AGILEHAND

D6.1- AGILEHAND Agile, Flexible and Rapid Reconfigurable SUITE v1

WP6 – BUILD: Agile, Flexible and Rapid Reconfigurable SUITE

AGILEHAND has received the funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101092043

Document Information

GRANT AGREEMENT NUMBER	101092043	ACRONYM	AGILEHAND
FULL TITLE	Smart grading, handling, and packaging solutions for soft and deformable products in agile and reconfigurable lines		
START DATE	01-01-2023	DURATION	36 months
PROJECT URL	https://agilehand.eu/		
DELIVERABLE	D6.1 – AGILEHAND Agile, Flexible and Rapid Reconfigurable SUITE v1		
WORK PACKAGE	WP6 – BUILD: Agile, Flexible and Rapid Reconfigurable SUITE		
DATE OF DELIVERY	CONTRACTUAL: 12-2024	ACTUAL	12-2024
NATURE	Report	DISSEMINATION LEVEL	Public
LEAD BENEFICIARY	UPV		
RESPONSIBLE AUTHOR	Marcos Terol Lloret (UPV), Javier Mateos Luengo (UPV), Ludovica Miele (UPV)		
CONTRIBUTIONS FROM	UPV, FHG, UPM		
TARGET AUDIENCE	1) AGILEHAND project partners; 2) industrial community; 3) other H2020/HEUROPE funded projects; 4) scientific community.		
DELIVERABLE CONTEXT/ DEPENDENCIES	This document is the first iteration. Its relationship to other documents is as follows: - D6.2 Title: AGILEHAND Agile, Flexible and Rapid Reconfigurable SUITE v2		
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	None		
READING NOTES	None		
ABSTRACT	The scope of D6.1 is to showcase the progress made so far within WP6 regarding the four pilots.		

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	07-11-2024	ToC	Created ToC and document structure	UPV
0.2	11-11-2024	Working version	1st draft of the document (T6.4)	UPV
0.3	27-11-2024	Working Version	2nd draft of the document (T6.2)	Stefano Croci (UPM)
0.4	09-12-2024	Working Version	3rd draft of the document (T6.1 and T6.3)	FHG
1.0	12/12/2024	Final Draft version	Final version for internal review	UPV

Disclaimer

Copyright message

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
CSV	Comma-Separated Values
CV	Cross Validation
DES	Discrete Event System
DT	Digital Twin
EC	European Commission
ERP	Enterprise Resource Planning
ETA	Estimated Time of Arrival
GBT	Gradient-Boosted Trees
GPC	Global Product Classification
GST	Goods and Services Tax
GSTIN	GST Identification Number
HACCP	Hazard Analysis and Critical Control Points
HTTP	HyperText Transfer Protocol
IEC	International Electrotechnical Commission
IR	Infrared
ISA	International Society of Automation
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MES	Manufacturing Execution System
MQTT	Message Queuing Telemetry Transport
NIR	Near-Infrared
OPC/UA	Open Platform Communications/Unified Architecture
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
RGB	Red, Green, and Blue
SVM	Support Vector Machines
T	Task
TCP/IP	Transmission Control Protocol/Internet Protocol
UI	User Interface
WP	Work Package

Executive summary

This deliverable describes the activities carried out within WP6 with AGILEHAND pilots, namely Multiscan, Sant'Orsola, Produmar, and Marelec. In particular, the production traceability solution, the data-driven digital twin, the intelligent rapid configuration of the production system, and the optimization of automated production line operations are described in the following sections.

Document structure

Section 1 presents the “Agile, Flexible and Rapid Reconfigurable SUITE” architecture to deliver the integrated traceability solution, the objective of WP6.

Section 2 describes the work carried out on the "AgileHand Production Traceability" solution, developed in task 6.1, describing the design status, the current implementation, and the transferability of the solution to other use cases.

Section 3 describes the work carried out on the "AgileHand Data-Driver Digital Twin," developed in task 6.2, describing the design status, the current implementation, and the transferability of the solution to other use cases.

Section 4 describes the work carried out on the "AgileHand Intelligent Rapid Configuration of Production System," developed in task 6.3, describing the design status, the current implementation, and the transferability of the solution to other use cases.

Section 5 describes the work carried out on the "AgileHand Automated Production Line Operations Optimisation" solution, developed in task 6.4, describing the design status, the current implementation, and the transferability of the solution to other use cases.

Section 6 includes the document's conclusions, and the work carried out in the four tasks of WP6.

1. Agile, Flexible and Rapid Reconfigurable SUITE

The agile, flexible, and rapidly reconfigurable suite aims to deliver an integrated traceability solution as part of a digital twin architecture designed to optimize production processes. To achieve this, the architecture incorporates two fundamental layers:

1. Persistence Layer

This layer ensures robust data management and interoperability through:

- **Standardized Data Model**: Data is organized and structured following the IEC-62264 standard, enhancing interoperability across systems. PostgreSQL is used as the primary database.
- **Data Source Integration**: PostgreSQL facilitates seamless integration with various data sources.
- **Time Series Database:** Timescale DB enables efficient storage and analysis of time-series data.

2. Integration Layer

This layer bridges data flow and system connectivity at multiple levels:

- **Shop Floor Level:** Node-RED is used to ingest and process real-time data from the production environment.
- **Higher-Level Systems:** PostgREST ensures integration with MES and ERP systems, enabling streamlined communication and functionality.

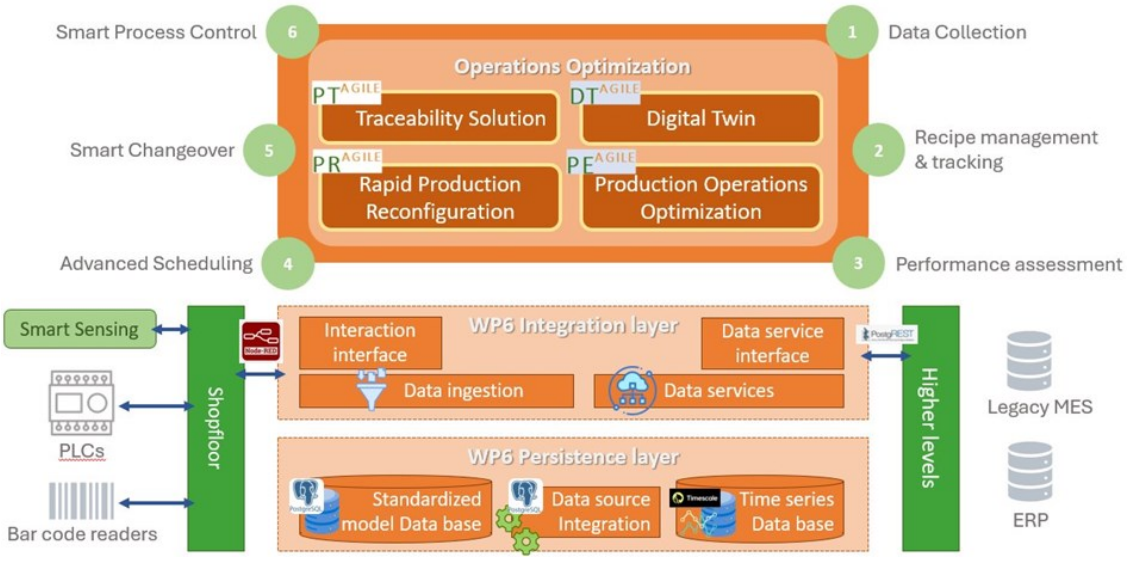

Figure 1: Agile, Flexible and Rapid Reconfigurable SUITE architecture

For the different pilots, the solution incorporates:

- Use Case-Agnostic System Architecture
- Data Model Definition
- Definition of pilot use case specifications:
    - Use case mapping based on the data model
    - Relevant parameters identification
    - Use case front-end mock-ups
    - Integration with existing information systems
    - Simulation/analysis functions
    - High-level functions

2. AGILEHAND Production Traceability Solution

2.1. Overview

The primary objective of T6.1 is the development of a product-oriented traceability solution. It aims to enhance productivity by delivering additional relevant information to support the production process. This involves the development of a database model for data orchestration, user-specific functions to provide decision-making support, and a user interface concept for streamlined interaction with the solution.

The general project settings align with the General Food Law Regulation and EU standards, including ISO 22000, ISO 22005, ISO 9001, and HACCP. The database traceability solution follows the IEC-62264 structure. The project will use open-source software development with PostgreSQL as the common database, and PgAdmin4 will be used to retrieve data using SQL queries.

In Figure 2, the concept of traceability solution is shown. The concept consists of three main layers. At the Input layer at the bottom, the individual data from the shopfloor machines, the MES, ERP, or other production systems will be acquired, which is important to trace the product along the line. Moreover, in the second layer (Traceability Solution), the acquired data will be systematically stored in a database. By developing multiple functions, specific product information, which is important for tracing the product or improving the productivity in the line, can be provided to the user. This will happen in the third layer (Output), where the specific information can be displayed over an individual user interface. By ensuring the alignment of several standards and norms, the concept can be transferred to further use cases.

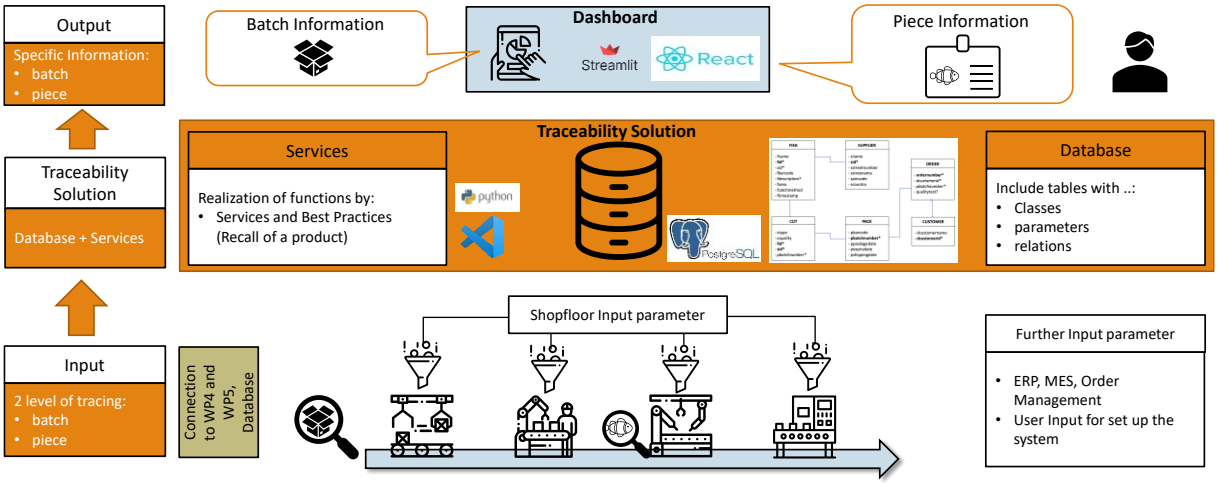

Figure 2: Big picture of the traceability concept

2.2. Design Status

The following paragraph will explain the current implementation and future steps of the traceability solution.

Traceability concept evolution

The development of the traceability concept can be divided into different phases. In this phase, three versions were developed, which will be described below.

V1: First version of database concept

The initial concept was developed by exclusively analyzing Produmar's use case, without a complete alignment with ISO patterns or standards in this preliminary version, but to provide a first draft of the database concept to start with. Consequently, six specific tables were introduced in this draft and are visualized in Figure 3:

- Fish: Stores data related to the products sold by Produmar, specifically different types of fish and their attributes.
- Supplier: Contains information regarding only the fish suppliers.
- Cut: Describes quality-related information about the fish cuts.
- Pack: Keep packaging data, including barcodes and important dates.
- Order: Houses data related to customer orders for the products.
- Customer: Holds customer information, including their names.

