains related to the assignment and scheduling of the works especially the difference between the fixed works and the one that can be suggested by the scheduler.

Description: The Decision Support System must consider in the proper way the tasks characteristics related with the categories described in paragraph 4.2.2.2.

In particular, the Decision Support System needs to accurately consider the task assignment:

- *Not assignable*: are all the tasks that do not require user assignment;
- *Assignable*: are all those tasks which needs to be assigned to a worker with a specific qualification;
- *Assigned*: are all those tasks which have a worker already assigned by the Maintenance Manager;

The different type of tasks should be displayed to enhance clarity and readability.

Frontend features: The task types and the related information must be displayed in a way in which they are easily identifiable and readable. For this reason, they are identified with different colors, shapes or borders depending on the type and periodicity. They can also present a different level of detail depending on the selected view, having an increasing level of detail moving from a general monthly view to a more specific weekly or daily view. If the task category is set on **assignable,** the Maintenance Manager has access to a suggested calendar where the maintenance activity is associated with a worker. If the task category is set on **not-assignable,** the scheduler suggests a time slot but not a worker. In this case, the Maintenance Manager can monitor the situation of the single maintenance activity and assign it to an operator.

DEC.OP.01

Type: It represents the decision support information that the Maintenance Manger expects to see about the operator.

*Description: T*he Maintenance Manager must receive from the Decision Support System the following information about the operator:

- *General info*: contains all the basic information about the operator;
- *Schedule*: shows the tasks that the operator must perform during its shift on a weekly and daily basis, following the same principles of the whole schedule view;
- *Status of current operation*: shows an indication of the progression of the task;

Frontend features: The Maintenance Manager can easily access all the information regarding the operators through the Operator View. This view shows the suggested operator activities, allowing to understand its workload for the following days/weeks.

In parallel the view shows the status of the current operation, for a more immediate planning of the maintenance activities and evaluate if the suggested plan is being respected. The operator view shows also the following general info about the operator for an easy access:

- *Name*;
- *Surname*;
- *Avatar/Icon*;
- *Qualification*: represents which operations the operator is able to perform;
- *Shift time*: represents the shift of the operator, from beginning to end;

DEC.MCH.01

Type: It represents the decision support information that the Maintenance Manger expects to see about the machines.

Description: The Maintenance Manager must receive from the Decision Support System the following information about the equipment:

- *Status*: The status represents what is happening to the machine, it can:
  - work correctly;
  - be under maintenance;
  - be in an error state;
  - be broken;
- *Schedule*: the maintenance activities that should be performed on the machine, when and in which conditions;
- *KPI*: additional indicators about the machine, e.g., the machine availability;

Frontend features: The Maintenance Manager has access to the machine related decision support information through the Machine View. This view aims to report and show all the most relevant info about the machine's activities and status, allowing the Maintenance Manager to better focus his attention on the single machine. This view is particularly complex because it has to show data arriving from multiple sources like the scheduler, the information system, the predictive maintenance, and the online quality control. The generic information about the machine, which comes from the Information System, is paired with the RUL (Remaining Useful Life) coming from the Condition Monitoring module. The Online Quality Control provides a real time status of the machines including:

- *Status messages*: which notify an unusual behavior of the machine to the Maintenance Manager;
- *Gallery of photo*: the photos taken by the vision system of the online quality;

DEC.WH.01

Type: It represents the decision support information that the Maintenance Manger expects to see about the management of the warehouse

Description: To correctly manage the warehouse, the Maintenance Manager must monitor the following information about spare parts and tools:

- *Indication of quantity*: the amount of common spare parts and tools available in the warehouse;
- *Minimum allowed capacity*: the threshold that triggers the reordering of material;
- *Restock suggestion*: a suggested date in which the Maintenance Manager should perform the reordering of the parts and tools, and the needed quantity;
- *KPI*: indicators about the overall warehouse status, the general availability of spare parts and tools and the optimization of their usage;

Frontend features: The Maintenance Manager can monitor and obtain decision support information from the Warehouse View. This view shows the expected level of stock for each spare parts or tools through different widgets which can highlight the expected level with a different color code. In addition, a series of support information is provided to the Maintenance Manager, for example:

- *Restock suggestion*: to be accomplished to fulfil the foreseen maintenance activities;
- Spare parts which are closer to be out-of-stock;

With this view the Maintenance Manager can keep track of all the spare parts and tools present in the plant and promptly react to future needs.

4.3 Component's interactions design

In this section it is analyzed the way in which the different modules of which interacts with the DSM exchange information.

4.3.1 Backend interaction

The interaction between frontend and backend is the core channel on which information are exchanged, it allows the correct functioning of the DSM and allows the frontend to access the algorithm which elaborates on the data.

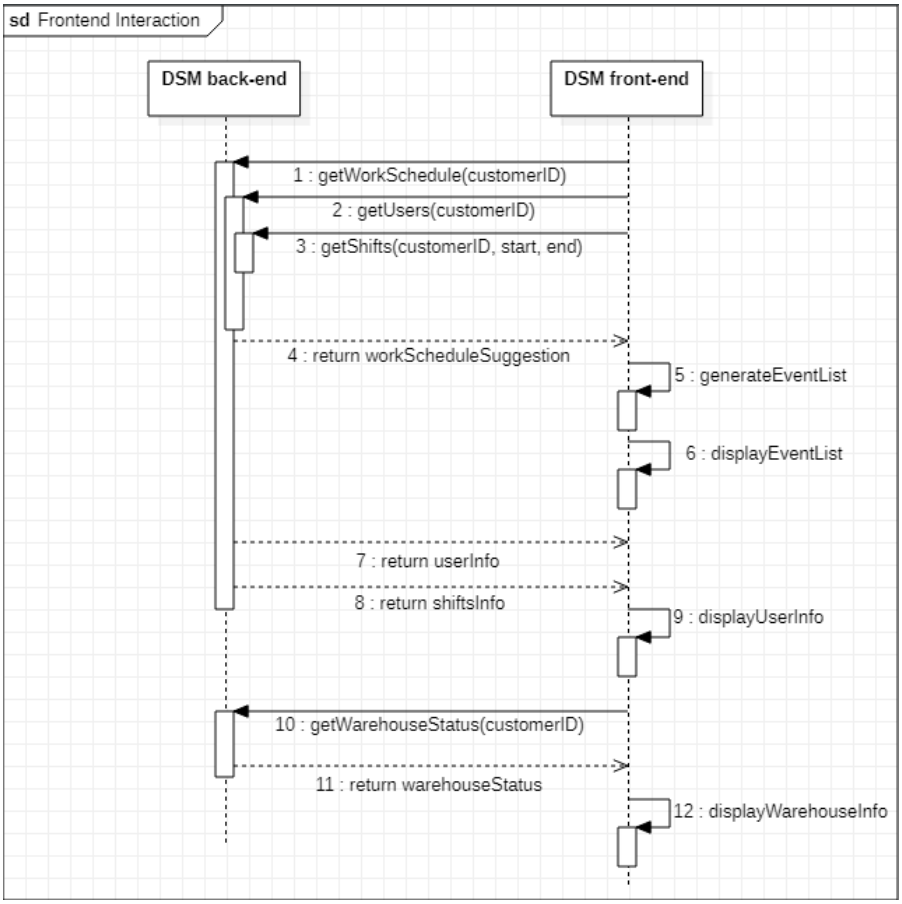


Figure 16.1: frontend-backend interaction

The Interaction between Frontend and Backend inside the DSM happens through the utilization of different GET APIs. The data acquiring process happens as follows:

1. The Frontend asks to the Backend the schedule generated by the compute API (5.5.10/compute) through the WorkSchedule API (5.5.8/work-schedule);

2. The Frontend asks to the Backend the users' info provided by the information system and held into our backend through the users API (5.5.1/users);
3. The Frontend asks to the Backend the shifts' info provided by the information system and held into our backend through the shift-schedule API (5.5.2/shifts-schedule);
4. The Backend returns the last generated schedule to the Frontend.
5. The Frontend reads the received Json file and converts it into a list of events which can be later displayed on the calendar;
6. The Frontend displays the obtained schedule inside the scheduler component (5.3.1.2 Scheduler);
7. The Backend return the users' info;
8. The Backend returns the shifts' info;
9. The Frontend displays the obtained users' info inside the operator component (5.3.1.4 Operators), the Frontend will also display the obtained shifts' info inside the operator component (5.3.1.4 Operators) within the dedicated space in the calendar displayed in the single operator view;
10. The Frontend asks for the data regarding the warehouse, which will provide both the status and the restock prediction, to the Backed;
11. The Backend returns the warehouse information;
12. The Frontend displays the information obtained inside the warehouse component (5.3.1.3 Warehouse);

4.3.2 External sources interaction

In this section are shown the interactions between the DSM backend and all the three different external sources from which it gathers the data that are then analyzed and elaborated by the decision algorithms.

4.3.2.1 Information system

The Information system is the main source of information for what regards the plant's general info, it provides: operators, shifts, works etc.

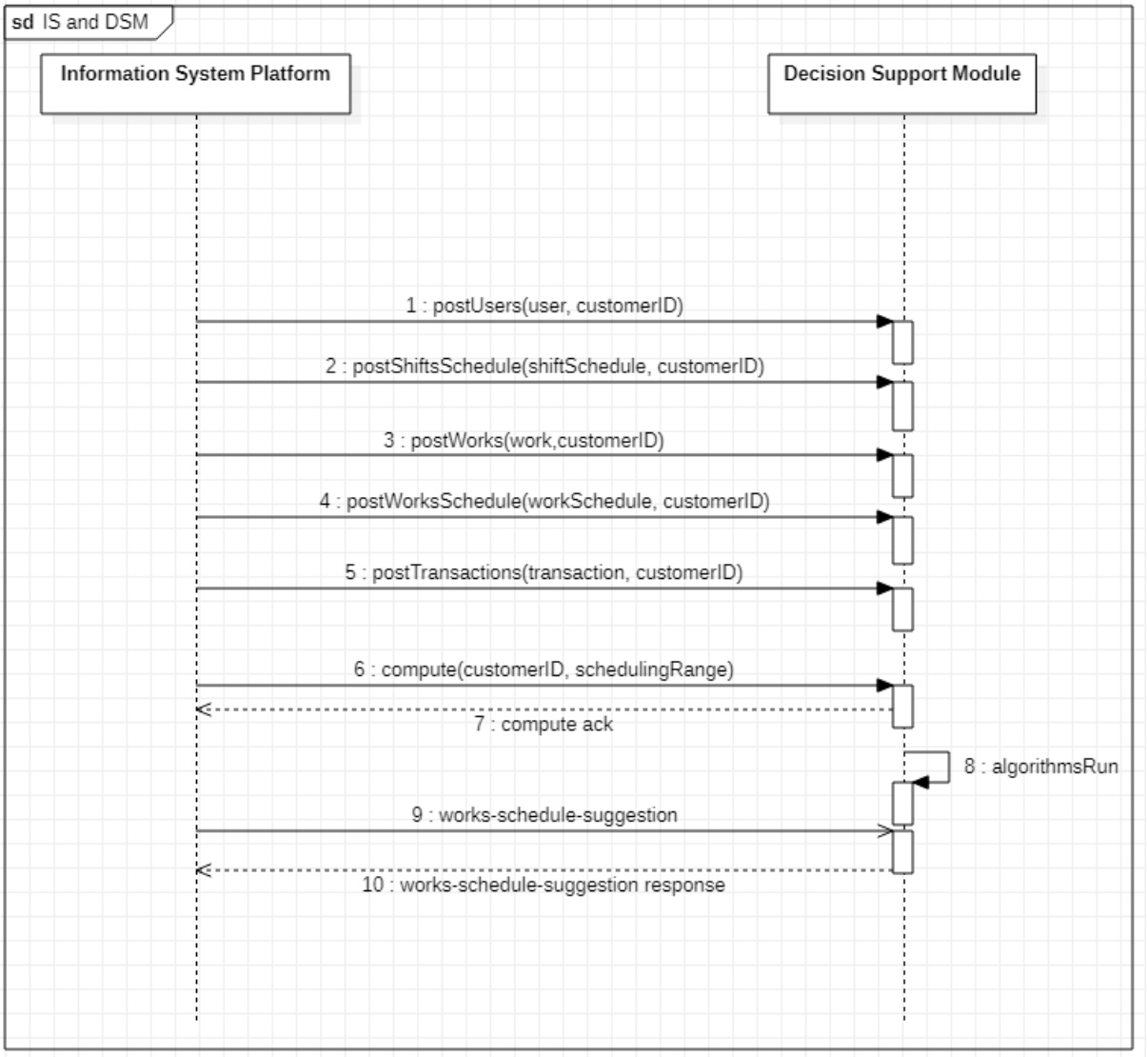


Figure 17.1: DSM backend - Information system interaction sequence diagram

The interaction is as follows:

The IS sends the data required by the DSM to perform its suggestions (the order is not important). The data is related to a customerID, to identify the customer for the computation of the schedule:

1. The IS send to the DSM backend the data about the users through the postUsers API. The user is identified by an ID that can be used in other API calls to refer to him (for API structure look at paragraph 5.5.1/users);
2. The IS sends to the DSM backend the data about the postShiftSchedule containing the shifts planning related to the users (for API data structure look at paragraph (5.5.2/shifts-schedule);
3. The IS sends to the DSM backend the data about the works to be scheduled with the post Works API (for API structure look at paragraph 5.5.5/works);
4. The IS sends to the DSM backend the data about the works scheduled (executed or not), the suggested schedules are not sent. The works only scheduled are used by the scheduler to calculate the resources already used. The already executed works are ignored in the scheduling process and their last execution is used for reference in the recurring works. The data is sent using the postWorks-schedule API (for API structure look at paragraph 5.5.8/work-schedule);
5. The IS sends to the DSM backend the data about the warehouse transactions. The scheduler uses this data to calculate the availability of spare parts and tools (for API structure look at paragraph 5.5.9/transactions);
6. When the IS sent all its updates it can trigger the suggestions computation:
7. The IS triggers the computation of the schedule through the compute API (for API structure look at paragraph 5.5.10/compute) providing the scheduling time range;
8. The /compute API sends back a reply. The response to the API call does not provide the algorithm result but the acknowledgement of the computational request (200 OK, request enqueued);
9. The DSM runs the scheduling algorithm with the Data provided by the IS;
10. The IS can ping the /work-schedule-suggestion API (defined at paragraph 5.5.7) to ask for the schedule results;
11. The response to the /work-schedule-suggestion API returns the last schedule execution result. The createdAt value may be checked to be use that the returned schedule is the last requested one;

A special mention has to be done for the *warehouse* decisions support, which is considered a key aspect of the DSM.