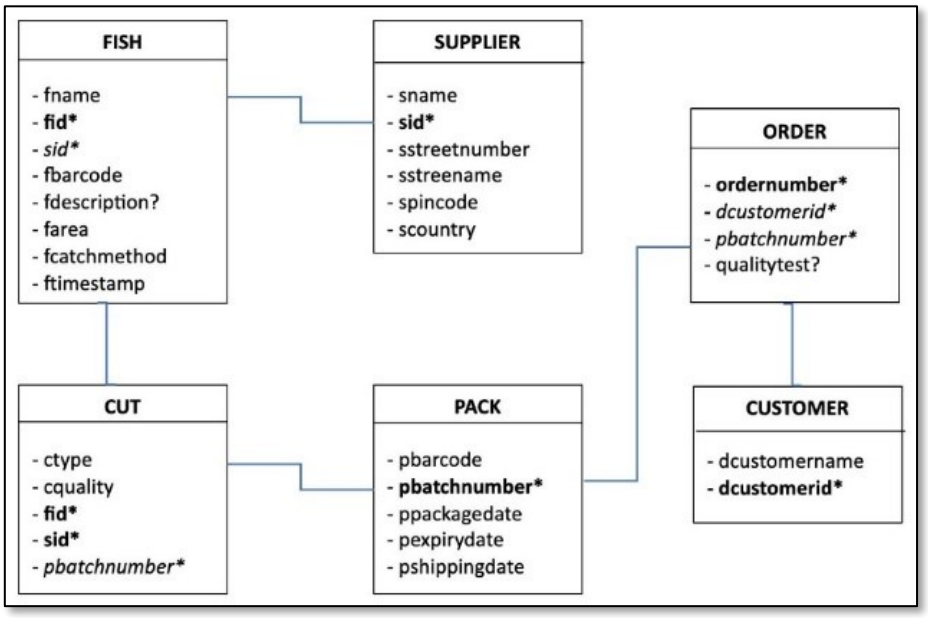

Figure 3: V1 First version of the database concept

The main purpose of this concept was to provide the pilot with a preliminary idea of how the solution could function. It served as a solid foundation for brainstorming more detailed functions. The concept was designed with a specific and not flexible model featuring just six tables. This structure meant that adding more data would require changes to the existing structure, making it less adaptable to evolving needs. Furthermore, this design needs to be updated to reflect the complexity of Produmar's data, which detracts from capturing the actual depth and breadth of information necessary for the application.

V2: Second version of database concept

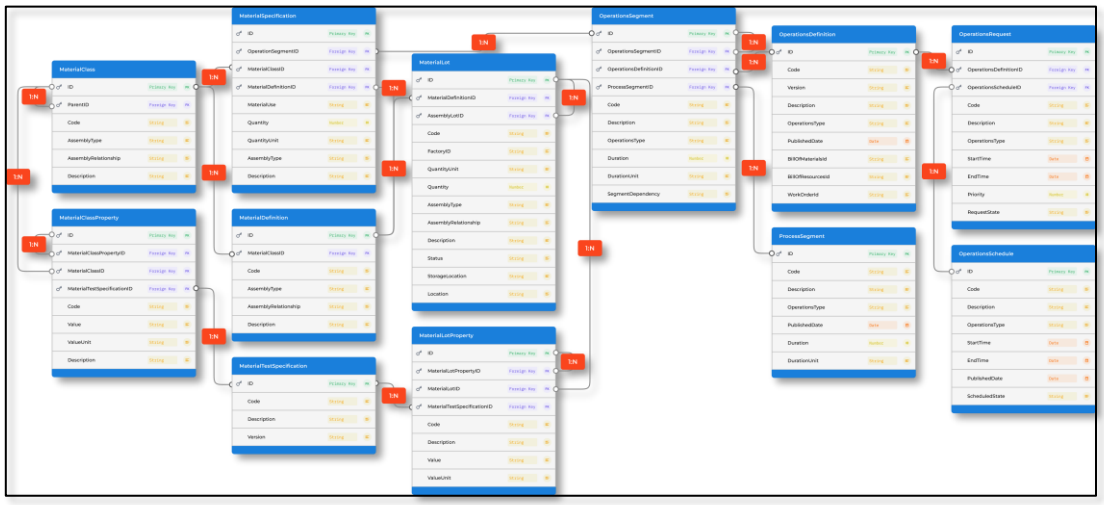

Figure 4: V2 Second version of the database concept

The second concept (Figure 4) was developed building up on the first version concept and in accordance with the IEC-62264 (ISA-95 Standard), which outlines a framework for implementing a database capable of delivering traceability information across various types of organizations. Initially, twelve tables were created to store the same information as the previous concept but with enhanced flexibility, including information about material, operation, and production process information.

The developed concept incorporated a generic, standardized, and flexible model. It was designed with adaptability in mind, being readily prepared to receive any type of product or process input. This allowed for seamless integration and usage across various product lines and processes. The concept consisted of a total of 12 tables, covering a broad spectrum of operations and processes. These included seven material tables, four operation tables, and one process table. The delineation of these tables facilitated better organization and understanding of the data. Furthermore, 18 relationships were defined to enhance the connectivity between the different entities. These relationships enabled better mapping of the data and processes, further illustrating the interdependencies and connections between different areas.

V3: Third version of database concept

Based on the V2 of the database concept, a new version (V3) for integrating further components and creating a more realistic traceability system is under development at the time of finalization of this report. Since it is based on V2, V3 is also aligned with the IEC-62264 standard. The main differences between the versions are that the V3 database concept now also includes specific tables and properties for the workers involved (personnel model), the used equipment (equipment model), and the capability of the used resources in the production process (capability model).

Furthermore, small adaptations to the tables and properties for the used materials and products align the solution with the given use case. In addition, small adjustments to the operations tables were made.

The V3 increases the complexity of mapping the production line into traceability by storing more information along the process. Still, it is also built as the base for further optimization tasks like

simulations, which can use the stored data. After finalizing the V3 concept, the concept will be implemented in the PostgreSQL database. The actual status of the developed concept can be seen in Figure 5, and the main components are briefly described below. The figure provides an overview of the different tables clustered to the category's material, operation, equipment, personnel, capability, and performance.

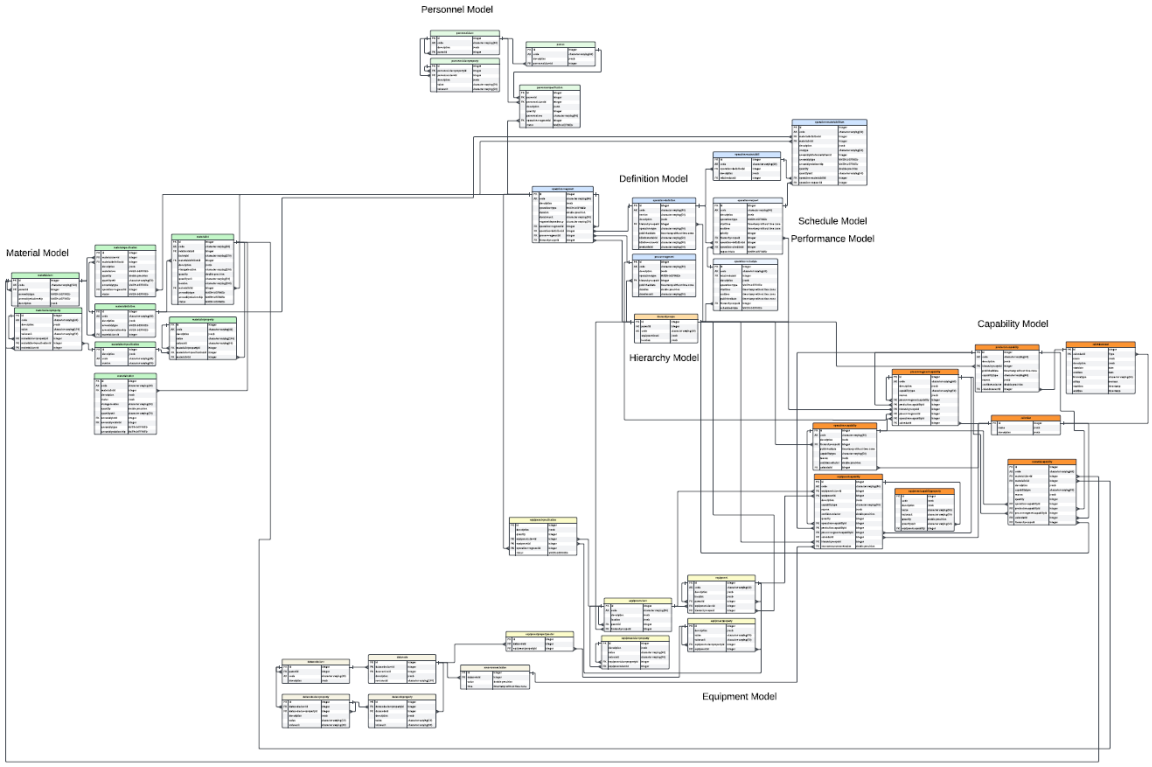

Figure 5: Actual state of the V3 third version of the database concept

To summarize, the developed concept follows the selected manufacturing standard and ensures data storage for gaining useful information, which can be provided to the users. The individual functions will use the data and provide suitable information to the user to support productivity, efficiency, and safety on the shop floor and at the management level.

2.2.1. Current Implementation

The pilot company, Produmar, was selected to develop the traceability solution. In this chapter, a short pilot description is given first. Then, the data collection process is described in more detail. After that, the defined requirements and planned functions to develop the solution are presented. Moreover, relevant norms, standards, and regulations are described. Additionally, integrating sensors for the traceability solution will be explained briefly. Finally, the developed front-end mock-up and the actual implementation status are summarized.

2.2.1.1. Produmar

In this chapter, the specific Produmar use case will be presented. The production line for the fish filet from Produmar can be described in ten steps. In the first step, the fish is unwrapped to begin the preparation. Next, it is cut into various pieces, creating different sizes of fillets to meet diverse

customer needs. The fish pieces are then stored in batches within a freezing chamber at -40°C, cooling them down to preserve freshness. Following the freezing, the fish enters a glazing bath. This step is crucial as it reduces the sticking of pieces to one another and creates a shiny ice surface over the fish, enhancing its appearance and quality. Once glazed, each piece is wrapped in plastic for protection. The plastic wrap and the fish pieces are then cut into individual portions to facilitate packaging. Afterwards, the plastic is shrunk around the fish to ensure a secure seal, preventing any exposure to air or contaminants. The wrapped fish undergoes inspection to check for any holes in the plastic or damage. Once inspected, each fish receives a barcode label for tracking and identification purposes. Finally, the fish is stored in a freezing warehouse, ready for transportation to the final customer.

Data Collection

The data provided by Produmar includes crucial product information essential for establishing the traceability system. This information was obtained from different sources: Product data was collected in a CSV file and a printed production report that offers more detailed information regarding batches and quality metrics.

However, the current data does not include all the information that needs to be stored in the database to fulfill the traceability standards. Therefore, further relevant sensors for tracing the product through production at specific steps and gaining further data is needed. The detailed concept of physical sensors will be worked out with partners from the other work packages (e.g., WP4).

To enhance the transfer of data into the traceability system and ensure comprehensive data integrity, a new data structure was recommended to Produmar. Following that structure ensures that external data can be added to the database easily.

Definition of requirements and functions

For the development of the traceability system, several requirements and suitable functions of the planned system were set.

A workshop was conducted to define and discuss the requirements and initial functions of the planned traceability system (Figure 6). The primary focus was ensuring the system could operate effectively within a cold and challenging environment while integrating seamlessly with existing systems without requiring a complete overhaul.

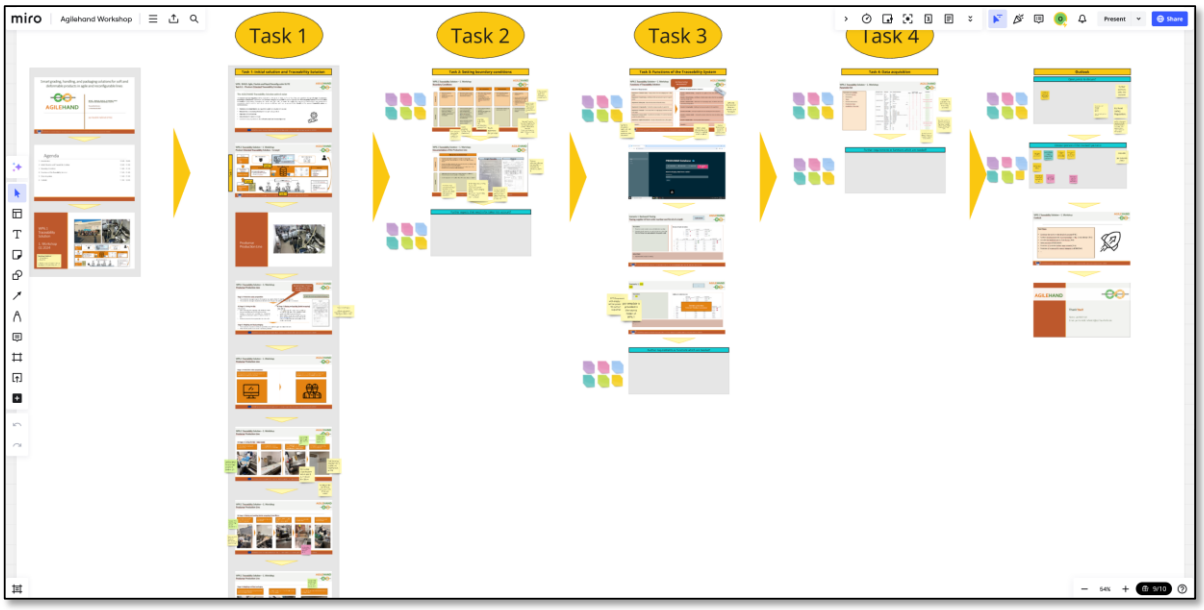

Figure 6: Workshop 2024 - Discussion on suitable requirements and functions for the traceability solution

Requirements for the traceability system include integrating sensors to monitor various parameters, as a significant percentage of the current processes are manual. Specific conditions for sensor operation must be established, and there is a need to eliminate printed documents in the production process documentation.