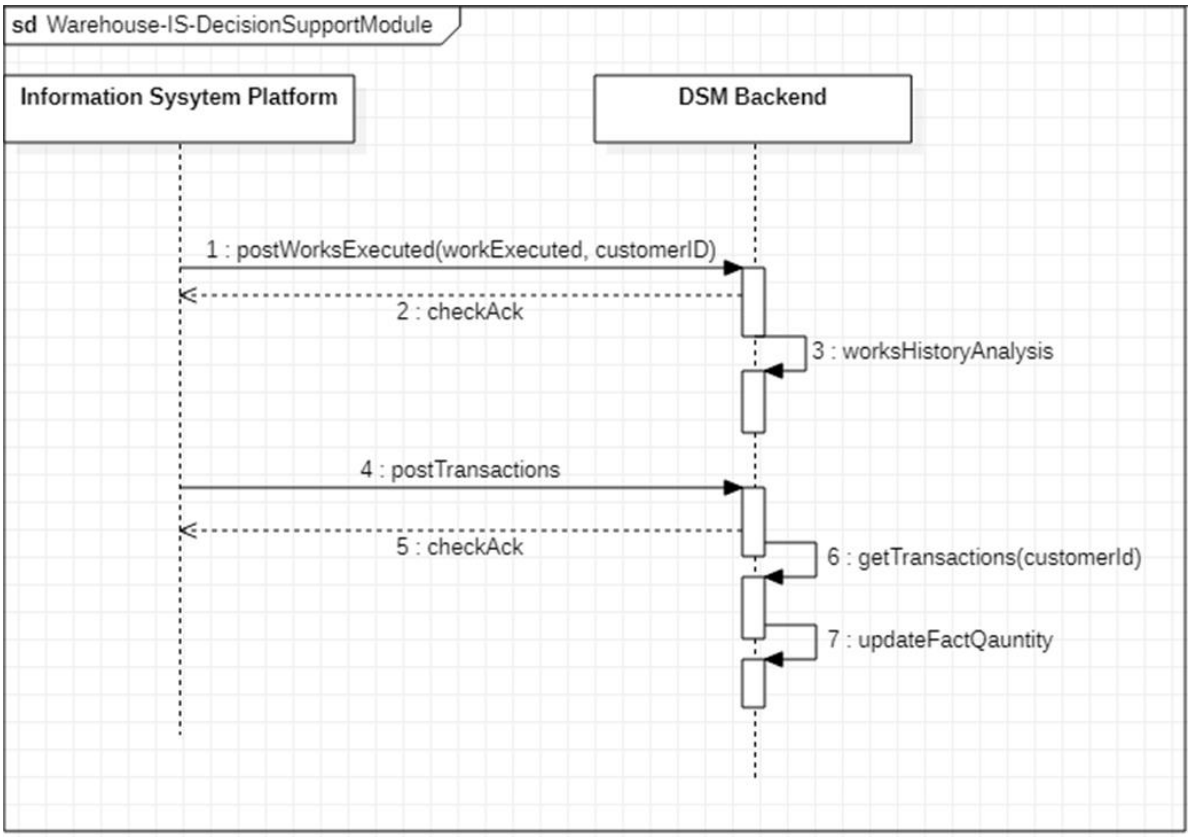


Figure 18.1: DSM- Information System Warehouse interaction sequencediagram

The interaction is as follows:

1. The IS posts the information about the executed works through the postWorks-schedule API (for API structure look at paragraph 5.5.8/work-schedule);
2. The DSM answers with a checkAck;
3. The DSM analyses the history of the executed works to compute the usage of spare parts and tools;
4. Once a Work is completed the IS sends the transaction of the executed work, adding the fact quantity of tools and parts utilization, and it sends it to the DSM;
5. The DSM answers with a checkAck
6. The scheduler gets the updated transactions from the DSM backend Database;
7. The DSM computes and updates the new Warehouse status using the new information about the completed transactions;

4.3.2.2 *Condition monitoring*

In this section it is analyzed the design of the interaction between the *DSM* and the *Condition Monitoring Module.*

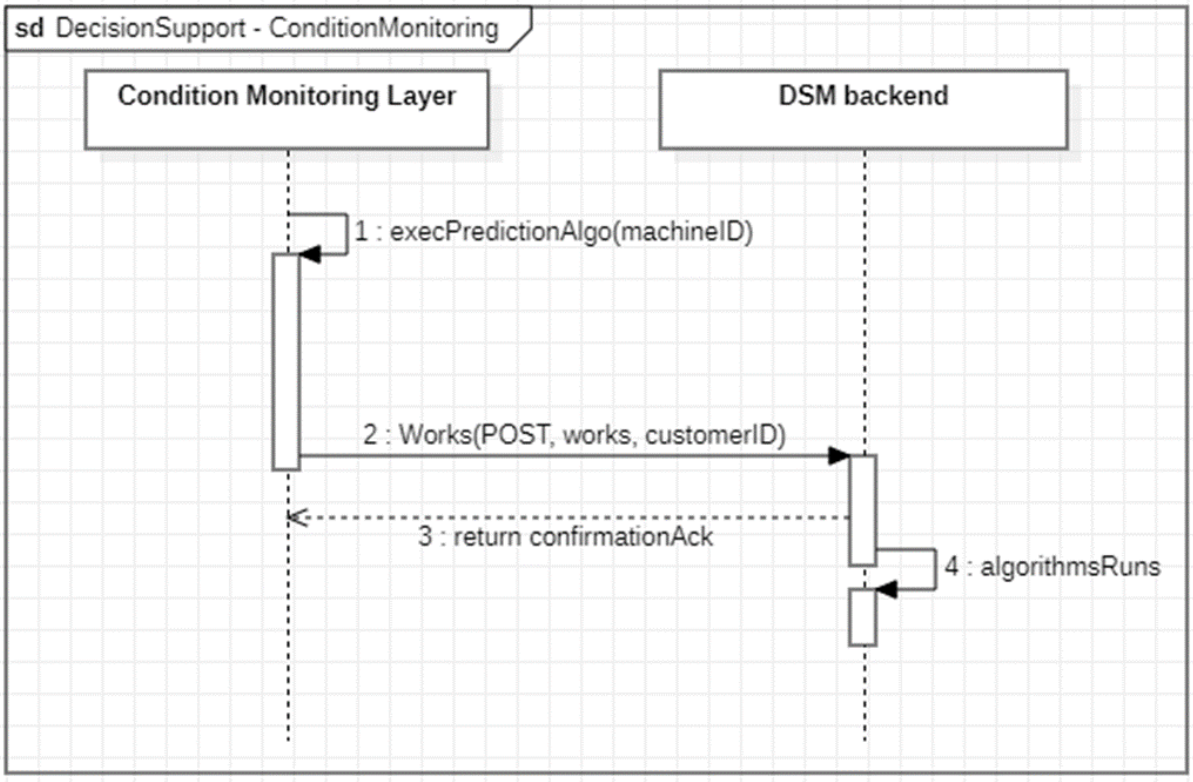


Figure 19.1: Condition Monioring Module and DSM interaction sequence diagram

The interaction is as follows:

1. The Condition Monitoring System sends to the Decision Support System the RUL (Remaining Useful Life) of the machines present in the plan through the /works API described at paragraph 5.5.5/works. An example of API call from the CSM is provided at paragraph 5.6.1:
2. When the sending of all the predictive maintenance elaborations is completed, the Condition Monitoring System sends to the Decision Support System a compute request through the /compute API defined at paragraph 5.5.10;
3. The Decision Support System provides an acknowledge of the received request;
4. The Decision Support System algorithms run and generate all the predictions and support information. The results can be visualized through the Decision Support System front-end;

4.3.2.3 *Online Quality Control*

The designed interaction between the DSM and the Online Quality Control Module is displayed in the subsequent sequence diagram (Figure 20.1).

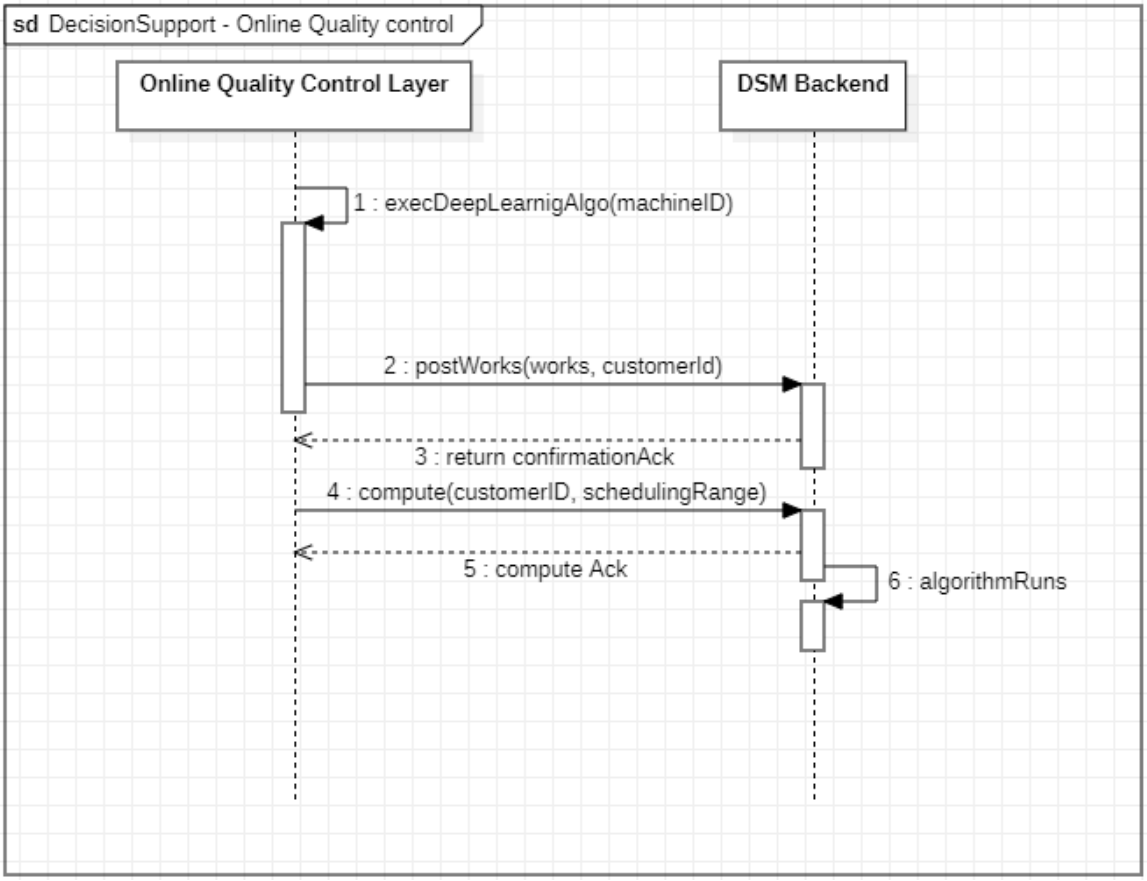


Figure 20.1: DSM and Online Quality Control Interaction sequence diagram

The interaction is as follows:

1. The Online Quality Control Layer will execute the Deep Learning Algorithm, which will dynamically monitor the state of the components produced by the machined, and it will highlight eventual problems by creating a Work;
2. The OQC sends the works to the DSM backend through the postWorks API (5.7.1), delivering the set of all works aligned with the cutomerID;
3. The DSM Backend sends a check to the OQC to acknowledge the reception of the message;
4. The OQC triggers the execution of the scheduling algorithm through the compute API (5.5.10/compute) which will provide to the scheduling

algorithm both the customerID and the schedulingRange for which it is necessary to calculate the schedule;

5. The DSM backend sends a confirmation ack to notify the triggering of the algorithm;
6. The DSM executes the decision support algorithms taking into account the new works send by the OQC;

4.4 Conclusions

In this section is analyzed the designed process which brought to the subsequent implementation of the DSM. It shows the high-level architecture on which the system has been based (4.2.1) showing then the interaction between the different components, and how they are designed to interact between each other displayed through the sequence diagrams.

5. Chapter 5

In this section it is going to be analyzed the actual implementation of the Decision support system, regarding the structure, technologies and methods used to create the final system.

5.1 Definitive System Architecture

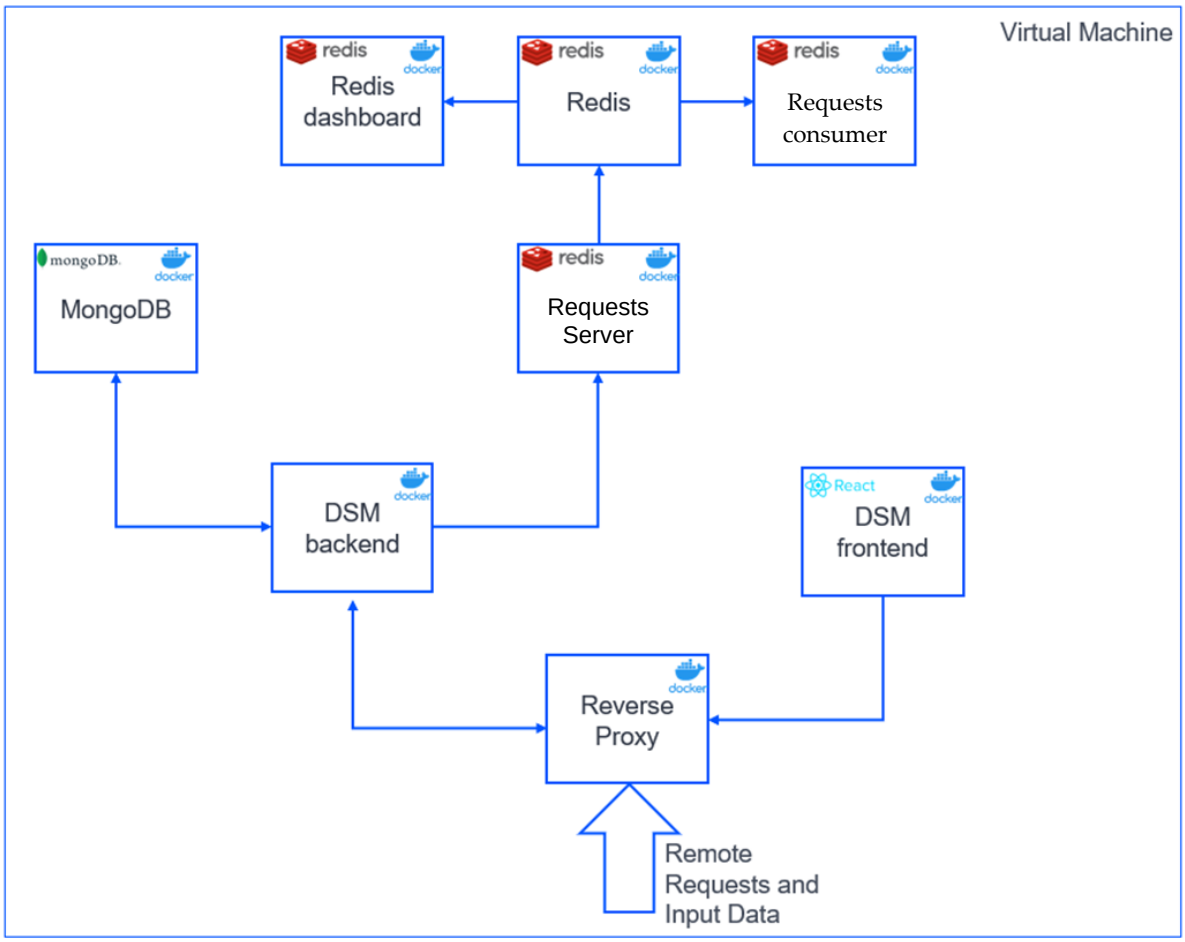


Figure 21.1: Detailed DSM architechture and containerization

The DSM is based on Docker, and it is divided into different containers, one for each component, as follows:

- *Reverse Proxy*: oversees all the incoming APIs and general requests routing them either to the Frontend or the Backend;

- *DSM Frontend*: encapsulates the frontend application, and it is the main interaction container, the one which the final user will interact the most with;
- *DSM Backend*: implements the APIs and manages the requests between the Mongo and the Request Sever;
- *MongoDB*: encapsulates the MongoDB;
- *Redis*: encapsulates Redis and all its services;
- *Redis Dashboard*: allows to see and monitor the status of the queues and the jobs;
- *Requests consumer*: encapsulates the Python algorithms made for the computation of the suggestions;
- *Requests Server*: exposes a private HTTP server to accept the computation requests for a given customer from the DSM backend;

The choice of using Docker for the implementation and distribution of the DSM is driven by the possibilities offered by Docker. It allows to instantiate the DSM onto any system which is capable of running docker, making the DSM itself hardware and OS independent. Modularity is another characteristic which made Docker the better solution for the needs of this project, it allows to quickly update containers without any particular restriction, allowing to change and customize the architecture, based on the changing needs of the project. Since Docker is hardware independent and based on distributable container, the solution can be executed on premise or on any remote device (virtual machine). In addition, its modularity makes it possible to minimize the hosting machine resources and increasing them when needed, without wasting resources during low usage.

5.1.1 Functioning Logic

The DSM expects the trigger of an external source in order to execute the computation of the Data. This trigger is sent through the API 5.5.10/compute which allows the system to start calculating all the predictions, basing his process on the data currently available in the system. The request and computation system are based on queues: there is a producer who inserts a request inside a queue (when the 5.5.10/compute API is used), then one consumer, listening to the common queue, begins the execution of the algorithms as soon as a previous request completes. The system is structured in this way in order to support load spikes which could potentially cause denial of service. There is a queue for each customer based on the customerID to allow parallel execution between different customers and avoid colliding requests. The presented solution based on Docker containers is deployed on an AWS virtual machine, thanks to the service reliability and popularity.

5.2 DSM Backend

The Backend container is the component where the DSM APIs are implemented, so it has the duty of managing the requests, incoming both from the Frontend and the Remote Resources. It will also manage the internal interaction between the Reasoner Server to schedule computation requests and the Mongo to return or store information according to the API request. The APIs are developed with Express.js [15] which is a framework made for the development of web application in Node.js [16], it is the most used framework for the creation of web applications based on Node and JavaScript because it allows to produce small volumes of optimized and readable code.

5.2.1 Mongo DB

The Mongo Container is based on MongoDB [17], a popular NoSQL database offering storage service. In the project context it is used to store the data involved in the project and return it according to the APIs requests.

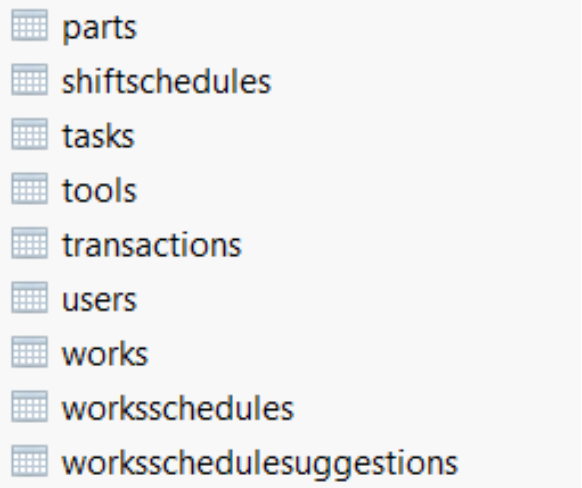


Figure 22.1: MongoDB Collections

MongoDB has been chosen thanks to its capability, being a NoSQL database, to easily upload entire JSON files inside its collections. The interaction between the backend and the database is granted by Mongoose [18], an ODM ("Object Data Model") which allows to develop the database with an object-oriented approach, directly working with collections instead of queries and delegating to mongoose the duty to convert high level commands into queries and vice versa.

5.2.2 Requests Server

The server manages the incoming calls and puts all the requests inside a waiting status until the execution is considered feasible. It is based on Flask [19], which is a framework for the creation of web servers in Python. On the Server there is a set of queues divided per customerID in order to reserve a different queue for each customer; this avoids problems related to conflicting requests and congestion.

5.2.3 Redis

Redis [20] is an open source, in-memory data structure storage, used as a database, cache, and message broker. Redis provides a variegated set of data structures (strings, lists, arrays, etc.). Redis has built-in replication, Lua scripting, LRU eviction, transactions, and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster. It allows to run a variety of atomic operations like:

- Appending strings;
- Push an element into a list;
- Compute intersections, unions, and differences;
- Perform ranks and sorts;

To achieve top performance, Redis works with an in-memory dataset. Depending on the specific use case, you can persist your data either by periodically dumping the dataset to disk or by appending each command to a disk-based log. You can also disable persistence if you just need a feature-rich, networked, in-memory cache. The Redis container manages the queues, taking a job from the server and serving it to the consumer. It delivers also the data to the Redis dashboard. Both the Server and Consumer are based on PythonRQ [21], which is a library backed by Redis, and made for jobs and works management using Python.

5.2.4 Requests Consumer

The consumer takes care of executing the jobs stacked in the queue shared with the producer. As for the queues, we will have a different consumer for each customer: this allows to execute jobs simultaneously for different customers, allowing to exploit parallelism and shortening the waiting time. Depending on the customers number, which could increase while time goes by, the computation requires additional resources to the hosting machine to complete the jobs. In such a case, to mitigate resources starvation problem, it is possible to increase the performance specifications initially adopted for the remote machine where the solution runs.

5.2.5 Redis Dashboard

The Redis Dashboard is a web-based application which monitors the status of the job execution and the queues. The information is gathered from the Redis container which manages the jobs between the available queues. Using the web interface, it is also possible to manually empty the execution queues, imposing an "hard reset" of the executions already planned but not computed yet.

5.2.6 Reverse Proxy

The Reverse proxy has the duty of managing all the incoming requests, it has the duty of discerning if the request is directed to get the frontend static content or if it is a call to an API exposed by the backed. The call to the backend can come either from an internal or external resource. Internal requests are those coming from the Frontend to the Backend in order to display all the decision support information generated by the algorithm. External requests are those coming from Condition Monitoring, Information System and Online Quality Control. They are usually post requests that upload Data inside the MongoDB, but they can also be computation requests which will trigger the execution of the algorithm.

5.3 DSM Frontend

The frontend has the goal of displaying the information in the clearest and most efficient way possible. To achieve this objective, it will take advantage of React [22], a JavaScript library, which allows the creation of a flexible and responsive interface, with broad compatibility with all browsers. React allows the webpage to quickly, and in real time update itself, displaying all the most recent information found in the database. This is done thanks to the DOM management of React which, when notices a change, quickly re-renders the page to display its latest version. It is used Material Ui to take advantage of its components, made appositely for React, which offer multiple functionalities paired with a more complex and structured graphical overhaul. On part with speed and responsiveness, the focus should be put on the user experience. It is important to display all the information in the clearest way possible. Consequently, the frontend takes advantage of the routing capabilities of react to display the information into different pages better organizing and displaying all the data provided by the different sources. The implementation of the different views can be seen in the 5.3.1 Components' mockup and description paragraph which displays the implemented views and explains the graphical and interaction details

5.3.1 Components' mockup and description

In this section are presented all the views of the DSM frontend, with a complete description of their features, elements, and principles behind their development.

5.3.1.1 Dashboard

The Dashboard view represents the landing page of the DSM, it incorporates the most important KPIs of the Decision Support System to help the Maintenance Manager to easily monitor the overall situation.