A comprehensive function list for the traceability system has been developed based on the defined requirements derived from literature, standard norms, and workshop discussions. The function list includes relevant parameters for the specific production steps. This list includes, on the one hand, general functions that are important to fulfill general traceability standards, norms, and regulations that are important to work on the global market. On the other hand, the list includes specific functions that should be developed to increase the productivity and security of the shop floor for the workers and support the decision process.

The planned traceability solution must comply with all defined standards, offering functionalities such as reporting and the ability to trace products backward and forward along the production line. The general functions will cover this.

Additionally, the system will incorporate specific functions to improve the production process. This includes providing specific information about stored batches in the freezing room, offering decision support related to these batches, and delivering insights on process step efficiency and defective products.

One practical example of the system's functionality is the Temperature Control Notification. This feature will provide real-time information about water temperature, track the duration that fish remain in the water bath, and issue early warnings to ensure that batches are removed promptly to maintain fish quality.

Relevant norms, standards, and regulations

For the development of a traceability solution, the following standards regulations, and norms are considered and will be described shortly:

- EC 178/2002: General Food Law (European Union): EC 178/2002 [1] is a regulation from the European Union that establishes the general principles (based HACCP 7 principles) and requirements of food law, aiming to protect public health and consumer interests. It includes provisions for food traceability, requiring businesses to be able to trace food, feed, and ingredients through all stages of production and distribution.
- HACCP (Hazard Analysis and Critical Control Points): HACCP [2] is a systematic approach to food safety that identifies, evaluates, and controls hazards that are significant for food safety. It ensures that food products are safe for consumption by preventing, eliminating, or reducing potential food safety hazards to acceptable levels.
- ISO 22000 Food Safety Management Systems: ISO 22000 [3] is an international standard for food safety management systems. It provides a framework for organizations in the food supply chain to ensure the safety of food products at every stage, from farm to fork. The standard integrates the principles of Hazard Analysis and Critical Control Points (HACCP) and incorporates risk management to ensure food safety from both a preventive and corrective approach.
- GST (Goods and Services Tax) Standard for Providing Unique Identifiers: The GST is a value-added tax applied to the sale of goods and services in many countries. Under the GST system, businesses are required to report transactions and pay taxes in a transparent manner. The GST system introduces unique identifiers like the GSTIN (Goods and Services Tax Identification Number), which helps in tracking goods and services through the supply chain. However, the standard provides a framework for creating meaningful and classified information carrying Unique Identifications with help of GPC brick codes. This helps encoding the sensors with information that contains minimum trackable information needs set by EC 178/2002 regulation.

For all the mentioned standards (ISO 22000, EC 178/2002, GST, and HACCP), the following types of data are commonly required for traceability:

- Unique Identifiers: (e.g., Batch numbers, Lot numbers, GSTIN)
- Supplier and Customer Information: To track the source and recipients of materials and goods (i.e., names and addresses).
- Production and Manufacturing Data: Dates and timestamps of specific steps in the production process.
- Test and Inspection Records: To ensure the quality and safety of the products.
- Monitoring and Control Data: For HACCP and ISO 22000, monitoring critical control points such as temperature is essential for food safety.

By implementing a robust traceability system for tracking and storing this data, organizations can meet regulatory requirements, maintain transparency, and ensure the safety and quality of their products throughout the supply chain.

Moreover, to develop the database structure, the IEC-62264 [4] standard was selected to map the pilot production line in a realistic and precise manner.

The IEC-62264: Enterprise-Control System Integration is an international standard that defines the framework for integrating enterprise systems with control systems in manufacturing. It provides guidelines for integrating business processes (such as inventory, order management, and scheduling) with production control systems (such as manufacturing execution systems and MES). The standard is designed to ensure that critical information flows seamlessly across all levels of an organization, from enterprise-level planning down to the control of individual machines on the production floor. It helps provide real-time visibility into production and quality, enabling better traceability of products and materials. It is crucial for maintaining high-quality standards, ensuring compliance, and facilitating product recalls when necessary.

Physical implementation of the traceability system

Produmar's use case requires monitoring the batches of frozen hake fish in the processing phase, including the steps previously presented. To track and record the temperature of the batches along these stages, an electronic system is being developed that will use a microcontroller with an embedded wireless network, battery, and temperature sensor encased in IP67 plastic protection, capable of enduring the low temperatures and high humidity of the operating conditions of Produmar's factory floor. Placed alongside each batch, the device, dubbed "Dori", will be able to communicate the positioning and sensor values and undergo the same manipulation as the actual fish stakes. The final intent is to have a Dori follow each batch in the production line and gather data. The collected information will be sent to a local (or remote) server by device using the MQTT protocol.

Front-end mock-up

The front-end development will focus on creating a user interface concept tailored to the needs of different user groups, such as production managers and process experts. This approach is crucial for ensuring users can easily access information relevant to their specific role. The software aims to enhance productivity and streamline workflows by presenting user-friendly information.

The interface will be designed to provide process-specific information to users, allowing them to make informed decisions quickly and efficiently. By prioritizing usability and clarity, the software will empower users to navigate the system effortlessly, ultimately leading to improved operational performance and collaboration among various teams.

The user interface will have four pages: Dashboard, Data Management, Production Line, and Production Line Details. The first sketch of the page design can be seen in Figure 7. Those pages' specific information and primary purpose will be explained below.

Figure 7: Sketch of the software mock-up for the traceability solution

Dashboard

The dashboard page has comprehensive information tailored to provide essential insights on production line productivity. It includes data from the last weeks, months, or even years, serving as a valuable resource for managers to review and analyze performance over time.

The main purpose of this page is to cater to the needs of those in managerial positions. By providing key information about general costs, efficiency, and processes on the shop floor, managers are equipped with the means to monitor and control these elements effectively. Furthermore, the transparency offered by the dashboard enhances understanding of shopfloor processes, thereby supporting decision-making and strategic planning.

Data Management (Batches)

The data management page is a comprehensive platform for viewing and managing batch information. It displays all registered batches in an easily digestible table format. Users can select a specific product from this table to view more detailed information in a corresponding card. Users can upload historical data through a properly formatted CSV file to expand the database. Currently, this is the only page connected to the back-end; however, additional pages are planned to be integrated in future development steps.

The primary purpose of this page is to assist production experts by providing crucial insights about batch processing. This includes forward and backward tracing of production lines and

visibility into supplier and customer details for each batch. With this wealth of information, production experts can manage and optimize their processes more effectively.

An updated design version for the data management page was developed and can be seen in Figure 8. The page name was changed to “Batches”.

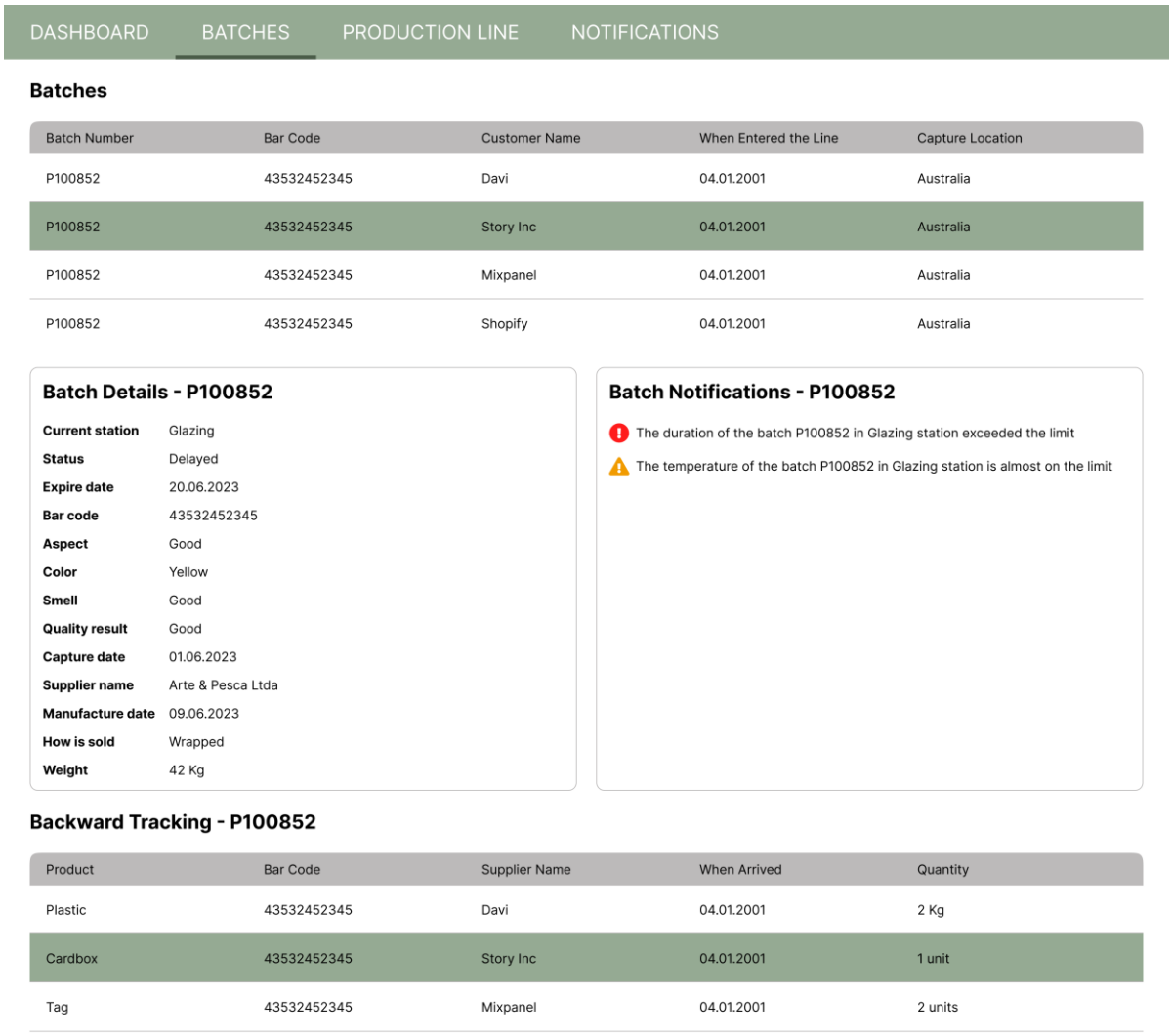

Figure 8: User interface mock-up - Batches page for the glazing step

Production Line

The production line page presents a comprehensive view of all stations across the entire production line, showcasing key production metrics such as process name, product being produced, capacity, temperature, prediction of leftover foil, average run time, longest run time, total product defects, remaining time, defective products per batch, stored batch numbers, yield per batch, and daily total yield. Users can also benefit from real-time alerts or warnings for any station and can navigate directly to the production line details page for more information.

The primary purpose of this page is to provide users with a high-level overview of the production line's status and performance. It facilitates a clear understanding of available stations and their current status, enabling efficient monitoring and management of the production process.

Production Line Details

The production line details page provides a detailed overview of the selected station. For instance, if the glazing station is selected, users can view key information such as the current batch, duration, time limit, and temperature. Besides, the page also highlights station-specific notifications, keeping users informed about crucial updates and alerts.

The primary purpose of this page is to present detailed information specific to each production station. Doing so offers valuable insights into the individual station's operations.

Demonstrator implementation

The implementation of the project consists of three main components:

- Front-end: Developed in JavaScript using the ReactJS framework, following componentization principles and Atomic Design methodology.
- Back-end: Implemented in Python utilizing the FastAPI framework.
- Database: Built with PostgreSQL.

The front-end communicates with the back-end through REST API requests. The process is as follows:

- Front-end API Call: The front-end sends a REST API request, including the necessary parameters.
- Back-end Request Handling: The back-end receives the request along with its parameters, and it performs the required logic based on the request.
- Database Interaction (if necessary): If the operation requires data retrieval or modification, the back-end connects to the database using the SQLAlchemy framework.
- Response to Front-end: The back-end processes the information and returns the relevant data to the front-end in response to the API call.

The front-end implementation has started based on the Figma design with the first three pages: Data Management, Production Line, and Production Line Details. In the back-end two endpoints currently implemented to call information from the database and provide it to the front end. These consist of an endpoint for calling specific information from the listed batches in the database. The batch list endpoint queries different tables like MaterialLot, MaterialDefinition, MaterialClass, MaterialClassProperty, and MaterialLotProperty and returns all products stored in the database along with their properties in a standardized response body. Furthermore, the second endpoint, which was developed, realizes the upload of external data from the company to enter easily to the database if the data is given in the structured format. The data upload endpoint accepts a file containing structured data, fulfills the database, and returns a success indicator (1) upon successful operation.

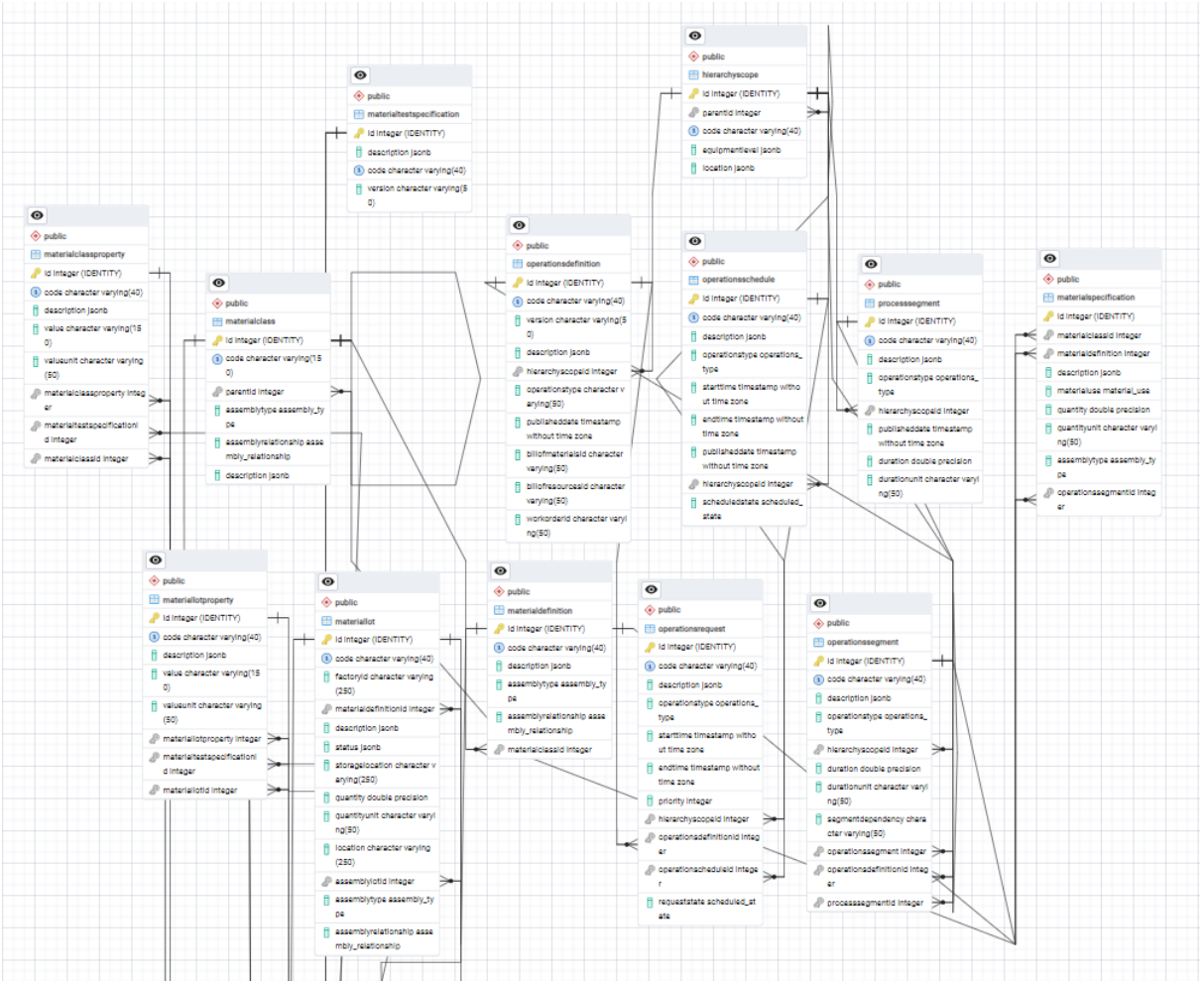

Figure 9: Implemented database concept V2 as a PostgreSQL database

Moreover, database V2 was constructed in a PostgreSQL database and can be accessed and maintained through the pgAdmin interface. The implemented database concept can be seen in Figure 9. Adapting to the database concept V3 will be one of the next steps.

Additionally, the implementation process will continue and involve the development of additional front-end screens and the integration of all front-end components with the back-end. Furthermore, the database will be updated according to the latest concepts and populated with Produmar's data.

Furthermore, the developed front-end and database will be validated with the experts and users from the pilots if the solution is user-friendly and can support and increase productivity in daily work.

To summarize the work done so far, the developed database concept ensures the alignment of the selected standards for tracing food products and ensures individual functions for small companies in that sector even when their factory is on a lower digitalization level. The developed front-end design should take care of the possibility of increasing the productivity and stability of the daily manual processes from the worker by providing individual functions as decision support. The developed solution will provide useful information to different factory experts and increase the transparency of production processes. Moreover, it will help by identifying causes for possible errors earlier.

2.2.2. Future Steps

Until the end of the project, the implementation of the third version of the database concept will continue. All discerned tables and parameters will be examined in relation to the existing parameters from the pilot. Afterward, the PostgreSQL database concept will be enriched with more product data from the pilot.
Furthermore, the analyzed and chosen sensors will be incorporated into the traceability solution, and their data will be collected. Parallel to these steps, the developed user interface mock-up will be actualized, and implementation will continue. It will be guaranteed that all defined functions are conveyed in a user-friendly manner, in addition to meeting the regulations and standards for product traceability.
Meanwhile, the work on the user interface's development and implementation in an application will proceed. After these activities, the solution developed specifically for the pilot will undergo a testing phase.

2.2.3. Functional and Implementation Viewpoint

In these sections, the overview of the functional and implementation viewpoints of the Production Traceability Solution task are represented.

2.2.3.1. Functional Viewpoint

The traceability system store and provide production data which are relevant for tracking and tracing products. The data will be provided to the user interface where the production manager und expert in the line can use the information as a decision support in their daily work.

Data structure of Traceability Solution

Format	Input / Output	Example
.json/.csv	Input	Acquiring data from the production systems (e.g., warehouse, orders, production line)
.json/.txt/.csv	Output	Production time and order processing

Table 1 – Data Structure

Software requirements

Software Component	Description / Role	Required Version / Configuration	Dependencies
Windows OS	Operating system needed to use the tool	Windows 10 Pro	N/A

Python	Programming language	Latest	N/A
Visual Studio Code	Editor	Latest	N/A
JavaScript	Programming language	Latest	N/A
Chrome	Browser	N/A	N/A

Table 2 – Software Requirements

Objects

Front-end: Design a graphical user interface (GUI) that is intuitive and user-friendly. The interface will enable users to interact with the system and display crucial information for tracing products along the production line for managers and shopfloor experts.

Back-end: Represent the functions which are feeding the front-end with the data from the database.

Database: Storing all relevant data for tracing and tracking products in the production line.

2.2.3.2. Implementation Viewpoint

A summary of the implementation of the database and user interface is given in the following tables.

Implementation Components	Description
Database	The database concept is implemented as a PostgreSQL database. The Software pgadmin4 was used to fill the database with information.
Back-end	The different functions will be developed to call information from the database and provide it over the front-end (user interface) to the user. First functions of calling batch information and uploading data are implemented.
Front-end	Display specific information to the user to provide decision support.

Table 3 – Implementation Components

In the following tables the specific implementation components will be described separately.

Implementation component	Database
Description of the implementation component	The second version of the database concept is implemented. The implementation of the third version is under development.
Used technology	Postgre SQL database
Technical Description of the Component	*Dependencies*

	Interfaces
	User Interface: PGadmin4

Table 4 – Database

Implementation component	Back end
Description of the implementation component	Creation of functions for calling information from the specific tables in the database. Creation of functions for providing new information based on the stored data to support the decision process of the worker.
Used technology	Python
Technical Description of the Component	*Dependencies*
	Development Language: Python
	Interfaces
	User Interface: Visual Studio Code

Table 5 – Back end

Implementation component	Front end
Description of the implementation component	Displaying information to the user. Getting input and feedback from the user and insert the content in the database.
Used technology	Python, REACT
Technical Description of the Component	*Dependencies*
	Development Language: Python, JavaScript
	Interfaces
	User Interface: Web Site

Table 6 – Front end

2.3. Transferability of the solution

The developed solution can be used for different companies in the food sector which are not fully digitalized, and which have manual processes in their production line. The solution is aligned with the given standards for tracking and tracing the product along the production line by following the given standards, norms and regulations. For implementation with other companies the selected tables and given properties can be individually used based on the given production line and conditions. Nevertheless, minor adaptation needs to be done to transfer the solution to other companies or use cases to make them as suitable as possible and gain the full benefits of the system.

By using a modular approach in the development of the user interface it was taken care that the general page style can be used and only small adaptations at the individual given functions needs to be done based on the new crucial production parameter in case of a new scenario.

3. AGILEHAND Data-Driven Digital Twin

3.1. Overview

This task focuses on developing a Digital Twin (DT) framework to enhance decision-making in managing production and logistics processes. It involves a data-driven approach to enable the automated generation of simulation models, forming the foundation for digital twins capable of real-time and near-real-time simulation, planning, and synchronization of production and logistics systems during line reconfiguration (Figure 10). The core aspect of DT implementation is the digital modelling activity, which is essential for running simulations. This was achieved by building a simulation environment through SimPy, an open-source framework for process-based discrete-event simulation written in Python. The framework was developed to function as a modelling tool, enabling the creation of a digital representation of the AGILEHAND pilots' production lines. It allows for simulations of the production line's behaviour under various operational scenarios. The results of these simulations, along with the processed information, are displayed on a user interface designed for end users. This interface allows customers to set simulation parameters, customize scenarios, and observe the results in real-time. This interactive capability is a decision-making support tool, enabling users to test different configurations, predict outcomes, and make informed choices to optimize their production systems effectively.

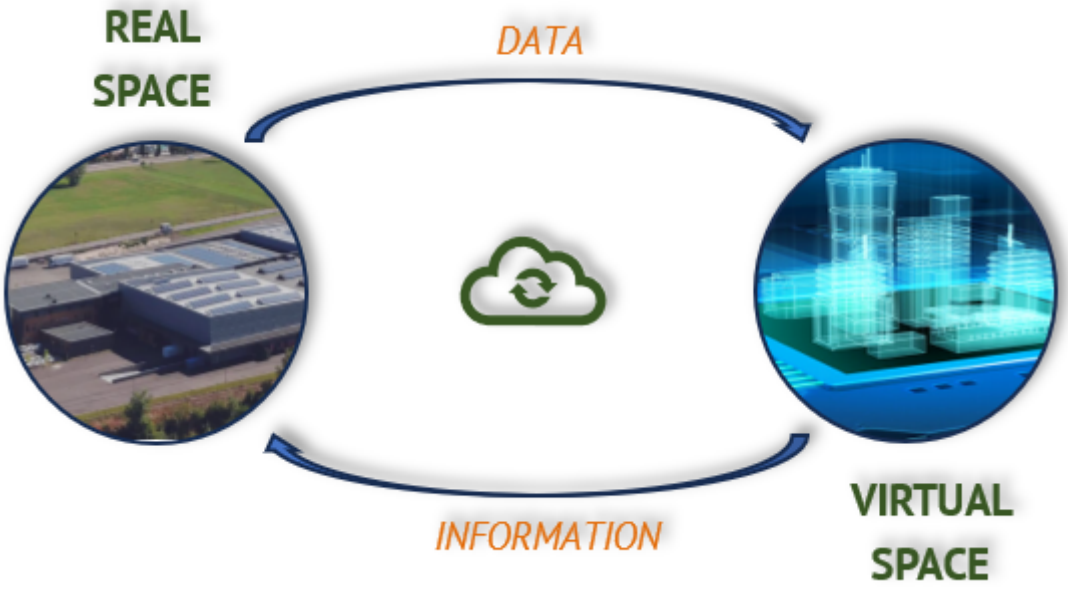

Figure 10: Digital Twin Environment

3.2. Design Status

The DT framework we developed consists of interconnected modules that support the system lifecycle. The Supervision module enables real-time monitoring, data analysis, control command issuance, and alert generation. The Connection Management module ensures secure communication between the DT and the physical system. The Simulation module replicates the system's behaviour, allowing scenario testing without impacting operations. The Monitoring module tracks performance using real-time data to identify trends and anomalies. The Optimization module adjusts production parameters to maximize efficiency and reduce costs,

interacting with the Supervision module for operator adjustments. Finally, the Adaptation module dynamically implements optimization recommendations, adjusting the system in real-time and updating the simulation model. The system modelling phase of the DT framework we developed employs the discrete event simulation (DES) technique, which was selected for its alignment with the company's physical structure. In this framework, physical entities are modelled according to DES principles, each with relevant attributes. Operators are modelled as renewable resources, with attributes defining their skills and assigned workstations. The team consists of standard operators and a team leader with additional responsibilities such as task delegation. Stations are modelled in a hierarchical structure, starting with a basic station that functions as a simple buffer. The Base Station class manages the flow of entities, handling their movement between stations as an extension of the basic station. Specialized station types, such as Process Stations and Conveyor Stations, are then developed, each adding specific functionalities to model production line operations. This hierarchical approach provides flexibility in defining stations of increasing complexity while maintaining a clear structure and functionality throughout the system.