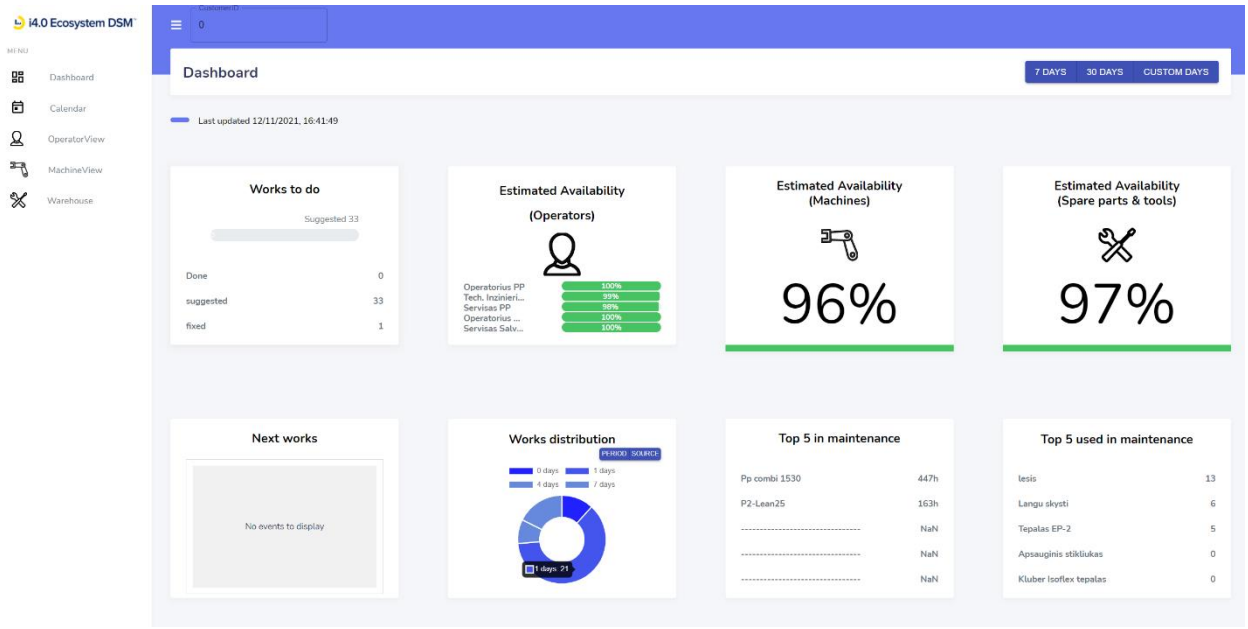


Figure 23.1: Dashboard view mockup

The Dashboard view, showed in Figure 23.1, is composed of the following elements:

- *The amount of works to be performed*: shows the total number of works suggested by the Scheduling Algorithm, paired with a progression bar which will show the amount of completed works over the suggested ones, and the list of works divided in:
  - *Done;*
  - *Postponed;*
  - *Suggested;*
- *The estimated availability of the operators*: shows the expected availability of the operators depending on the qualifications present in the different shifts and maintenance activities suggested by the Decision Support Algorithm;

- *The estimated availability of spare parts and tools*: shows the percentage of expected spare parts and tools still available in the warehouse considering the maintenance activities suggested by the Decision Support Algorithm;
- *Next works*: the list of works (suggested by the algorithm or fixed by the user) to be performed in the near future;
- *Works distribution*: shows the amount of works (suggested by the algorithm or fixed by the user) distinguished by their type;
- *Top 5 in maintenance*: shows the top 5 machines that are expected to be under maintenance;
- *Top 5 most used*: shows the spare parts and tools that are mostly used in the forecasted activities;

5.3.1.2 Scheduler

The Scheduler represents one of the core functionalities of the DSM, it will display the results of the computations made by the enhanced scheduling algorithm.

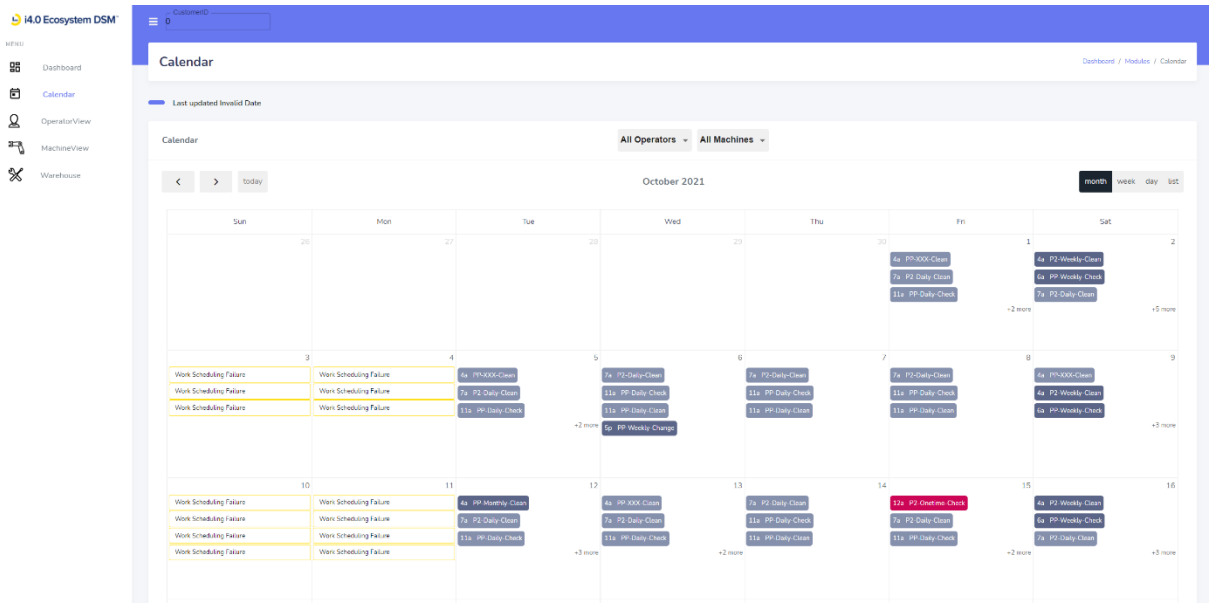


Figure 24.1: Calendar weekly view mockup

In the scheduler view, showed in Figure 24.1, the works are represented following the characteristics defined in paragraph Solution requirements by using different colors and icons:

- *Recurring Works*: characterized by a circling arrow icon and different colors depending on the periodicity
  - *Daily*: characterized by a light grey coloration (#8590AD);

- *Weekly*: characterized by a grey coloration (#5B6587) and a pair of Icon-Information related to the machine and the operator who executes the work;
- *Monthly*: characterized by a dark grey coloration (#454D60) and a pair of Icon-Information related to the machine and the operator who executes the work;

- *One-Time Works*: this type of work can derive from different sources:
  - *Information System*: works inserted directly by the Maintenance Manager;
  - *Condition Monitoring*: works generated by the predictive maintenance layer;
  - *Online Quality Control*: works generated by the vision system placed on the machines;

  These works are characterized by a different level of urgency and a pair of Icon-Information related to the machine and the operator who executes the work, the urgency is represented through different colors:

  - *Red (#CC0556)*: for high urgency works;
  - *Yellow (#F8A300)*: for average urgency works;
  - *Green (#16DB85)*: for low urgency works;
- *Fixed Works*: these works are the ones which are verified and validated by the Maintenance Manager, their color is set to blue (#003193);

Figure 25.1: Generic event detail

- *Work scheduling failures*: which is a special type of event which occurs whenever the scheduling algorithm was unable to schedule a work, it displays a table showing the possible failures for each hour (Figure 26.1);

Figure 26.1: Work scheduling failure detail

This view allows the Maintenance Manager to filter the works based on different filters:

- Operator Filter: which allows the Maintenance Manager to filter the view selecting a single operator;

Figure 27.1: Operator filter

- Machine Filter: which allows the Maintenance Manager to filter the view selecting a single machine;

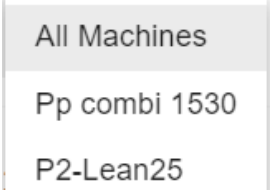


Figure 28.1: Machine filter

5.3.1.3 *Warehouse*

The Warehouse component aims to provide decision support information to the Maintenance Manager to correctly handle the warehouse material considering the expected maintenance activities. For this reason, the Warehouse View contains the dynamic suggestions of the possible restock dates of spare parts and tools and tracks the status of the warehouse.

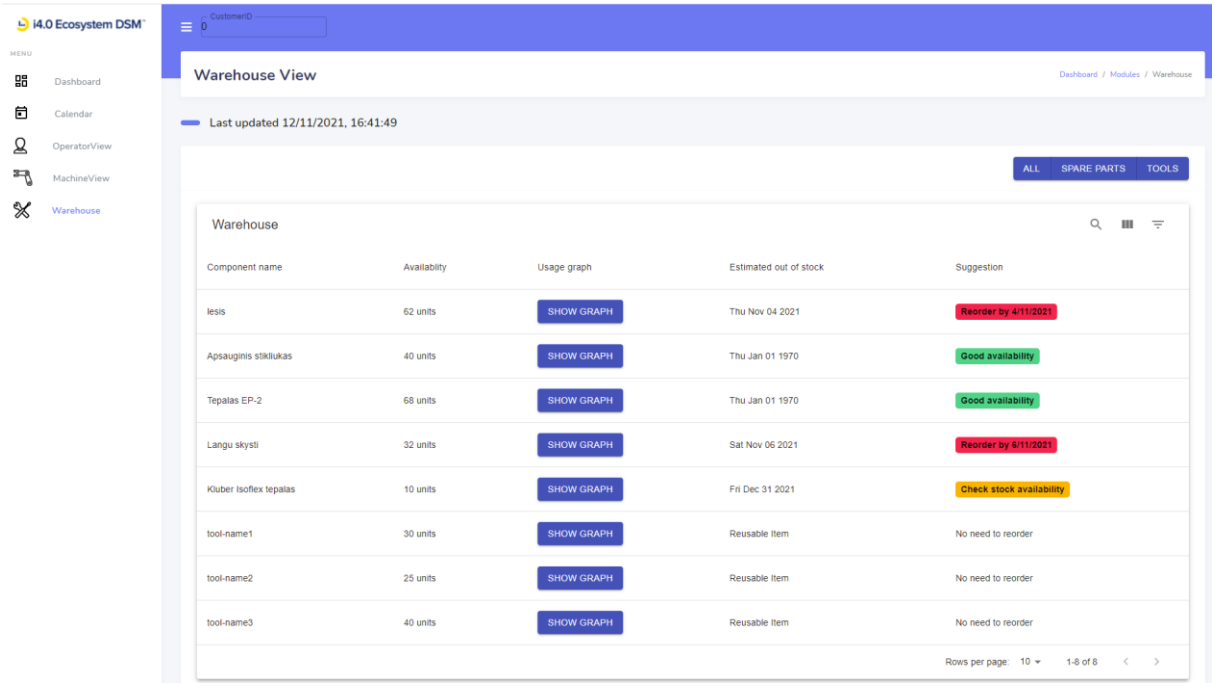


Figure 29.1: Warehouse component view

The Warehouse View (in Figure 29.1) provides the following information to the Maintenance Manager:

- Component Name: the identification name of the spare part/tool;
- Availability: shows the quantity available in the warehouse in that moment;
- Usage Graph: it allows, by clicking on show graph, to have a deeper look on the consumption trend for the specific part or tool, the graph will be shown on a separate component which will pop over the table (Figure 30.1);

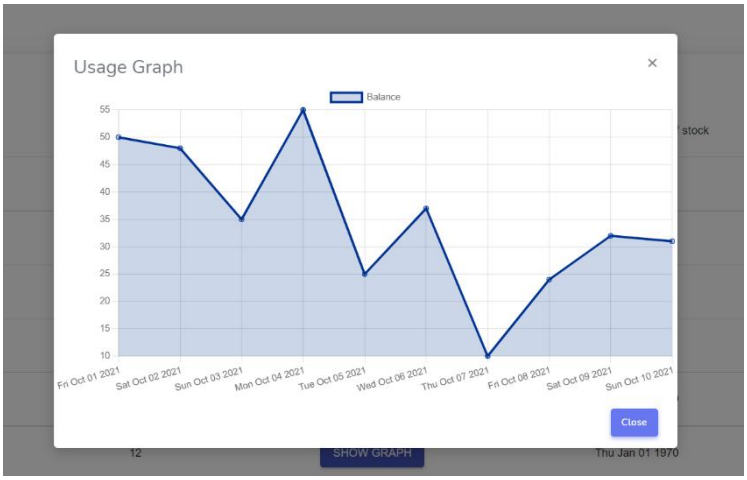


Figure 30.1: Usage graph

- Estimated out of stock: it shows the date in which the spare part/tool will run out, depending on the transactions provided by the information system;
- Suggestion: it shows a suggested action to the Maintenance manager about a specific spare part/tool, e.g. a reordering action because potentially the part may run out of stock;

The table allows to order the warehouse items depending on their category and to apply the following filters:

- Spare parts;
- Tools;
- Both;

5.3.1.4 *Operators*

The Operator View allows the Maintenance Manager to monitor the operators and their suggested activities with different degrees of details.