3.2.1. Current Implementation

3.2.1.1. Sant'Orsola

The digital modelling phase for Sant'Orsola has been successfully completed using a Python-based framework, as previously described. This framework replicates and simulates the company's production processes under varying productivity levels. This activity focused on raspberry batching and handling operations and was specifically chosen as a case study within the scope of the AGILEHAND project. The production lines are divided into two operational categories: two lines operate using manual techniques to charge crates into the line. At the same time, the other two rely on automated machinery for this task. Each line typically operates with four assigned operators; however, the company occasionally reduces this number to three operators per line to optimize resource use and enhance efficiency. The setup of these lines is determined through demand forecasting, which is based on historical data from equivalent days in the same month of prior years. This analysis is conducted the day before production and is the basis for configuring the production lines. On the production day, the setup is executed according to the forecasted demand. As order confirmations are received throughout the production day, the accuracy of the forecast is assessed. Forecasting starts with preprocessing to clean and organize historical data, ensuring it is accurate and ready to use. Before any other steps, the Gradient-Boosted Trees (GBT) algorithm was chosen as the forecasting model because it performed well in initial tests and effectively handled the company's data. After selecting GBT, four database structures were tested to find the best way to capture demand trends and improve forecasting accuracy:

Row-Based Database: Tests were conducted using the GBT model to evaluate its performance with data structured in rows.
Column-Based Database: Tests used a dynamic version of the GBT model to analyse data structured in columns.
Weekday-Aligned Database: This structure incorporated weekly seasonality and was tested with a dynamic GBT model designed to handle these patterns.

Week-Based Database: This database also accounted for weekly seasonality but applied continuous standardizations and was tested with the dynamic GBT model.
Approximately 150 tests were conducted across these structures, leading to adjustments and optimizations that improved the model's performance. The forecast accuracy analysis significantly improved compared to the existing forecast system.

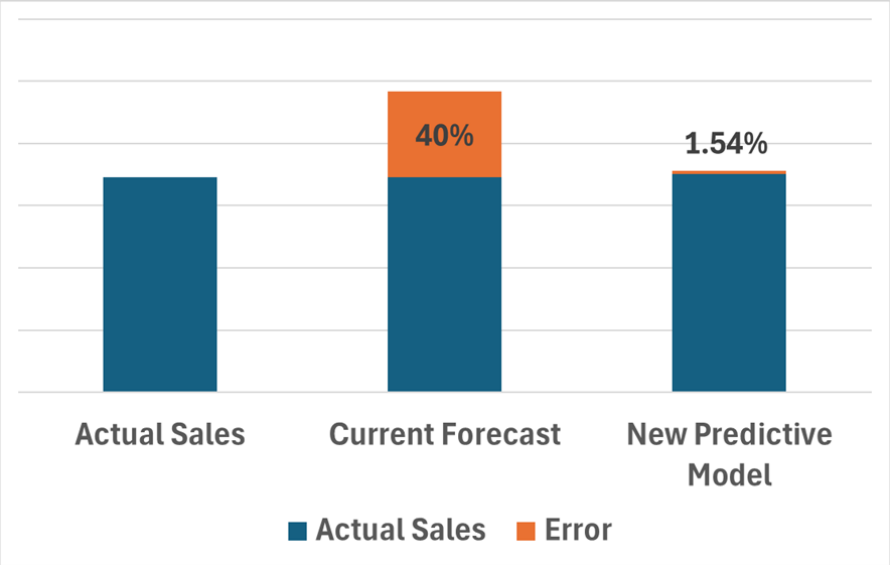

Figure 11: Current Forecast and New Predicive Model

Figure 11 showed a forecast error of about equal to 40%, whereas the new predictive model achieved a drastically reduced error rate of 1.54%. This demonstrates the model's superior capability to align production planning with actual demand, reducing inefficiencies and waste. The difference between forecasted demand and actual demand reflects an error in the forecasting process, which can lead to inefficiencies in production. Another critical challenge addressed through the modelling process was determining the optimal assignment of operators to each line and identifying the appropriate number of lines required to satisfy forecasted demand. The digital model was developed to replicate the behaviour of both manual and automated lines under configurations involving either three or four operators. It included detailed modelling of production line dynamics and extensive simulations to validate the model's accuracy and reliability based on current data. The model is designed to manage system dynamics in relation to critical parameter values and the occurrence of critical events.

PRODUCTIVITY	3 OPERATORS					4 OPERATORS				
	Expected	C3/C4 (not AUTO)		C1/C2 (AUTO)		Expected	C3/C4 (not AUTO)		C1/C2 (AUTO)	
		Real	ERR %	Real	ERR %		Real	ERR %	Real	ERR %
74	0h19m	0h16m37s	-10.14%	0h16m15s	-12.21%	0h15m	0h15m36s	5%	0h16m46s	13%
161	0h40m	0h34m36s	-14.05%	0h33m49s	-15.98%	0h32m	0h32m30s	1%	0h35m3s	9%
235	0h59m	0h49m59s	-14.93%	0h48m28s	-17.50%	0h47m	0h47m6s	0%	0h50m15s	7%
300	1h15m	**1h3m28s**	-15.37%	**1h1m33s**	-17.94%	1h0m	**0h59m41s**	-1%	**1h3m53s**	6%
427	1h47m	1h32m24s	-13.45%	1h29m57s	-15.74%	1h25m	1h27m53s	3%	1h34m7s	10%
511	2h8m	1h49m24s	-14.36%	1h46m33s	-16.59%	1h42m	1h44m11s	2%	1h51m22s	9%
757	3h9m	2h42m29s	-14.15%	2h38m49s	-16.08%	2h31m	2h34m46s	2%	2h45m22s	9%
1000	4h10m	**3h32m27s**	-15.02%	**3h27m15s**	-17.10%	3h20m	**3h21m57s**	1%	**3h37m7s**	9%
1499	6h15m	5h18m23s	-15.04%	5h9m36s	-17.39%	4h60m	5h2m21s	1%	5h20m32s	7%
2002	8h21m	7h8m6s	-14.46%	6h55m39s	-16.95%	6h40m	6h45m56s	1%	7h11m39s	8%
2665	11h6m	9h30m55s	-14.31%	9h13m41s	-16.89%	8h53m	9h0m29s	1%	9h36m6s	8%
3000	12h30m	**10h43m1s**	-14.26%	**10h23m41s**	-16.84%	10h0m	**10h8m18s**	1%	**10h49m19s**	8%
MEAN			-14%		-16%			2%		9%
STD. DEV.			1%		1%			1%		2%
REL. STD. DEV.			-10%		-9%			94%		21%

Figure 12: Simulation Results

Using a specific monitoring method built on the SimPy framework, the real system behaviour was successfully replicated across all possible configurations. This capability forms a foundation for simulating different scenarios and obtaining the production time required by the lines. Additionally, it allows system performance to be verified in both real-time and non-real-time contexts. Figure 12 compares expected and real times across different configurations (manual vs. automated, 3 vs. 4 operators) and productivity levels. The real columns show actual simulation results, while "ERR%" quantifies deviations from expected values. Automation generally improves accuracy (lower "ERR%"), and higher operator counts reduce variability. However, at higher productivity levels, deviations increase, indicating the need for further model fine-tuning. The summary statistics (last three rows) evaluate bias (average ERR%), consistency (absolute standard deviation), and relative variability (relative standard deviation), providing an overall assessment of model reliability. These metrics highlight the accuracy and precision of the simulation model: the average deviation reflects the model's general bias, while the standard deviations measure the consistency of results across scenarios. Currently, the expected values serve as generic benchmarks to compare with simulation results. However, the objective is to refine the model further to make it increasingly realistic and precise, reducing discrepancies and improving its alignment with real-world data. This refinement supports decision-making activities to optimize production and resource allocation to the lines. The resulting digital model provides valuable insights into optimizing resource allocation and production strategies. Beyond enabling the company to meet demand efficiently while minimizing operational costs, the model also minimizes forecasting errors, directly reducing waste in terms of time, operational costs, and raw materials.

3.2.1.2. Marelec

The most important station, where fillet conveyors from two lines feed into an automated station consisting of 30 splitters (Figure 13). At this station, a flow scale weighs and grades each product individually, ensuring a continuous throughput of raw materials. After the splitter machine processes 400 products per minute, the material is divided and weighed into 30 sub-batches of 1, 2, 4.5, or 5 kilograms each. Each batch consists of fillets within a narrow weight range, depending on the order's specifications. Typically, 4 to 6 sub-batches will be defined with the same batch size and specific fillet range. Fillets that fall outside the required range are sent to the end of the line, where a floating operator removes the whole box.
This station is the company's core, as it allows the production manager to decide which type of chicken to select for entry into the four lines, whether to separate the fillet from the inside and how to configure the splitters. This process ensures that the chickens selected correspond to the weight range required by the customer, allowing orders to be processed on time. Once the process is complete, the station sends the batch information to the pre-labelled trays station, and the product moves to the next stage.

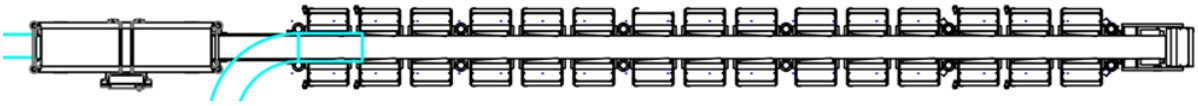

Figure 13: Splitter Station

To address this issue, a program was developed to collect and analyze warehouse data, identifying the types of chicken available and their respective suppliers. At the same time, the program processes customer order details, including the required weight range of fillets and the total quantity for each order (as shown in Figure 14).

WAREHOUSE		
SUPPLIER	CHICKEN WEIGHT	QUANTITY
Supplier A	CK 400g - 550g	15000
	CK 550g - 750g	8500
	CK > 750g	7500
Supplier B	CK 400g - 550g	9000
	CK 550g - 750g	5000
	CK > 750g	6500
Supplier C	CK 400g - 550g	9000
	CK 550g - 750g	10000
	CK > 750g	7000
Supplier D	CK 400g - 550g	8000
	CK 550g - 750g	9000
	CK > 750g	7000

ORDERS				
CUSTOMER	N° TRAYS	BATCH WEIGHT	BATCH WEIGHT	BATCH WEIGHT
Customer A	4000	5000	150	180
Customer B	5500	4500	180	210
Customer C	5000	2500	210	240
Customer D	3000	1000	240	270

Figure 14: Warehouse and Orders Data

By matching this data through an overlap analysis, the program determines the number of chickens needed, specifying whether they should be processed with or without inner portions and identifying the most suitable category to meet each request. This enables precise calculations of the required chickens and their attributes, as well as verification of stock availability. The results

are then integrated into a simulation, where chickens are allocated to different production lines based on the required configurations, optimizing the production process.

Customer	Completed Trays	Total Trays	Missing Trays	Used Chickens	Typology Used	Check
Customer A	2000	2000		34482	CK400-550 WITH	OK
Customer A	2000	2000		31250	CK550-750 WITHOUT	OK
Customer B	113	5500	5387	1250	*CK550-750 WITH	NOT OK
Customer C	0	2500	2500	0	*CK550-750 WITH	NOT OK
Customer C	2500	2500		14534	CK>750 WITHOUT	OK
Customer D	3000	3000		5554	CK>750 WITH	OK

Figure 15: Orders Status Results

Based on the Figure 14, Figure 15 provides a clear overview of each customer's order status, including the number of trays completed, the type of chicken used, and whether the orders have been successfully completed.

With these results, it becomes possible to configure the splitter, a critical component of the plant. For this purpose, a dedicated model was developed that utilizes order details, such as fillet weight range, tray weight, and the required number of trays, in conjunction with the previously processed data.

The model adjusts the splitter settings according to these parameters, tracking tray production. Once the required number of trays for an order is produced, the splitter automatically reconfigures itself to process the next order in the queue.

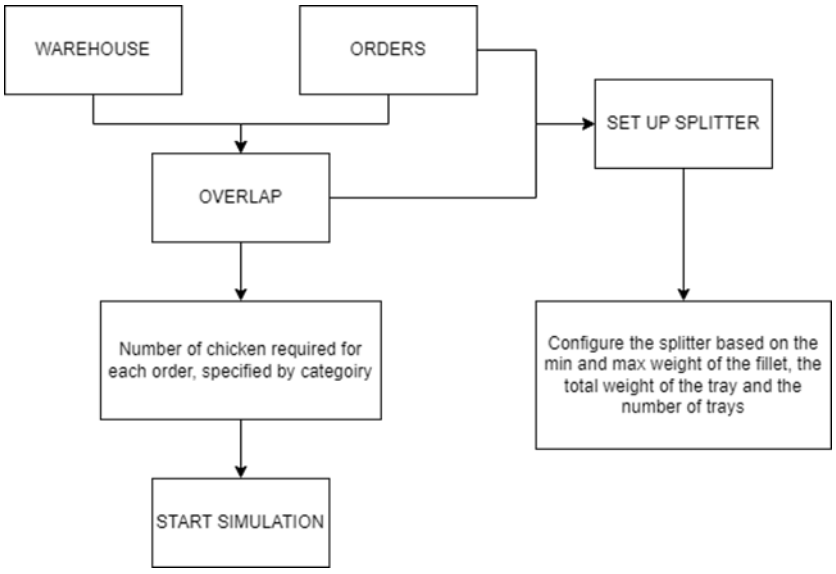

Figure 16: Model Functioning Flowchart

Figure 16 represents all the connections between the models to obtain the results.

3.2.2. Future Steps

The model will be shared with both pilot plants to validate its functionality and refine its performance. This process will incorporate feedback from real-world operations to enhance the model's accuracy and reliability. Simultaneously, efforts will focus on finalizing the user interface to ensure it meets usability standards and aligns with the requirements of end users. Furthermore, a direct connection between the plant systems and the model will be established, enabling real-

time data sharing. This integration will support continuous updates and improve the model's responsiveness to dynamic plant operations.

3.2.3. Functional and Implementation Viewpoints

In these sections, the overview of the functional and implementation viewpoints of the Data-Driven Digital Twin for Production and Logistic System Synchronization tasks are represented.

3.2.3.1. Functional Viewpoint

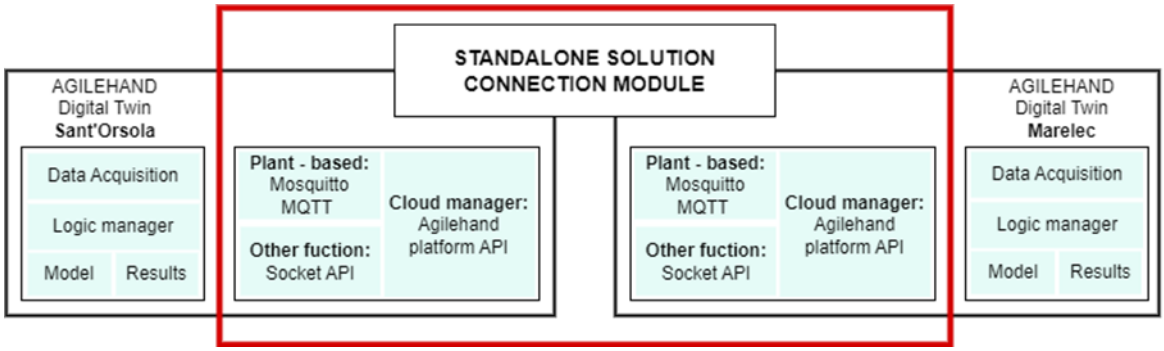

Figure 17: Functional Viewpoint

The functional viewpoint (Figure 17) of the Data-Driven Digital Twin suite is presented in the figure. Digital Twin collects data from the company information system, including production, operator, supplier, and customer information. It stimulates the production process while accounting for uncertainty and variability. The model then analyses the simulation results to enhance demand forecasting and improve the estimation of the order's expected time of arrival (ETA).

Data structure of Digital Twin

Format	Input / Output	Example
.json/.csv	Input	Acquiring data from the plant (e.g., warehouse, orders)
.json/.txt/.csv	Output	Production time and order processing

Table 7: Data Structure

Software requirements

Software Component	Description / Role	Required Version / Configuration	Dependencies
Windows OS	Operating system needed to use the tool	Windows 10 Pro	N/A
Python	Programming language	3.11	N/A

Visual Studio Code	Editor	Latest	N/A
JavaScript	Programming language	Latest	N/A
Html	Programming language	Latest	N/A
Node.js	Programming language	Latest	N/A
Chrome	Browser	N/A	N/A
Mosquitto MQTT	Back-end system that coordinates messages between different clients	Latest	N/A

Table 8: Software Requirements

Objects

Front-end: represent the user interface and the presentation layer.

Native user interface: Design a graphical user interface (GUI) that is both intuitive and user-friendly, featuring functional menus. The interface will enable users to run simulations, monitor production processes in real-time, customize various parameters based on the requirements of the production manager, and visualize the results effectively.

Back-end: represent the application logic and the server layer.

Native Solution Component

Core Functions: Data Acquisition, Logic Manager, Model Simulation and Results.

Connection Manager: Network socket.

3.2.3.2. Implementation Viewpoint

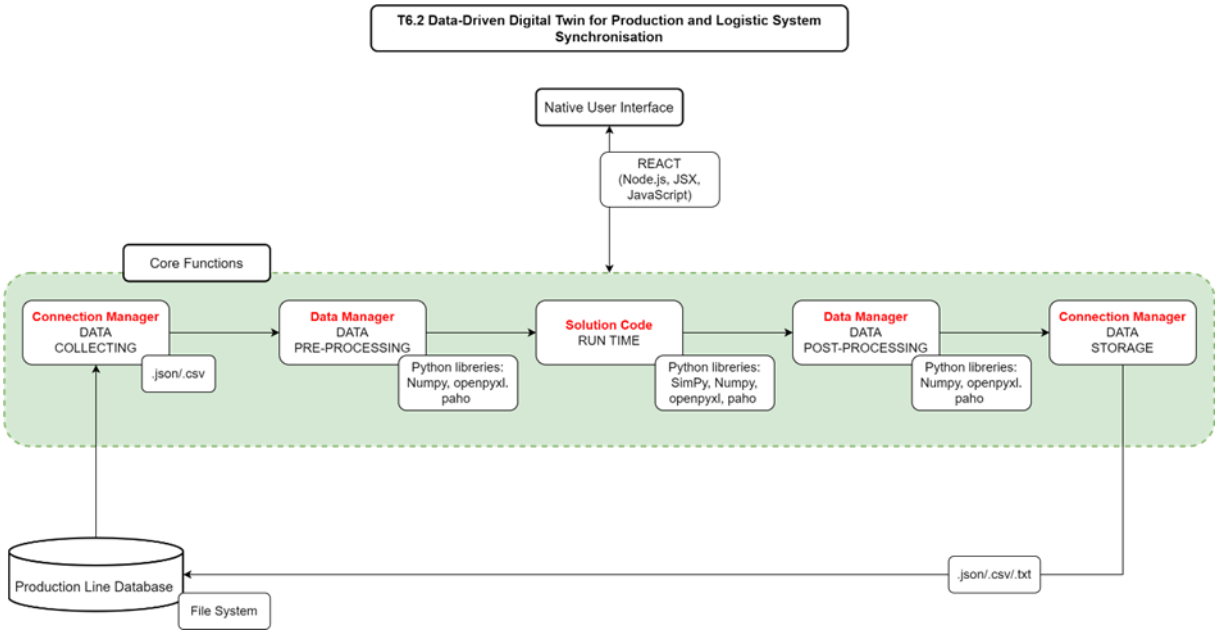

Figure 18: Implementation Viewpoint

The implementation architecture represented in the figure presents an overview of the implementation components, which are detailed in the following tables.

Implementation Components	Description
Data Collecting	Acquires input data from the company's MES, ERP, and PLC systems to gather all relevant information related to production, operators, suppliers, and customers.
Data Pre-Processing	Serves as a data processing step to enable further processing stages. For example, historical customer order data is organized by year, season, and other key parameters to classify the orders.
Run Time	Serves to generate accurate forecasted input demand and simulate various scenarios based on the desired productivity.
Data Post-Processing	Serves to obtain the optimal solution from the previous step and set the production lines.
Data Storage	Serves to store simulation results.

Table 9: Implementation Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required for implementing each component.

Implementation component	Data Collecting
Description of the implementation component	Acquires input data from the company's MES, ERP, and PLC systems to gather all relevant information related to production, operators, suppliers, and customers.
Used technology	Python, REACT
Technical Description of the Component	*Dependencies*
	Development Language: Python, JavaScript Libraries: Numpy, Openpyxl, Paho-mqtt, Json, CSV
	Interfaces
	User Interface: Web Site Network/Protocols: HTTP/MQTT

Table 10: Data Collecting

Implementation component	Data Pre-Processing
Description of the implementation component	Serves as a data processing step to enable further processing stages. For example, historical customer order data is organized by year, season, and other key parameters to classify the orders.
Used technology	Python, REACT
Technical Description of the Component	*Dependencies*

	Development Language: Python, JavaScript Libraries: Numpy, Openpyxl, Paho-mqtt
	Interfaces
	User Interface: Web Site Network/Protocols: HTTP/MQTT

Table 11: Data Pre-Processing

Implementation component	Run Time
Description of the implementation component	Serves to generate accurate forecasted input demand and simulate various scenarios based on the desired productivity.
Used technology	Python, REACT
Technical Description of the Component	*Dependencies*
	Development Language: Python, JavaScript Libraries: Numpy, Openpyxl, Paho-mqtt, SimPy
	Interfaces
	User Interface: Web Site Network/Protocols: HTTP/MQTT

Table 12: Run Time

Implementation component	Data Post-Processing
Description of the implementation component	Serves to obtain the optimal solution from the previous step and set the production lines.
Used technology	Python, REACT
Technical Description of the Component	*Dependencies*
	Development Language: Python, JavaScript Libraries: Numpy, Openpyxl, Paho-mqtt, SimPy
	Interfaces
	User Interface: Web Site Network/Protocols: HTTP/MQTT

Table 13: Data Post-Processing

Implementation component	Data Storage
Description of the implementation component	Serves to store simulation results.
Used technology	Python
Technical Description of the Component	*Dependencies*
	Development Language: Python, JavaScript Libraries: Json, Txt, CSV

Table 14: Data Storage

3.3. Transferability of the solution

The DT framework is designed to be both scalable and transferable, offering significant flexibility for use across different industries and production systems. Its modular architecture allows for easy adaptation to various production lines, whether manual, automated, or hybrid. The framework's scalability is supported by its use of Python and open-source libraries like Numpy, SimPy, and Paho-mqtt, which enable it to handle larger datasets, complex simulations, and integrate additional data sources or machines as the system grows.

In terms of transferability, the DT framework can be easily applied to other manufacturing environments by adjusting operational parameters, such as simulation models and data acquisition methods. Components like Data Collection and Pre-Processing are based on standard protocols (HTTP/MQTT) and common data formats (JSON, CSV), making them easily adaptable to various factory setups. The modular structure of the framework allows for quick reconfiguration to accommodate different product flows, forecasting models, and production parameters, ensuring minimal adjustments when transferring to other industries.

Furthermore, the real-time data sharing and adaptive control mechanisms in the framework can be transferred to other systems by modifying the Connection Management module to support various communication protocols. The integration of cloud-based solutions or external platforms further enhances its scalability and transferability to larger and more distributed production environments. The user interface is customizable, making it suitable for different industries or operational needs.

4. AGILEHAND Intelligent Rapid Configuration of Production System

4.1. Overview

This task aims to implement advanced predictive models designed to assess the quality of product batches, focusing specifically on items such as chickens and raspberries. In addition, the project will develop models aimed at making initial quantity predictions. To achieve these goals, various individual data sources will be integrated, including open API weather data, historical production records, and essential information from suppliers and customers.

The motivation to drive this initiative is to ensure product quality early in the production process. By harnessing the power of data analytics and sophisticated machine learning techniques, the project aspires to enhance operational efficiency and significantly reduce waste, ultimately fostering a more sustainable production environment.

This initiative's anticipated outcomes include improved quality assurance and process optimization and the promotion of data-driven decision-making. The implemented models will effectively support factory operations during the critical production planning phase by providing valuable insights and predictive capabilities. This comprehensive approach will lead to a more efficient and effective production process, ensuring high-quality products meet market demands while minimizing resource expenditure.

The Figure 19 The overall concept for the solution of predicting the product quality and quantity is described below. Therefore, individual data sources from the different suppliers will be provided and preprocessed to feed the developed models. There are two main prediction models: one for quality prediction and the other for quantity production. Both are connected to the prediction output from the other model as an input. The exact procedure and connection of the selected models will be described in the pilot-specific development chapters.