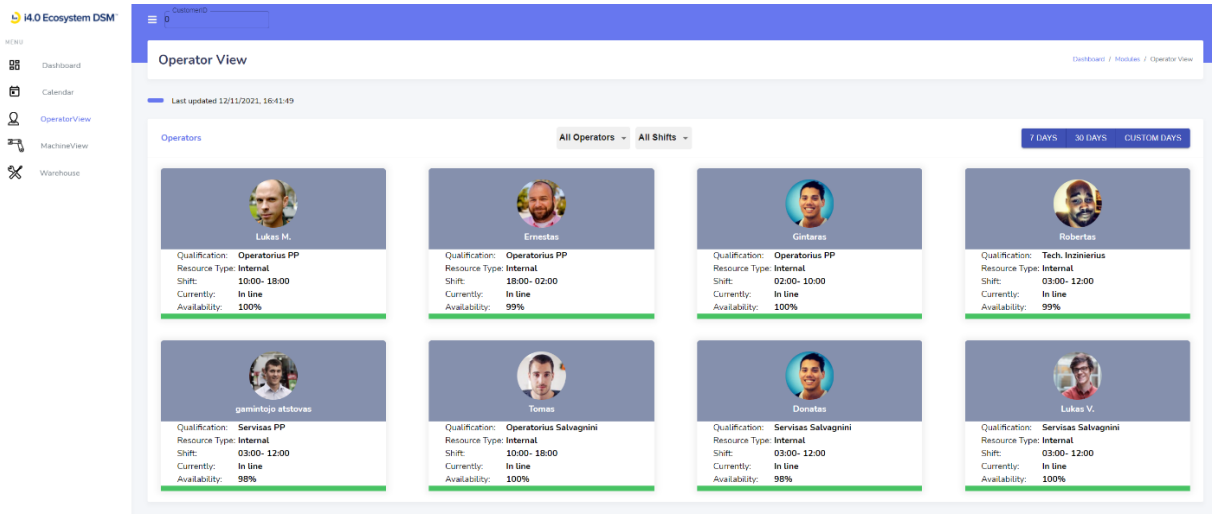


Figure 31.1: Operators view mockup

In the Operator View (in Figure 31.1) the Maintenance Manager has the possibility to look at all the operators in the plant and to monitor their situation also considering a forecast of their expected activities. It will display a list of operators with:

- Avatar: identification photo of the operator
- Name: identification name of the operator
- Qualification: the list of qualifications associated with the operator
- Resource Type: it states if the operator is internal or external
- Shift: shows the shift currently assigned to the operator
- Currently: shows the current status of the operator (busy, free, etc)
- Availability: shows the operator expected availability in the selected time frame, it follows the following pattern:
    - Red: if the operator is busy most of the selected period (> 70% of expected activities)
    - Orange: if the free time of the operator covers nearly half of the selected period (between 30% and 70% of expected activities)
    - Green: if the operator is free for most of the selected period (< 30% of expected activities)

The view allows to filter the operators following the same principles of the Scheduler View (5.3.1.2 Scheduler).

By clicking on an operator card the detailed operator view is shown, which displays all the details about a single operator.

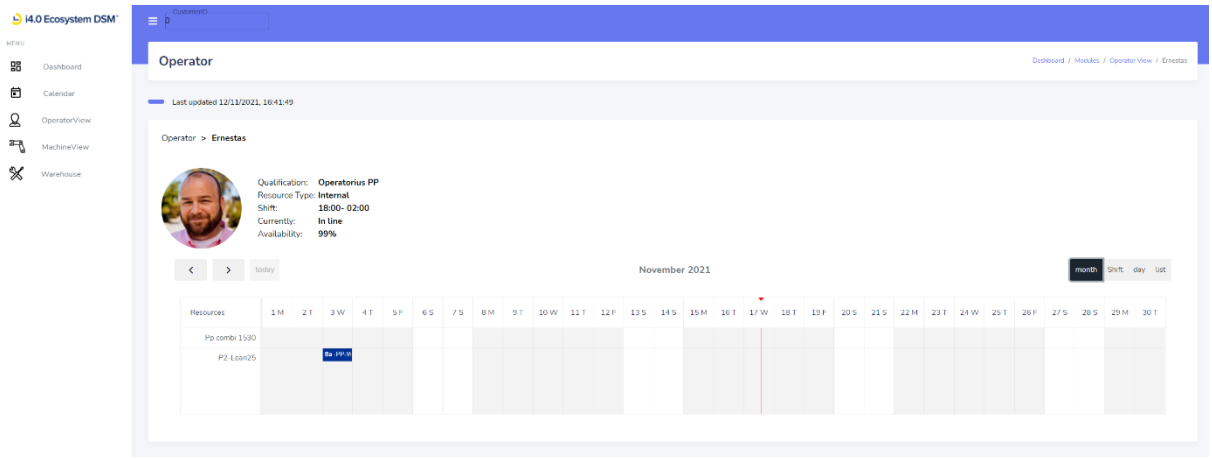


Figure 32.1: Operator's detail view mockup

This part of the view (showed in Figure 32.1) aims to show a detailed snapshot of the operator's status, it comprises all the elements present in the previous view with some additional information. In particular, a different type of calendar is displayed to show all the works related to the selected operator and divided by machine. A set of additional information related to the operator is showed in the upper part to help the Maintenance Manager to understand the operator availability and his qualifications.

5.3.1.5 Equipment

The Equipment View allows the Maintenance Manager to monitor the machines present in the plant with different degrees of details.

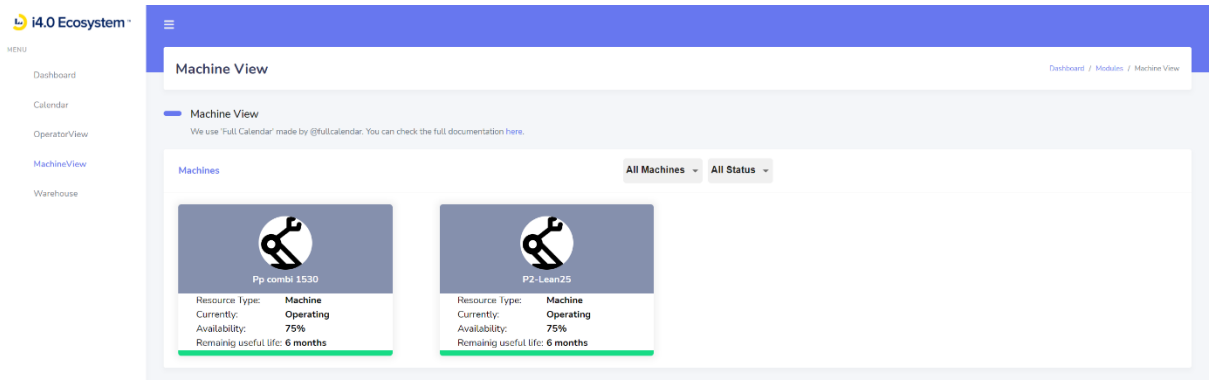


Figure 33.1: Equipment view mockup

With this view (in Figure 33.1) the Maintenance Manager can see all the machines present in the plant and monitor their expected situation through a variety of information:

- *Resource Type*: it specifies type of tasks that can be performed by the machine
- *Currently*: it shows the current status of the machine (if under maintenance, working, etc.)
- *Availability*: it shows the expected availability of the machine considering the forecast on possible maintenance activities, depending on the availability the following colour code is used:
    a. *Red*: if the machine is busy most of the selected period (> 70% of expected activities)
    b. *Orange*: if the free time of the machine covers nearly half of the selected period (between 30% and 70% of expected activities)
    c. *Green*: if the machine is free for most of the selected period (< 30% of expected activities)
- *Remaining useful life*: it specifies the time occurring between now and the next predicted breakdown (if this information is provided by the Condition Monitoring System)

By clicking on one of the machine boxes is possible to navigate into a more detailed view of the single machine

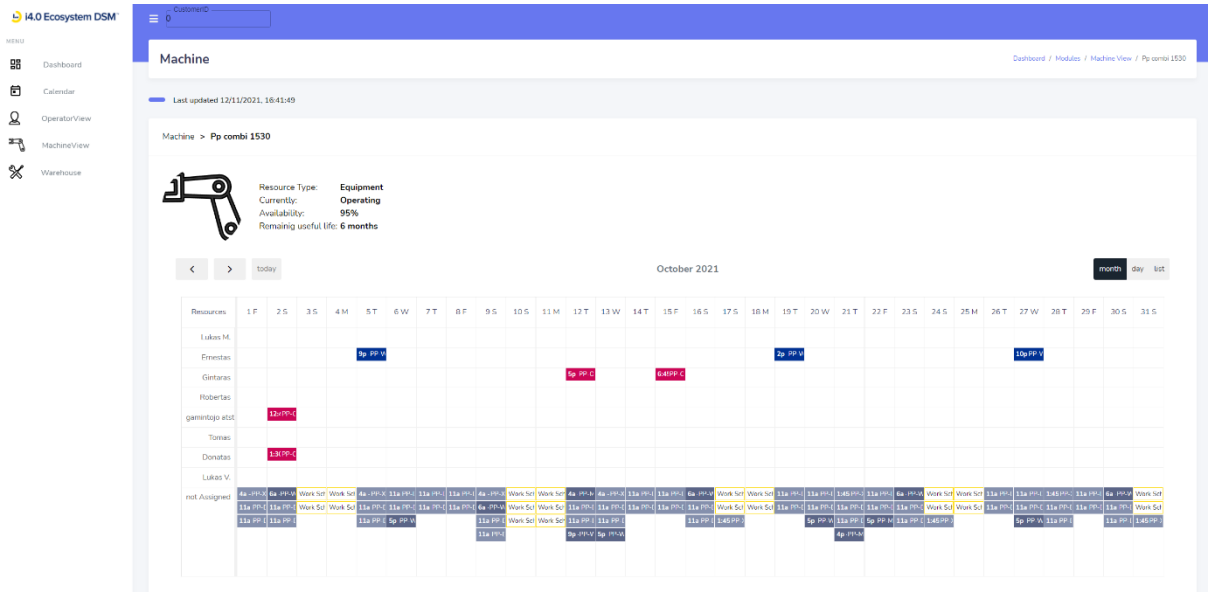


Figure 34.1: Machine detail view mockup

Inside this view (in Figure 34.1) it is possible to have a higher level of detail about the single machine. In particular, the view shows useful data about the expected

maintenance activities inside of the calendar. In addition, the Maintenance Manager can navigate to the operator assigned to the activity or evaluate the average expected availability of the machine in the selected period.

5.3.2 Technologies adopted

As already mentioned for the implementation of the frontend it has been chosen React. More precisely depending on the view a variety of different widgets and plugins has been used. The general graphical impact has been mased on the widget provided by material-ui [23], which provided the styling for all the buttons, and the modals. The Usage graphs present in the warehouse has been plotted through the utilization of Chart.js [24], in particular it has been used a wrapper made appositely for React called react-chartjs-2 [25]. It allows to create personalized charts based on structurally strict dataset, which can be dynamically created and rendered. It has been used also for the drought chart in the dashboard view. For the tables present in the frontend it has been used MUI-Datatables [26] which is a wrapper for React of the basic material-ui datatable. It allows a real time allocation of the data with dynamically defined columns and content; it has been used in the warehouse and in the work scheduling failure detail (Figure 26.1). Finally the calendar has been provided by FullCalendar [27] a framework which allows the rendering of a calendar, with a lor of pre-existent features:

- *Time location;*
- *Possibility to change timeframe;*
- *Date management;*

Keeping a good degree of personalization, allowing to fully personalize the way in which events are displayed and how the user can interact with them. It allows also to use different type of calendar:

- *Daygrid*: used for the main scheduler it shows the events on a grid either by Month, week or day;
- *Timeline*: which allows to show the timeline with the events, crossed with a personalized set of resources (used for the detail of the operators and machines Figure 34.1)

5.3.3 Frontend Navigation

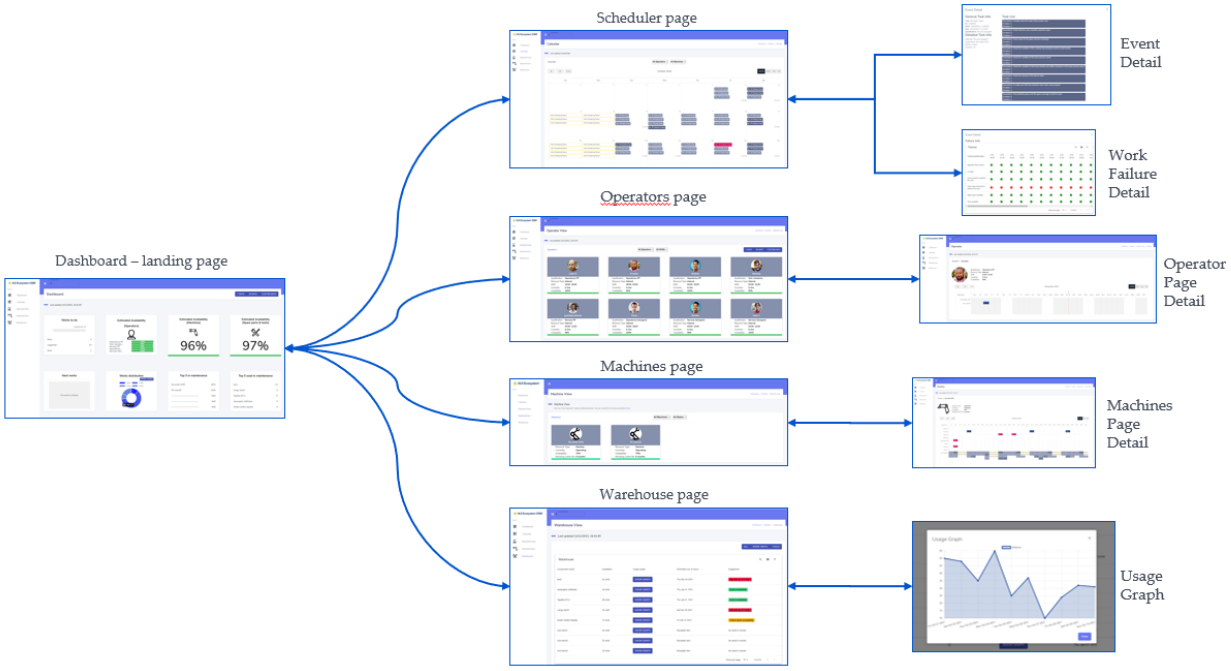


Figure 35.1: Frontend view navigation

The DSM frontend has been designed to be fast to use and easy to navigate. Once the application is opened the user is presented with the dashboard component (Dashboard), which displays the main KPI and information, giving immediate feedback on the main possible issues present in the plant (not completed works, operators' availability etc.). from the dashboard is possible to navigate to four different views:

- *Scheduler (*Scheduler*):* which displays the generated schedule and allows to interact with the events and the timeframe to be displayed. From here the user is able to open a detailed page by clicking on any event, this is done without redirecting to another page, but by opening a modal widget over the calendar, this widget can shoe either the:
  - *Work assignment failure detail;*
  - *Event detail;*
- *Operator view (*Operators*):* which shows the list of all the operators of the plant. From here the user can navigate to a detailed view of the specific selected operator;
- *Machine view (*Equipment*):* which behaves in the same way as the operator view, but focused on the machine present inside the plant;

- *Warehouse:* which shows the list of all spare parts and tools and allows to navigate to a graph of the utilization of each element, in the same way as the details of the event on the calendar;

5.4 Decision Algorithms

In this section are explained the features of the decision algorithms, it is not possible to display any form of code because is protected by an NDA agreement with EIT Manufacturing.

5.4.1 Algorithms' Structure

The decision algorithms are based on python and are structured in different objects, two for each element (works, operators etc.) one class is for the definition of the attributes of the object itself, the other one defines the ways in which the instance of the object interacts with others. The decision process starts by retrieving the Json files of each external source from the database. Then using Pandas [28] the Json files are converted in arrays in order to be analyzed and elaborated. One of the advantages of Pandas is the possibility to execute checks directly using the timestamp as a parameter, which allows fast controls over the possibility of scheduling a work in predetermined time slot. For the DSM are used different algorithms:

- *Scheduling Algorithm*: it generates the maintenance schedule forecast keeping into account all the constraints provided by the different systems;
- *Operator Algorithm*: it elaborates data related to the users to provide insights about their availability and suggestions on their optimization;
- *Equipment Algorithm*: it elaborates data related to the machines to provide insights about their maintenance activities and availability to optimize their efficiency;
- *Warehouse Algorithm*: it elaborates the data related with the warehouse and in particular the expected spare parts and tools transaction to perform a projection of their availability. The output of the algorithm is used to provide suggestions on the management of the warehouse (e.g., reordering of material);

5.4.1.1 Scheduling algorithm

Its aim is to distribute the maintenance works within a given time interval, taking into account the different periodicity of the work. First of all, the algorithm defines a priority according to the periodicity of the work. It starts by scheduling one-time works and ends by planning recurring ones. As illustrated in section 4.2.2.1, the algorithm differentiates:

- Recurring: the highest priority is assigned to works with higher frequency. For example, daily works are planned before monthly ones. In these cases, the algorithm proceeds in three steps:
    1. It finds the last execution date of the work;
    2. From the identified date it creates a possible projection of the future executions of the work;
    3. It tries to plan all the expected activities along with the generated projection, evaluating also all the other constraints;
- One-time: this work comes from extraordinary maintenance. A fixed schedule interval is associated with the work and the algorithm associates a priority to the event depending on this interval. The smaller the interval, the higher the priority assigned to the work;

The algorithm implements a tolerance mechanism to search a slot that satisfies constraints, moving around a given date.

The Scheduling Algorithm must be able to handle different constraints depending on the assignability of the works, the warehouse situation and fixed maintenance activities. Regarding the constraints on the assignability of the work, the algorithm distinguishes:

- Not assignable: the algorithm searches for a user without checking if the qualification satisfies the required qualification for performing the work
- Assignable: the algorithm searches for a user from among those that match the required qualification for work execution
- Assigned: the algorithm searches directly a free slot of the user already indicated

When the algorithm identifies a different group of operators who can handle the maintenance work, it tries to select the operator with the most amount of time available to balance the works distribution. Afterwards, it checks the availability of spare parts and tools required for maintenance activity by looking at the projection of spare parts and tools availability performed by the Warehouse Algorithm. If a work schedule is fixed by the Maintenance Manager, the algorithm assigns the work to the indicated operator without performing any further check, i.e. without reserving tools (or spare parts) from the warehouse. This happens because these aspects are handled manually by the Maintenance Manager or through an external system, i.e. the Information System. All the controls can be either activated or deactivate to compare different scenarios, for example is possible to guess the schedule using infinite resources or a number much higher than the usual usage, this allows to compare with the normal case and determine if the proposed change is actually feasible to be implemented, so if the time advantage offsets the expense of using more resources.

5.4.1.2 Operator algorithm

The *Operator Algorithm* provides to the Maintenance Manager an estimate of the expected availability for an operator in a certain period as a result of the scheduling forecast. Data from external systems (e.g., Information System) and scheduling forecasts are aggregated in time series to create the KPIs discussed in section 5.3.1.4. These provide information on:

- *Name*: name of the operator
- *Qualification*: associated with the operator
- *Resource Type*: if internal or external
- *Shift*: shift assigned to the operator in the selected date
- *Currently*: status day by day of the operator (busy or free)
- *Availability*: operator expected availability in the selected time frame

5.4.1.3 Equipment algorithm

The Equipment Algorithm gives information about the expected status of the equipment according to the maintenance activity forecasts. Data from external systems (e.g., Information System) and scheduling forecasts are aggregated into time series, showing daily maintenance time per machine. To define the average expected availability of equipment, it's considered a service time of 24-hour per day.

5.4.1.4 Warehouse algorithm

The *Warehouse Algorithm* provides suggestions on the management of the warehouse based on the expected availability of items. The Decision Support System can advise for placing new orders to solve the stock shortage.

Based on data from external systems, the algorithm calculates the stock availability of items, generating a future projection that considers system transactions like material reorders. Once the maintenance activities are planned through the Scheduling Algorithm, the Warehouse Algorithm updates the projections according to the material used during the expected maintenance.

After estimating the effective availability of the items, the algorithm proceeds to achieve the KPIs described in section 5.3.1.3. These include:

- *Component Name*: the identification name of the spare part/tool;
- *Availability*: the effective availability in the warehouse for an item in the selected time frame;
- *Usage Graph*: the algorithm generates a time series keeping the minimum stock availability value for each day, so that the Maintenance Manager can have a preview of a possible stock shortage over time;

- *Estimated out of stock*: to estimate the out-of-stock date of an item, the algorithm first calculates the moving average of stock availability to reduce peaks in the time series. Then, it computes the average usage of the item over the last period and uses this value to empty the remaining availability of the item in the warehouse. This gives the expected out-of-stock date for all items;
- *Suggestion*: once the algorithm estimates the end date of availability, if that date falls within the scheduling period the algorithm provides a suggestion to reorder the item. Instead, if it falls further into the future, the algorithm gives an alert to check the availability trend of the element;

5.5 Information System APIs

This section describes the APIs that are exposed by the DSM backend to receive data from other system modules and return the suggestions. The POST APIs have a payload limit of 100KB, any message that exceed this limit should be split into multiple API calls.

5.5.1 /users

This API is made to return a single user based on the *_id* parameter, which is its unique primary key, and contains all the information about the user.

```
[
 {
  “_id”: 1,
  “customerId”: 1,
"name": "Lukas M.",
"qualification": [
   {“_id”: 1, "name": "tech inzenierus"},
   …]
 },
 …
]
```

Available methods: POST, GET, DELETE

Properties:

- *_id*: unique user id (primary key);
- *customerId*: customer unique identifier;

- *name*: username;
- *qualification*: list of user's qualifications:
  - _id: qualification id;
  - name: qualification name;

5.5.2 /shifts-schedule

This API defines the structure of a shift, with the timestamp of its beginning and ending, and the _ids of the users associated with it.

```
[
 {
  "_id": 1,
  "customerId": 1,
  "start": 1619758800000,
  "end": 1619791200000,
  "shiftType": {"_id": 1, "name": "Shift_1"},
 "userIds": [1, 5, …]
 },
 …
]
```

Available methods: POST, GET, DELETE

Properties:

- *_id*: unique shift schedule id (primary key);
- *customerId*: customer unique identifier;
- *start*: start time relative to start of shift [timestamp (in ms)]. Tab Shift Schedule;
  E.g.: 1619758800000 = Friday, April 30, 2021, 5:00:00
- *end*: end time relative to end of shift [timestamp (in ms)]. Tab Shift Schedule;
  E.g.: 1619791200000 = Friday, April 30, 2021, 14:00:00
- *shiftType*: information contained in the Tab Shifts;
  - *name*: shift name;
  - *_id*: shift id;
- *userIds*: users working in this shift, it refers to the _id provided in the /users API;

5.5.3 /parts

This API it is defined to retrieve all the info about the available parts inside the information system.


```
[
 {
  “_id”: 1,
  “customerId”: 1,
  "balance": 15,
  "name": "lesis",
  "units": {“_id”: 3, "name": "pcs", "type": "pcs"},
  "updatedAt": 1619791200000
 },
 …
]
```


Available methods: POST, GET, DELETE

Properties:

- *_id*: unique spare part id (primary key);
- *customerId*: customer unique identifier;
- *balance*: amount of spare part;
- *name*: spare part name;
- *units*: units for spare part. Tab Parts units.
  - *_id*: units id;
  - *name*: units name;
  - *type*: type of unit;
- *updateAt*: date on which this quantity of spare part is present in the warehouse;

5.5.4 /tools

This API it is defined to retrieve all the info about the available tools inside the information system.

```
[
 {
  “_id”: 1,
  “customerId”: 1,
  "balance": 25,
  "name": "tool-name",
  "type": "tool-type",
  "updatedAt": 1619791200000
 },
 …
]
```

Available methods: POST, GET, DELETE

Properties:

- *_id*: unique tool id (primary key);
- *customerId*: customer unique identifier;
- *balance*: amount of tool;
- *name*: tool name;
- *type*: type of tool;
- *updateAt*: date on which this quantity of tool is present in the warehouse;

5.5.5 /works

This API contains all the information about works and their structure.

```
[
 {
  "_id": 1,
  “customerId”: 1,
  "workCode":"PP-Daily-Clean",
  "workTime": 4,
  "startDate": 1619758800000,
  "endDate": 1620588800000,
  "workPeriod": {“_id”: 1, "name": "Daily"},
  "equipment": {“_id”: 1, "name": "Pp combi 1530"},
  "workType": {“_id”: 3,
     "name": "Change and lubricate",
     "recurring": true,
     "assignable": 1},
  "qualificationId": {1, 3, 5},
  "userId": null,
  "taskList": [1, 2, …],
  "source": "information-system",
 },
 …
]
```

Available methods: POST, GET, DELETE

Properties:

- *_id*: unique work id (primary key). Tab Works;
- *customerId*: customer unique identifier;
- *workCode*: descriptive code of the work to be performed. Tab Works;
- *workTime*: total time of the work calculated through the sum of the execution time of the task inserted into work [minutes];
- *startDate*: starting day to consider the work for scheduling by the algorithm;
- *endDate*: last day to consider the work for scheduling by the algorithm;
- *workPeriod*: information about the recurrence of the work. Tab Works and Tab Work Periods;

    - *_id*: period id;
    - *name*: period name;
- *equipment*: information on machines (or lines) subject to maintenance. Tab Equipment
    - *_id*: equipment id;
    - *name*: equipment name;
- *workType*: information about the type of work to be performed. Tab Work Types.
    - *_id*: workType id;
    - *name*: workType name;
    - *recurring*: identifies whether the work is recurrent [Boolean];
    - *assignable*: identifies whether the work is assignable:
        - 1: AI suggests an operator for the execution of the work;
        - 0: AI doesn't suggest an operator. Anyone with sufficient qualification can perform this work;
- *qualificationId*: the qualification identifier;
- *operator_id*: information about which operator should execute the work (for assigned works). If the work is not-assigned (therefore assignable or not-assignable) this field will be null;
- *taskList*: set of tasks to be performed in the related work, the value refers to the _id identifier passed by the /tasks API;
- *source*: defines the kind of work according to its origin. This can be set to the following values:
    - *information-system*: if it comes from the Information System (IS);
    - *predictive*: if it comes from the Condition Monitoring (CM);
    - *vision-control*: if it comes from Online Quality Control (OQC);

5.5.6 /tasks

This API is defined to retrieve all the information about the tasks inside a work.


```
[
{
 "_id": 1,
 “customerId”: 1,
 "description": "Clear brass …",
 "time": 2,
 "part": [{“_id”: 1, "quantity": 1}, …],
 "tool": [{“_id”: 1, "quantity": 1}, …]
```

```
  },
  {
   "_id": 2,
   “customerId”: 1,
   "description": "Clear brass waste ring …",
   "time": 2,
   "part": [{“_id”: 4, "quantity": 2}, …],
   "tool": [{“_id”: 2, "quantity": 1}, …]
 }
  …
 ]
```