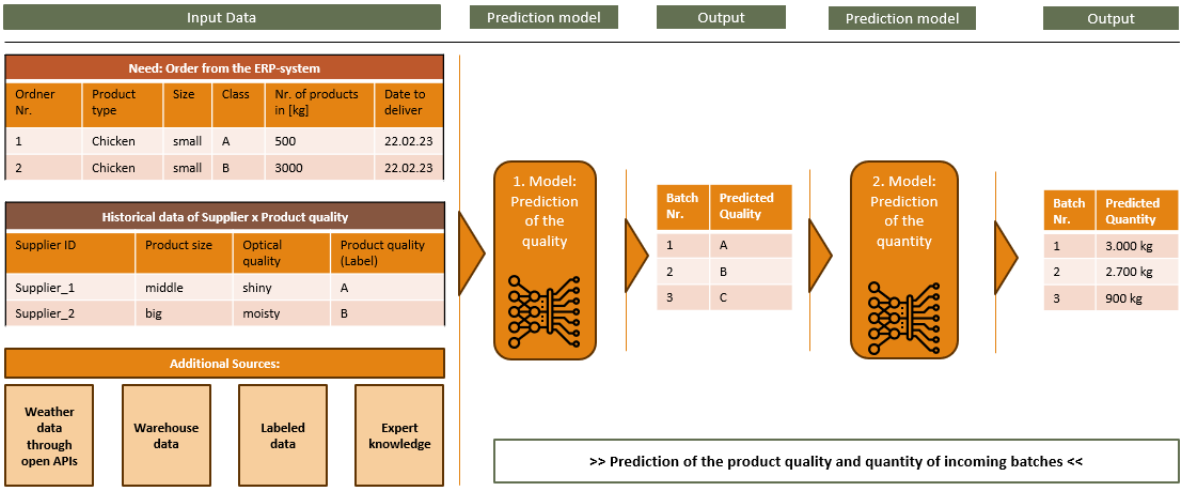

Figure 19: Big Picture of the concept for the product quality and quantity prediction

4.2. Design Status

This paragraph will present the different development stages of the solutions for the various pilots. First, the identified pilot-specific job optimization scenarios are presented. After that, the

general requirements and boundary conditions for the solution are defined. Moreover, detailed status information about the progress made in development is given. Finally, the planned steps for future growth will be presented.

Identified job optimization scenarios at the pilots

1.Sant'Orsola

Predicting fruit quality across varieties: The primary aim is to utilize advanced predictive analytics to analyze a comprehensive array of production parameters. The project seeks to forecast the quality of fruit produced across different varieties by doing so. This forecasting can significantly impact various facets of agricultural production, from refining cultivation practices to optimizing post-harvest treatments and strategically positioning products in the market. Such insights are crucial for maintaining high product quality and achieving competitive advantage.

Predicting production quantity for specific fruit varieties: These objectives target forecasting production quantities for designated fruit varieties using historical data and current production inputs. The aim is to align production with market demand efficiently, manage planting schedules, optimize resource allocation, and mitigate the risks associated with fluctuations in supply, such as overproduction or supply shortages. Effective forecasting can enhance supply chain optimization and prevent financial losses caused by mismatches in supply and demand.

2.Marelec

To streamline production operations, the quality of chicken will be predicted by analyzing historical trends, production data, and supplier information. Additionally, models will be developed to forecast the total quality of chicken that will be delivered by suppliers, including the proportion of Quality A and Quality B chickens. These predictive insights will enhance operational efficiency, improve decision-making, and reduce waste across the production line.

General requirements and boundary conditions

Different requirements and boundary conditions were set for the selected pilot, which will be described below.

1.Sant'Orsola

Dataset: The dataset for this pilot includes various details such as the type of fruit, its variety, the location and altitude at which it was grown, planting data, and harvesting details. Furthermore, data from existing APIs, like temperature, daylight duration, and wind speed are used in order to enhance the accuracy of the predictive models.

Yield and quality forecasting: The goal is to develop models that can forecast the fruit yield's quantity and quality. The quantity is measured in total kilograms, and the quality is according to a specified quality code.

Historical data utilization: Historical data is a significant resource for this use case. Past production details, environmental data, and trends are leveraged to train the predictive models to be as accurate as possible.

Scalability: Moreover, it should be ensured that the developed solution can scale to other similar products. The solution should be designed in a way that allows for expansion to other fruits, regions, or additional data sources.

2.Marelec

Dataset: The dataset includes basic supplier identifiers and key production metrics necessary for robust predictive analysis. Information related to product quality, supplier performance, weight measurements, and operational variables is essential for deriving actionable insights. It is critical to ensure that the dataset is complete and high-quality, comprising consistent data provided by different suppliers.

Yield and Quality forecasting: Development of predictive models that can estimate the quantity and quality of the chicken.

Historical data utilization: Marelec will provide the supplier with quality distributions throughout the year. This data will be utilized to train prediction models.

Scalability: The solutions should be adaptable across various production lines in the chicken packaging industry. Additionally, integration of the predictive models with existing management systems is essential. Furthermore, transferability to similar products should be ensured by adapting the solution.

4.2.1. Current Implementation

The current implementation will be described based on the selected pilot use cases and are presented below.

4.2.1.1. Sant'Orsola

Overview of the Pipeline

A comprehensive data processing and predictive modeling procedure for forecasting both the quantity and quality of harvested agricultural products, specifically focusing on raspberries, was developed and is given in Figure 20. The procedure is based on CRISP-DM, a common approach for developing machine learning pipelines. Below a concise explanation of the developed procedure is described:

1. **Business understanding:** Definition of the main objectives to develop the solution.
2. **Data acquisition and data understanding**: The process begins with the collection of raw data, which is then converted into a weekly format suitable for analysis.
3. **Data preparation**: In the data preprocessing phase relevant data is gathered via APIs, likely to include real-time environmental and operational data. After that, the data is then cleaned and pre-processed to handle missing values, normalize numerical data, and encode categorical variables, ensuring it's ready for modelling.
4. **Modeling:**
   Quantity prediction: In this phase, the prepared data, including field characteristics, weather data, and information about the growers, is fed to different machine learning

models. Moreover, the selected algorithms like XGBoost analyze the data to predict the total quantity of harvested products, denoted as 'Harvested kg tot'. **Quality prediction**: For the quality prediction, the output from the quantity prediction model serves as an input for the quality prediction model, supplemented by additional environmental and temporal data. By using advanced machine learning models, the aim was to predict specific quality metrics of the harvested products, initially focusing on a primary quality metric (Quality_1). After predicting the primary quality metric, the system uses similar methodologies to predict other quality aspects (e.g., Quality 2, 3, 4, 5) of the harvested products.

5. **Evaluation**: The different results of the models developed will be validated.

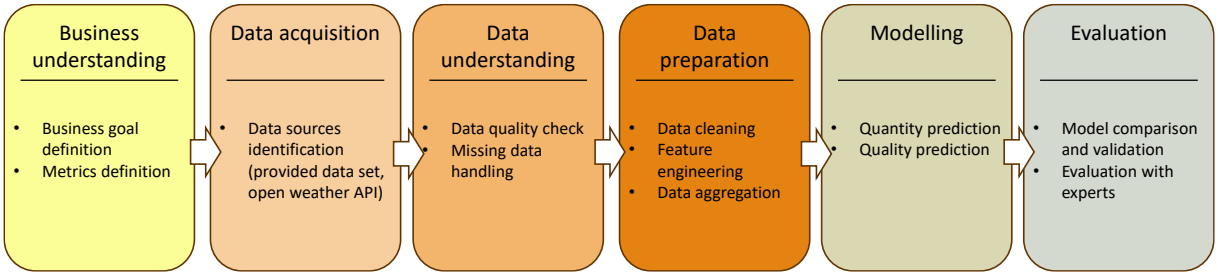

Figure 20: Process Procedure for product quality and quantity prediction of the Sant'Orsola pilot

Data understanding

The dataset from SantOrsola contains data from January 2022 to May 2024. It encompasses a wide array of data points from basic identifiers like the Name of Fruit and Variety of Fruit, to more complex agricultural and geographical metrics such as Planting Density, Geographical Location, and Altitude where the crops are cultivated. This dataset not only tracks the crop's lifecycle with data points like Year of Planting and Harvesting Date but also includes regulatory and operational details such as the Allowed Harvesting Period and Grower ID, facilitating a comprehensive analysis of agricultural practices and their outcomes.

Moreover, additional features such as Kg Per Quality (total weight of delivered products from an individual supplier from a specific field) and Quality Percent (Prediction of the percentage value of a specific quality class from the KG Per Quality) offer insights into the harvest's quality, allowing stakeholders to assess and enhance product standards. By documenting both environmental conditions and operational practices, the Sant'Orsola dataset provides a foundational tool for predictive modelling, aimed at optimizing yield forecasts and improving fruit quality across various geographical and climatic conditions.

Data is also sourced through Open APIs, complementing the Sant'Orsola dataset. These APIs facilitate the integration of real-time environmental data, such as temperature, precipitation, and humidity levels, directly influencing raspberry cultivation. The real-time data acquisition through APIs ensures that the models can adjust predictions based on current conditions, making them more dynamic and responsive to environmental changes.

Additionally, the project involves newly generated features based on existing data. These features are engineered to uncover deeper insights and hidden patterns within the data, enhancing the predictive models' accuracy. By analyzing interactions between different variables and creating

indices or composite metrics, these new features help capture more complex aspects of the data that standard analyses might overlook.

Data Preparation

In the data preprocessing phase, two pipelines are developed to prepare the data to predict product quantity and quality.

Product quantity:

The developed preparation pipeline for the product quantity prediction is divided into 6 steps and is shown in Figure 21.

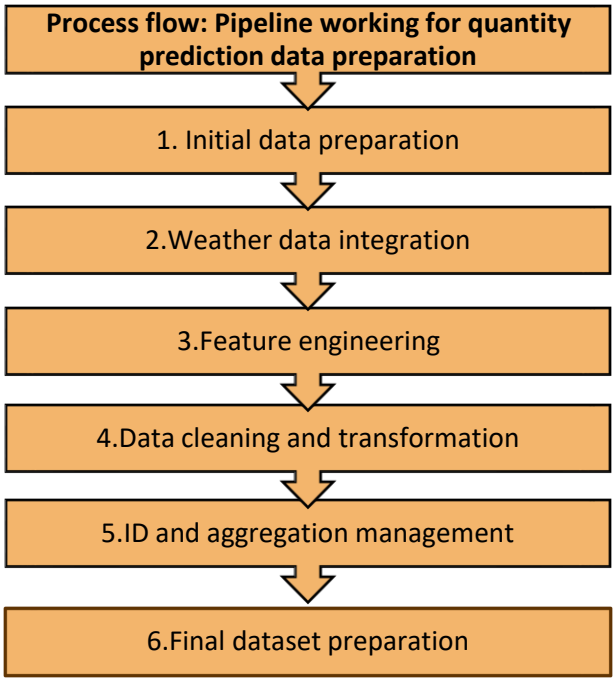

Figure 21: Preprocessing pipeline for quantity prediction for Sant'Orsola pilot

- **Initial data preparation**: Initial data preparation enhances the granularity of raw agricultural data and appending timestamp data. The timestamp facilitates future merging with weather data.
- **Weather data integration**: The process further involves partitioning the dataset based on geographical location to integrate localized weather data. The steps engaged in this phase include fetching weather data corresponding to each location and harvested dates from a weather API. The retrieved data is subsequently integrated back into the main dataset.
- **Feature engineering:** The Feature Engineering stage involves calculations for the 'Harvesting Period' and 'Total days for harvesting'. Additionally, synthetic columns such as 'Peak Status', or 'Variety Peak Status' are introduced into the dataset to capture peak conditions.
- **Data cleaning and transformation:** This stage detects and removes outliers in the dataset. The process involves rounding numerical data to ensure consistency, filtering valid entries, and extracting detailed temporal analysis from the harvested data by subdividing it into yearly, monthly or daily data.
- **Aggregation management**: The process assigns unique identifiers for growers, yielding a streamlined supplier tracking system. Additionally, quality metrics, including 'Quality Percent', or 'Kg Per Quality', are transformed for better representation. The process averages daily entries and eliminates duplicates to ensure dataset uniqueness and accuracy.
- **Final dataset preparation:** The concluding process requires final checks to certify data integrity and readiness. The data are then segregated into training and testing sets for model validation.

The final data set of parameters utilized for predicting the quantity of raspberry production can be categorized into three main sources: Data set from Sant'Orsola, Open access API data, and Generated features. Each category plays a crucial role in forming a comprehensive data foundation for the predictive models.

The data set from Sant'Orsola includes essential field-specific variables such as field size, number of plants, planting density, and temporal factors like month, week, the specific harvesting day, Location and Altitude.

The open access API data: Provides environmental conditions that directly influence agricultural outcomes, including maximum temperature (measured at a standard height of 2 meters), daylight duration, relative humidity, and total sunlight duration.

The generated features include operational and geographic variables derived from existing data to enhance model insight, such as harvesting period, days from allowed start to harvest, total days of harvesting and ProductionCatagory.