Available methods: POST, GET, DELETE

Properties:

- *taskList*: set of tasks to be performed in the related work. Tab Tasks.
  - *_id*: unique task id (primary key);
  - *description*: description of the task to be executed;
  - *time*: task execution time [minutes];
  - *part*: list of spare parts used for the task (id and quantity);
  - *tool*: list of tools used for the task (id and quantity);

5.5.7 /work-schedule-suggestion

It is the main API for what regards the scheduling, it returns a Json file with all the suggestion generated by the scheduling algorithm(Scheduling algorithm), and it allows to perform a join with the upper mention APIs to create the final schedule displayed in the Fronted.


```
 [
  {
  “customerId”: 1,
  "createdAt": 1619791200000
  “suggestion”: [{
    "workId": 1,
    "userId": 4,
    "shiftId": 4,
    "scheduleDate": 1615668800000,
```

```
    "scheduleDateEnd": 1615668800000,
    },
    …
]
```


Available Methods: POST, GET, DELETE

Properties:

- *customerId*: customer unique identifier;
- *createdAt*: moment of the suggestions generation;
- *suggestion*: list of suggested schedules:
  - *workId*: work id, it refers to the _id value passed in the /works API;
  - *userId*: id of the user assigned to this schedule; it refers to the _id passed in the /users API;
  - *shiftId*: id of the shift that refers to this schedule, it refers to the _id of shiftType passed in the /shifts-schedule API;
  - *scheduleDate*: start of the work scheduling [timestamp (in ms)];
  - *scheduleDateEnd*: end of the work scheduling [timestamp (in ms)];

5.5.8 /work-schedule

This API provides the work schedules set by the Maintenance Manager and the work performed with execution information (fact date and task execution time).


```
[{
  "_id":10,
  “customerId”: 1,
  "workId": 1,
  "scheduleDate": 1615668800000,
  "userId": 4,
  "factDate": 1619982800000,
  "task": [
    {
    "taskScheduleId":10,
    "taskId":1,
    "executionTime": 3},
   …
]
```

```
  },
  {
    "_id":11,
    “customerId”: 1,
    "workId": 2,
    "scheduleDate": 1616668800000,
    "userId": 5,
    "factDate": null,
    "task": null},
  …
  ]
```


Available Methods: POST, GET, DELETE

Properties:

- *_id*: unique identifier related to the generated work schedule (primary key);
- *customerId*: customer unique identifier;
- *workId*: work identifier, it refers to the _id passed with the /works API;
- *scheduleDate*: work schedule date. [timestamp (in ms)];
- *userId*: user identifier of the operator assigned to this schedule; it refers to the _id passed with the /users API;
- *factDate*: date of execution of the work schedule. If the work schedule has not yet been executed, the value is null;
- *task*: list of executed tasks for the work schedule. If the work schedule has not yet been executed, the value will be null;
    - *taskSchedule_id*: task schedule id;
    - *taskId*: task identifier, it refers to the _id passed with the /tasks API;
    - *executionTime*: task execution time;

5.5.9 /transactions

This API represents the transactions made for each work, so it displays an event in which some parts or tools are either removed or added to the warehouse table inside the database, it determines the balance of parts and tools displayed in the frontend.


```
[
 {
    "_id": 1,
```

```
    “customerId”: 1,
    "transactionDate": 1619982800000,
    "taskScheduleId": 1,
    "status": "reserved",
    "action": "subtraction",
    "partId": 2,
    "toolId": null,
    "planQuantity": 1,
    "factQuantity ": null
},
{
    "transactionId": 2,
    "transactionDate": 1619982800000,
    "taskScheduleId": 2,
    "status": "completed",
    "action": "subtraction",
    "partId": 2,
    "toolId": null,
    "planQuantity": 1,
    "factQuantity ": 2
},
…
 ]
```


Available Methods: POST, GET, DELETE

Properties:

- *_id*: unique transaction identifier (primary key);
- *customerId*: customer unique identifier;
- *transactionDate*: date of transaction execution;
- *taskScheduleId*: id related to the task schedule that generated the transaction, it refers to task. taskScheduleId of the /work-schedule API;
- *status*: identifies the status of the transaction. It can take the value reserved (transactions planned) or completed (for transactions expired);
- *action*: identifies the type of transaction. It can take the value addition for material reorder transactions or a subtraction value for transactions involving the use of material;

- *partId*: spare part related to the transaction, null If not provided;
- *toolId*: tool related to the transaction, null if not provided;
- *planQuantity*: quantity scheduled for the transaction;
- *factQuantity*: quantity effectively used for the task. As long as the task has not been executed, the value is null;

5.5.10/compute

It is the API which triggers the execution of the algorithms, based on the customerId which uniquely identifies a customer, and the date in which the schedule starts (the schedule will automatically schedule for the next six months).

```
{
  "customerId": 1,
  "to": 1619740800000,
}
```

Available Methods: POST, GET

Properties:

- *customerId*: customer id;
- *to*: end date of scheduling [timestamp (in ms)];

5.6 Condition Monitoring APIs

In this section we are going to cover the integration with the Condition Monitoring module (predictive maintenance). It must be specified that the realization of this system was not part of the DSM development, and it had to be completed by a partner company, which did not finish it yet. Anyway, the API to interact with this module has been defined, so that whenever it will be complete the DSM frontend and backend will be ready to be linked with the information coming from that source.

5.6.1 API


```
[
 {
  "_id": 1,
  “customerId”: 1,
  "workCode": "Generic-Machine-Check",
  "workTime": 4,
  "startDate": 1619758800000,
  "endDate": 1620588800000,
  "workPeriod": {“_id”: 6, "name": "Onetime"},
  "equipment": {“_id”: 1, "name": "Pp combi 1530"},
  "workType": {“_id”: 0,
     "name": "Check machine",
     "recurring": false,
     "assignable": 1},
  "qualificationId": null,
  "userId": null,
  "taskList": null,
  "source": "predictive",
 },
 …
]
```


The idea is that the Condition Monitoring will create a new work similar to the ones provided by the information system, with a different source though, which it will specify from which system the work is generated. It will be displayed uniquely inside the frontend calendar, and it will be considered a priority depending on the distance between the present date and the actual predicted date for the machine to break down. For the description of the fields refer to 5.5.5 /works,

5.7 Online Quality Control APIs

The same can be said for this module, since is not finished yet. It is presented the designed API, made to easily integrate the Module once it is ready.

5.7.1 API


```
[
    {
     "_id": 1,
     “customerId”: 1,
     "workCode": "Generic-Machine-Check",
     "workTime": 4,
     "startDate": 1619758800000,
     "endDate": 1620588800000,
     "workPeriod": {“_id”: 6, "name": "Onetime"},
     "equipment": {“_id”: 1, "name": "Production line 1"},
     "workType": {“_id”: 0,
        "name": "Check line",
        "recurring": false,
        "assignable": 1},
     "qualificationId": null,
     "userId": null,
     "taskList": null,
     "source": "vision-control",
    },
    …
  ]
```


The idea behind this API reflects the one of the Condition Monitoring, it adds a work similar to the one coming from the Information System, but with a different source. The peculiarities follow the one of the Condition Monitoring Module.

5.8 Conclusions

All the implementation has been designed to optimize the interaction between the different components of the DSM. The APIs, for example, have been structured in order to be easily and quickly readable from the frontend, avoiding useless loops which would have slow down the whole user experience. Also, the algorithms, since they can be called in any time and even multiple times in parallel have been thought with the queue management of the backend made through Redis and the avoidance of useless loops inside the decision-making algorithms.

6. Chapter 6

6.1 Laboratory Tests

In this section are presented all the experimentation and tests done to verify the validity of the developed system. Since a set of real data is yet to be updated inside the Information System, a set of synthetic data has been created in order to test the DSM.

6.1.1 Generated Dataset

To test the Decision Support system functionalities, a set of data is generated from data collected directly from a production plant following a process of data augmentation. In the context of simulating a real maintenance planning problem, it is decided to verify the solution by testing one month of scheduling (specifically October 2021). In addition, different data sources are simulated to verify the independence of the solution from single systems (e.g., Information Systems, Condition Monitoring System or Online Quality Control).

The data are generated with the following criteria:

- *users*: users with one or more qualifications are created, to match the qualifications required by the expected works;
- *shifts-schedule*: a set of shifts is generated in the range of dates from 2021/10/01 to 2021/10/31. The shifts are divided into four different time frames of 8-9 hours, with the possibility that a shift could be across two consecutive days. Two days of rest per operator, per week, were maintained to simulate a week composed of 5 working days;
- *parts*: an amount of different spare parts is generated to cover a minimum of maintenance activities of one month;
- *tools*: a sufficient amount of tools is generated (as for the spare parts) to satisfy a month of maintenance activities;
- *works*: a set of different types of works, such as recurring (daily, weekly, monthly, etc.) or one-time works, is generated to provide different priorities to be planned. In addition, a scheduling period is randomly generated for each work. The works are assigned to different machines and require different users' qualifications for their execution (one or more qualifications). To simulate a more realistic situation, also assignable, not assignable or assigned

work (as described in the section 3.2.1) are generated. Moreover, works from the different systems interacting with the solution are generated to test the independence of the DSM from the external systems;

- *tasks*: a set of tasks is generated to guarantee a sufficient turnover in the composition of the works. Each task is associated with a plausible execution time (a random time value between 1 minute and 30 minutes) that refers to cleaning, check or repair tasks;
- *works-schedule*: a set of fixed works schedules are generated to test two kinds of functionalities:
    - *Recurring works scheduling*: the already performed works are evaluated to define the starting point for the subsequent recurring works. For example, if a weekly work was last performed on Monday, then the suggestion of the next scheduling should be the following Monday. At the same time the algorithm evaluates if, for example, the work is already scheduled the next Tuesday avoiding suggesting it on Monday;
    - *Fixed works scheduling*: the works fixed by the Maintenance Manager need to be already assigned to a user and placed in the calendar. A date is randomly set within the scheduling period;
- *transactions*: a set of warehouse transactions is generated in random time periods to simulate some reservation or reordering of items;

The schema in Figure 36.1 summarizes the number of objects created for each entity type described above and the amount of the generated suggestions.

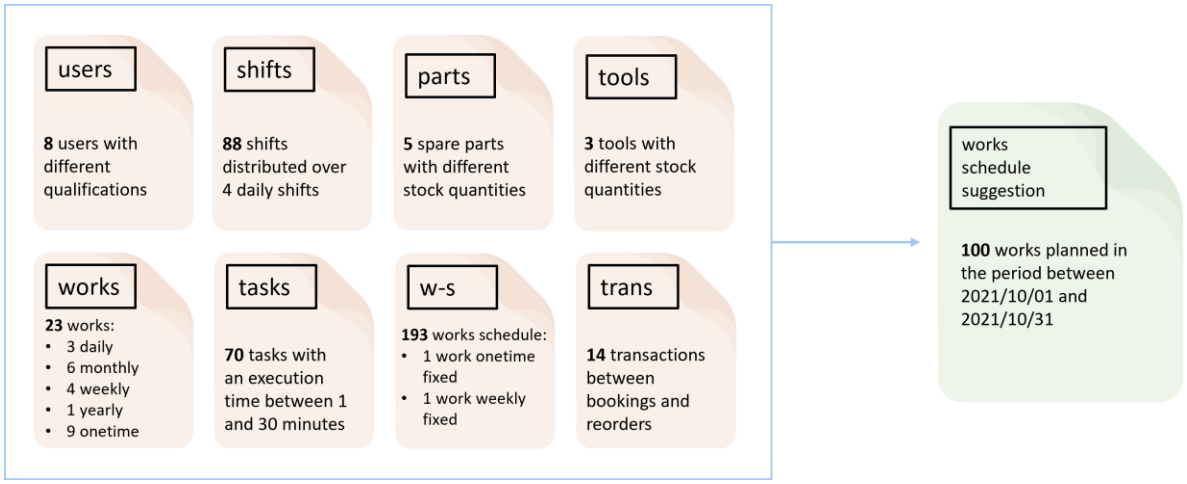


Figure 36.1: Summary of input sources (left)and output (right) achieved

6.1.2 Test

A first set of tests was performed on a local machine to perform a functional test of the system. In a second phase, the tests were performed on a virtual machine via the AWS cloud computing service. The AWS service selected is an EC2 t2.micro instance providing 1 CPU and 1 GB of ram. This type of solution allows us to balance the required network resources and the workload required by the algorithm's calculation.

As mentioned in the previous section, with the aim of simulating a plausible maintenance planning situation, a maintenance month was set as the scheduling range. This has allowed the solution to be tested over a reasonable number of days to estimate the computational load on the machine. In addition, to evaluate the robustness and independence of the solution, tests were performed with data sources from the different integration systems (illustrated in section 5.5). The entire test was performed by simulating the execution for a single customer.

6.1.3 Results

The results of the performed tests are summarized in Table 5

Work type	n° total of work to plan	n° work scheduled	n° failed work scheduling	Details of failed work schedule
Daily	93	66	27	failed works on days when there are no shifts (not relevant)
Weekly	19	19	-	-
Monthly	6	6	-	-
Yearly	1	-	1	failed because its last execution date is the previous month
Onetime	9	9	-	-
Fixed	2 (1 onetime and 1 weekly)	2	-	-

Table 5: Type of work present in the scheduling simulation

The first column shows the total number of works expected to be scheduled within the month, the second column shows the number of works that have been planned in the scheduling range and the third column shows the number of works that the algorithm failed to schedule in the scheduling range. We can notice that for the daily works 93 works were expected to be planned while only 66 were effectively planned and this results in 27 works not scheduled. An analysis on the causes shows that the

not scheduled works correspond exactly to the works expected to be executed in the 2 days off per week. This means that this is a correct behavior since the system respects the provided requirements for the needed qualification. For the yearly work the provided timeframe was too short, a test on a more extended period showed a correct scheduling of this kind of task. The calendar in Figure 37.1 shows the obtained results in graphical form. The different colors represent the different types of planned works (as discussed in section 5.3.1.2). Also note that in the two days off (Sunday and Monday) no maintenance work was planned.

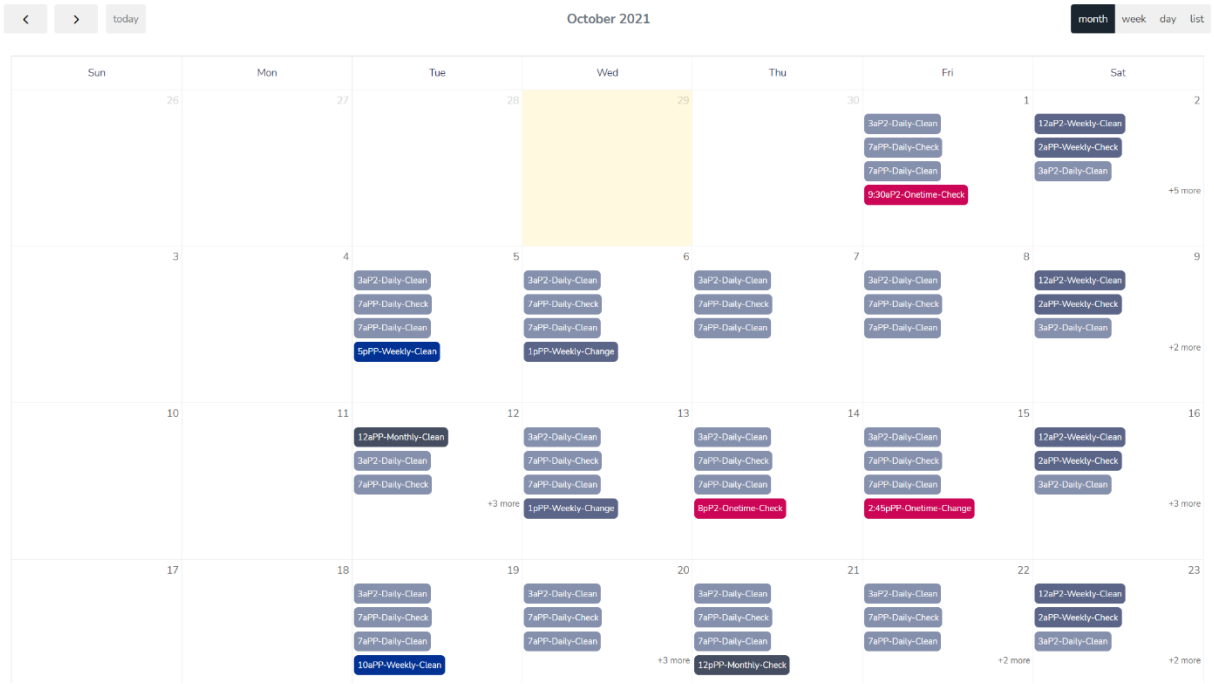


Figure 37.1: Scheduling obtained from the test

The illustrates the results obtained from the different steps of the algorithm in terms of execution time. These are average times estimated from multiple simulations run on the same scheduling range (October).

	Get data API	works	users	warehouse	schedule	Total time (s)
Time (s) (average values estimated from different simulations run on the same scheduling range)	~ 1.5	~ 0.2	~ 0.4	~ 0.1	~ 0.8	~ 3.0

Table 6: Execution times of the different sections of the algorithm

For what regards the frontend, the adopted method is the waiting of the composition of the main event list got by the /work-schedule-suggestion API. The approach is this because the generated list contains some information which are necessary for the correct rendering of all the views of the frontend. The time needed for the elaboration and realization of this list is settled around 15 seconds, which is naturally expected to increase for the scheduling of a longer period of time.

6.1.4 Conclusions

Considering the test results, the solution is able to schedule the works provided by different systems, failing only on days when specific qualifications are not present. Even recurring works that are planned outside of the scheduling period (e.g., yearly work) are not scheduled.

A log containing all the works that fail due to lack of operators or lack of materials in the warehouse is added to the system to record the causes and provide them to the Maintenance Manager. The objective is to provide to the Maintenance Manager an additional set of decision support information that can help him to evaluate the situation and take the proper actions (e.g., increase of shifts, personnel, materials etc.) to perform all the expected maintenance activities.

Regarding the execution time, it is measured that the API call and data ingestion is currently the most time-consuming task respect to the algorithm execution. This calculation is based on the number of created entities (discussed in the section 5.1) and, with the increment of the number of works and users, the difference between data ingestion and algorithm execution is expected to increase.

Finally, we can establish that the algorithm succeeds in finding solutions that respect the defined constraints, and in returning valid suggestions to the Maintenance Manager. Possible adjustments can be explored with the aim of improving the amount of suggestions for the Maintenance Manager, for example providing a set of causes for the not scheduled tasks. With the usage of the system more real data will be available from the plant, this will allow to test the effectiveness of the scheduling algorithm through accurate metrics (e.g., work costs, material costs, maintenance, and downtime costs, etc.).

7. Conclusion and future development

The developed Decision Support System is an innovative solution to the problems related to the decision making inside the production plant. It will allow the Maintenance Manager to easily monitor the present and future status of the plant. The DSM has the objective of helping the Maintenance Manager in taking immediate decisions and to easily tackle unexpected events. This is why the system has been developed with the focus on speed, usability, and clarity, developing a fast and intuitive solution which will ease the work of the person in charge of taking decision over matters which, without any support, are mostly experienced based.

The system provides a wide variety of feature:

- Decision Support Module;
- Information System;
- Condition Monitoring;
- Online Quality Control;

At least in theory, in fact due to issues external from our range of influence the last to feature were not developed so they are impossible to be practically integrated. This is a symptom of a multi layered project, composed by different companies, which have to develop their own modules independently, but in the end collaborate, that had an insufficient level of managerial skills from whom had the task to coordinate the work between different partners. This had a snowball effect, because the absence of a constant guide in coordinating the different modules, ended up in delays in the delivery of certain functionalities. From our prospective we ended up waiting and loosing time because of the lack of a certain functionalities that would have been due from another partner. This also conditioned the testing phase, due to the absence of real data we had to create a synthetic set of data which would represent at his best the plant on which this first version of the system is based. Denying the possibilities of applying the DSM to a realistic scenario. Internally the situation was much different, the deliverables have been all updated on time and the project design and development had been constant for what was possible.

The system is still rough, missing some functionalities due to external causes, but also still lacking robustness since the testing had been very limited in scale. A margin of optimization is palpable both in the frontend and in the backend, in fact the API calling process requires a time which is acceptable for the tested scenario, but that will probably be unacceptable for any realistic 6-month schedule, so a work of optimization is required to entitle the system as market ready. The algorithms on the other end lack robustness, because they expect only the planned document formats and lack an optimal management of errors and unexpected situations. These are all problems, though, that do not change the innovative nature of the project in a sector often overlooked and with little literature which covers the same subjects. Is an attempt to coordinate different well-known practices into a unique system which tackles an, often perceived as marginal but actually fundamental, aspect of the production endeavor, maintenance.

Thanks to having modularity in mind since the beginning, the DSM is open to be updated in many different directions, depending mostly on the needs of the specific plant. In fact depending on each infrastructure it could be possible to add more and more features. For example it would be possible, even if very intense, to add a digital twin representation for the monitoring of the equipment. A system which monitors in real time the status of execution of the works, or a more complex predictive system based not only on operative research, but on more advance techniques which can take advantage on the latest trends in AI, Big Data and industry4.0.

Bibliography

[1] R. D. &. P. A. Scarf, “On the impact of optimisation models in maintenance decision making: the state of the art,” *Elsevier Science Limited*, vol. 1, no. 1, p. 9, 1998.

[2] O. Merkt, "Predictive Models for Maintenance Optimization: an Analytical Literature Survey of Industrial Maintenance Strategies," *Hohenheim University, Schloss Hohenheim 1, 70599 Stuttgart, Germany*, vol. 1, no. 1, p. 20.

[3] A. De Mauro, M. Greco and M. Grimaldi, "A formal definition of Big Data based on its essential features," *Emerald Sight*, p. 14, 2016.

[4] R. Alessandro, "Big Data - Architettura, tecnologia e metodi per l'utilizzo di grandi basi di dati," p. 300, 2013.

[5] S. SAGIROGLU and D. SINANC, "Big Data: A Review," p. 6, 2013.

[6] C. R. Murciano, "What is the challenge in creating a process-based digital twin?," p. 116.

[7] F. Wortmann and K. Flu¨chter, "Internet of Things Technology and Value Added," p. 4.

[8] N. Jun and J. Xiaoning, "Decision support systems for effective maintenance operations," p. 4, 2012.

[9] C. J. Turner, C. Emmanouilidis, T. Tomiyama, A. Tiwari and R. Roy, "Intelligent decision support for maintenance: an overview and future trends," *International Journal of Computer Integrated Manufacturing*, pp. 936-959, 2019.

[10] D. Bumblauskas, D. Gemmill, A. Igou and J. Anzengruber, "Smart Maintenance Decision Support Systems (SMDSS) based on corporate big data analytics," *Expert Systems With Applications*, vol. 90, pp. 303-3017, 2017.

[11] S. Selcuk, "Predictive maintenance, its implementation and latest trends,"

Institution of Mechanical Engineers, pp. 1670-1679, 2016.

[12] Z. Weiting, Y. Dong and W. Hongchao, "Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey," *IEEE SYSTEMS JOURNAL,* vol. 13, no. 3, pp. 2213-2227, 2019.

[13] R. GALAMBOŠ, J. GALAMBOŠOVÁ, V. RATAJ and M. KAVKA, "DESIGN OF CONDITION-BASED DECISION SUPPORT SYSTEM FOR PREVENTIVE MAINTENANCE," *DE GRUYTER OPEN,* no. 1, pp. 19-22, 2017.

[14] N. Nikolakis, A. Papavasileiou, K. Dimoulas, K. Bourmpouchakis and S. Makris, On a versatile scheduling concept of maintenace activities for increased availability of production resources, Science Direct, 2018, pp. 172-177.

[15] Express, "Express Api Documentation," [Online]. Available: https://expressjs.com/it/guide/routing.html.

[16] O. J. Founfation, "Node.JS Documentation," [Online]. Available: https://nodejs.org/it/about/.

[17] M. DB, "Mongo DB documentation," [Online]. Available: https://docs.mongodb.com/.

[18] MIT, "Mongoose Documentation," [Online]. Available: https://mongoosejs.com/.

[19] Pallets, "Flask Python web server documentation," [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/.

[20] S. Sanfilippo, "Redis Documentation," [Online]. Available: https://redis.io/documentation.

[21] V. Driessen, "PythonRQ Documentation," [Online]. Available: https://python-rq.org/docs/.

[22] F. Open Source, "React Documentation," Facebook, [Online]. Available: https://it.reactjs.org/docs/getting-started.html.

[23] G. inc., "Material Ui documentation," Google, [Online]. Available: https://mui.com/getting-started/usage/.

[24] MIT, "Chart js documentation," MIT, [Online]. Available: https://github.com/chartjs/Chart.js.

[25] J. Ayerst, "react-chartjs-2 documentation," MIT, [Online]. Available: https://www.npmjs.com/package/react-chartjs-2.

[26] MIT, "MUI-Datatables - Datatables for Material-UI," MIT, [Online]. Available: https://github.com/gregnb/mui-datatables.

[27] F. LLC, "FullCalendar Documentation," FullCalendar LLC, [Online]. Available: https://fullcalendar.io/docs.

[28] t. p. d. team, "Pandas Documentation," pandas development team, [Online]. Available: https://pandas.pydata.org/docs/.

List of Figures

List of Tables

List of Symbols

Variable	Description	SI unit
TTPM	Time to preventive maintenance [13]	Date
ICC	Informative cycle counter [13]	Pure
M_c	Cumulative material consumption per week [13]	Units
EL_t	Equipment list defines time or cycle [13]	Pure
MRP	Material request planning/Average material forecasting per week [13]	Units
PM_R	Records of preventive maintenance done per each operation of PM needed [13]	Pure
PM_f	preventive maintenance frequency – frequency of either time, or cycles prescribed by equipment producer and/or an additional organizational setup [13]	Pure