Product quality:

The data preparation pipeline for the product quality prediction is described below and consists of 4 phases (Figure 22):

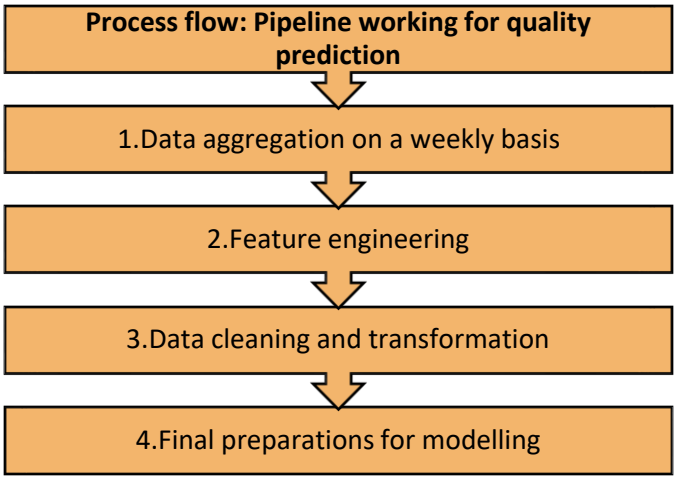

Figure 22: Preprocessing pipeline for quality prediction for Sant'Orsola pilot

- **Data aggregation on a weekly basis**: The process begins with data aggregation every week—summarizing daily observations into weekly intervals consisting of encompassing aggregate environmental conditions and operational metrics. The mean of quality percentages from quality percent 1 to 5 for each week is calculated, providing an average quality percentage for each category per week. In total 3696 data points for the weekly quality prediction could be used.
- **Feature engineering**: The next step is creating synthetic features that provide additional insight into quality predictions. Examples include 'Status Week Quality_1', a feature derived from weekly averages, and lagged features that incorporate the quality percentage of the previous week to consider time dependencies in quality evolution.
- **Data cleaning and transformation**: The data cleaning phase addresses any data gaps, especially noticeable in aggregated datasets. It involves the application of imputation techniques or algorithms that can handle missing values. Normalization or standardization is also applied to the dataset to normalize data ranges, leading to better performance for models sensitive to input scales, such as neural networks or distance-based algorithms.
- **Final preparations for modelling:** Final preparations for modelling include the elimination of any duplicate entries ensuing aggregation, ensuring unique data instances for model training. The partitioning of the processed data into training and testing sets, with the potential inclusion of stratification, ensures quality categories' proportion consistency across splits.

- **Modelling**

For the raspberry production forecasting project, the approach involved evaluating a broad range of predictive models to identify the most effective techniques for forecasting quality and quantity metrics. The selected models include XGBoost, AdaBoost, Random Forest Regressor, and AdaBoost enhanced with Optuna optimization.

- **XGBoost**: Known for its high performance in regression tasks, XGBoost was chosen for its computational efficiency and capability to handle diverse features efficiently. This model was pivotal due to its feature importance functionality, which is essential for identifying key factors influencing raspberry production.

- **AdaBoost**: Implemented both as a standalone model and in conjunction with the Optuna optimization framework, AdaBoost was selected for its ability to sequentially correct its predecessors' errors, focusing on challenging instances that were misclassified previously, thus refining the overall accuracy.
- **ANA Model (Artificial Neural Network)**: Introduced to leverage its ability to model complex, non-linear interactions within agricultural data, providing deep learning capabilities that capture intricate patterns often missed by traditional models.
- **Random Forest Regressor**: Chosen for its robustness in handling overfitting and its adeptness at processing large datasets with high dimensionality, Random Forest uses an ensemble of decision trees to ensure stable and reliable forecasts, making it well-suited for agricultural data complexities.
- **AdaBoost with Optuna**: This configuration was explored to optimize AdaBoost's parameters like the number of estimators and learning rate, ensuring optimal model settings. Optuna provides a systematic approach to hyperparameter tuning, enhancing model performance in predictive accuracy.

Hyperparameter Tuning and Optimization: To ensure optimal performance, various strategies were employed to tune the models' parameters:

- **Bayesian Optimization**: Utilized for its probabilistic model-based approach to finding the optimal model parameters, particularly effective in tuning the complex configurations of ANA models and XGBoost model.
- **Optuna**: Implemented for its efficient and flexible architecture that facilitates the quick and thorough exploration of parameter spaces across various models, including AdaBoost.
- **Grid Search**: Applied primarily to simpler models or when model parameters and their interactions are well-understood, ensuring exhaustive search through the predefined parameter grid.
- **Random Search CV**: This CV is used for its randomness in selecting parameters, often leading to discovering new and effective model configurations. It is especially useful in expansive parameter spaces typical in neural networks.

Model Training: The models were trained using a dataset compiled from various sources, including the SantOrsola dataset, open API for real-time environmental data, and internally generated features. The dataset was preprocessed to handle missing values and normalize and encode categorical variables for better model performance.

- **Data Preprocessing**: This step involved normalization and one-hot encoding to prepare the dataset for efficient model training. For instance, features like geographical location and variety of fruit were encoded to transform qualitative data into a machine-readable format.
- **Parameter Tuning**: The models were fine-tuned to identify optimal configurations using techniques such as grid search for Random Forest and Bayesian optimization for AdaBoost via Optuna. This step was crucial to balance model complexity and prediction accuracy.

- **Evaluation**

In the evaluation phase, the developed predictive model's accuracy, reliability, and practical applicability will be analyzed. This phase ensures that the developed models perform well under controlled conditions and real-world scenarios, providing valuable insights that drive decision-making processes in raspberry cultivation and management. To validate the models, k-fold cross-validation is employed, which helps ensure that the models generalize well to new data. This method also helps identify potential overfitting issues.

- **Cross-validation**: Each model was evaluated using multiple subsets of the data to verify consistency across different data splits. This method provided a robust measure of model performance and further helped fine-tune model parameters.
- **Performance Metrics**: Models were primarily assessed using RMSE and R^2 metrics. These metrics helped quantify the models' accuracy and the proportion of variance in the dependent variable that they could explain, respectively.

Results to predict the product quantity:

The actual results of the product quantity prediction will be described below.

DS001 - Original Dataset Analysis

- **Description**: The dataset consists of only original data which is preprocessed.
- **Results**: The models do not perform as well on the test set, suggesting issues with either the model's ability to generalize or the adequacy of the feature set and model complexity. The models exhibit a moderate R^2, indicating moderate predictive accuracy.

DS013 - Combined Synthetic and Original Data (2016-2024)

- **Description**: Using a synthetic dataset and merging it with the original data set (synthetic and original data) with a span of 2016 to 2024.
- **Results**: The initial model demonstrates satisfactory training performance but achieves lower testing performance. After Bayesian optimization, training R^2 improves significantly to 0.7589, and testing R^2 increases to 0.7299, indicating better generalization.

DS002 - Optimized Original Dataset

- **Description**: After a deep analysis and adding new important features based on the existing data set. Data was aggregated based on Unique ID for each day to enhance model training.
- **Results**: The models achieve a good RMSE (16.37) and R^2-Score (0.81) and showing strong predictive performance which also extended to testing scenarios. These results indicate a

robust model fit and effective generalization which needs to be analyzed deeper with the experts in further tests.

Using the data set DS002 and testing the different models a high R^2 and good RMSE score could be achieved. The tested models show robust performance in the training and testing phases due to strategic feature additions and data aggregation to that data set and demonstrating the value of thoughtful data preparation and feature engineering. By testing the models with the DS013 dataset and integrating synthetic data into the original one over an extended period, the potential for improving model accuracy and reliability in real-world applications by using a combination of a synthetic and original data set was identified.

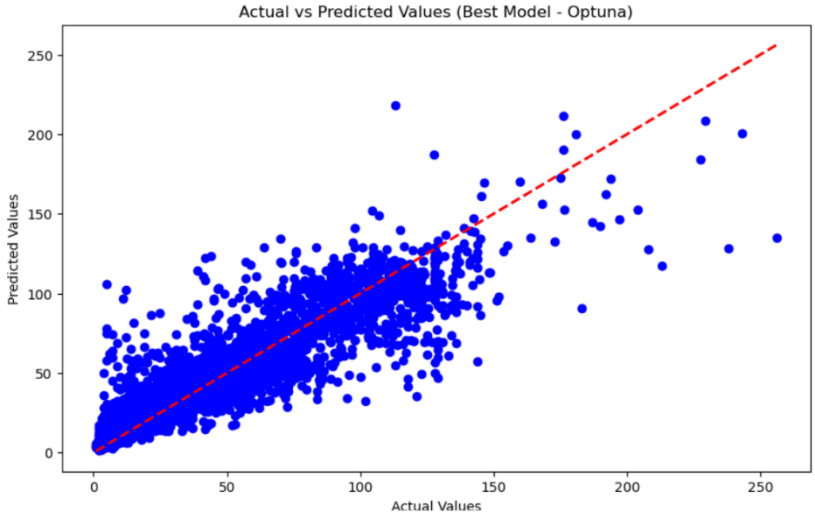

Figure 23: Results of the prediction accuracy for the product quantity by using the XGBoost (Optimized Optuna)

In Figure 23 the prediction accuracy of the selected machine learning model XGBoost model optimized with Optuna, for forecasting the quantity of raspberry fruit in kilograms, is represented. This model achieves a commendable R^2 value of 0.81, indicating a good predictive capability where the model explains 81% of the variance in the quantity of raspberries harvested.

In the plot the x-axis shows the actual measured quantities of raspberries in kilograms and the y-axis represents the quantities predicted by the model. The blue dots scattered across the graph depict individual predictions against the actual quantities. The dashed red line represents the ideal scenario where the predicted values perfectly match the actual values. The proximity of the blue dots to this line reflects the accuracy of the predictions.

From the plot, it's evident that while the model performs well for a broad range of values, it tends to slightly underpredict the higher quantities (as seen from the blue dots trailing below the line at higher actual values).

Results to predict product quality:

The actual results of the product quality prediction will be described below.

Initially, the focus was on predicting the quality percent of harvested fruits, an approach designed to quantify the proportion of the harvest that meets specific quality standards. However, several challenges impacted the efficiency of these models. The subjective nature of quality assessments,

influenced by various external factors like weather information, lack of data or detail level of harvesting information, made accurate modelling more challenging. This complexity resulted in moderate initial R^2 values, reflecting the models' limited capability to capture all nuances affecting quality.

DS003 (Original data)

- **Description**: The original data set is used.
- **Results**: The initial model showed moderate performance. After applying Bayesian Optimization, the testing R^2 increased to 0.6013. Despite improvements, the model faced challenges in capturing universally applicable patterns across datasets, possibly indicating overfitting.

DS014 (Incorporating synthetic data)

- Description: Using the original data set including synthetic data.
- **Results**: Initially, the model performed reasonably well, achieving a testing R^2 of 0.6907. After applying Optuna Optimization, the testing R^2 improved to 0.7803. The reliance on synthetic data alongside original data significantly boosted performance. However, further refinement of the synthetic dataset quality could potentially enhance the model's robustness.

Recognizing the limitations in predicting the precise quality percent for specific supplier, the prediction output parameter was shifted to the specific value of total kg per quality per week. This move aims to generate more directly measurable and operationally relevant outcomes: First tests are showing the performance results from the selected model to predict the specific value for quality class 1 and 2.

DS003 (Original data with adapted preprocessing steps)

- **Description:** The original data set is used, and the preprocessing steps were adapted based on the new selected prediction output parameter.
- **Results for Quality_1 with simple preprocessing:** The model initially demonstrates strong performance. After applying Bayesian Optimization, the testing R^2 improved to 0.93995, reflecting enhanced generalization and reliability.
- **Results for Quality_2 with applying enhanced feature engineering:** Initially, the model achieved testing R^2 of 0.9186. After applying Bayesian Optimization, the testing R^2 improved to 0.93995, underscoring the model's reliability.

These metrics underscore significant improvements in accuracy, reliability, and generalization capability of the models, making them highly valuable for operational planning and strategic decision-making in raspberry production. Based on the provided data set and the created synthetic data set the achieved results need to be proven. Further validation of the developed algorithms needs to be done with further data provided by the pilots in a real scenario.

Moreover, the performance of predicting product quality by using the XGBoost model optimized with Optuna running on the DS003 can be seen below in Figure 24. The figure displays the relationship between actual and predicted values for the "Kg Per Quality_1" dataset using the model XGBoost optimized with Optuna. The blue dots represent individual predictions plotted against the actual values on the x-axis and predicted values on the y-axis. The dashed red line, which indicates perfect prediction accuracy, shows that most predictions closely align with actual values, demonstrating the model's high accuracy. The dense clustering of points around this line across all value ranges indicates strong model performance, particularly in accurately predicting typical quantities, although there are slight deviations at higher values.

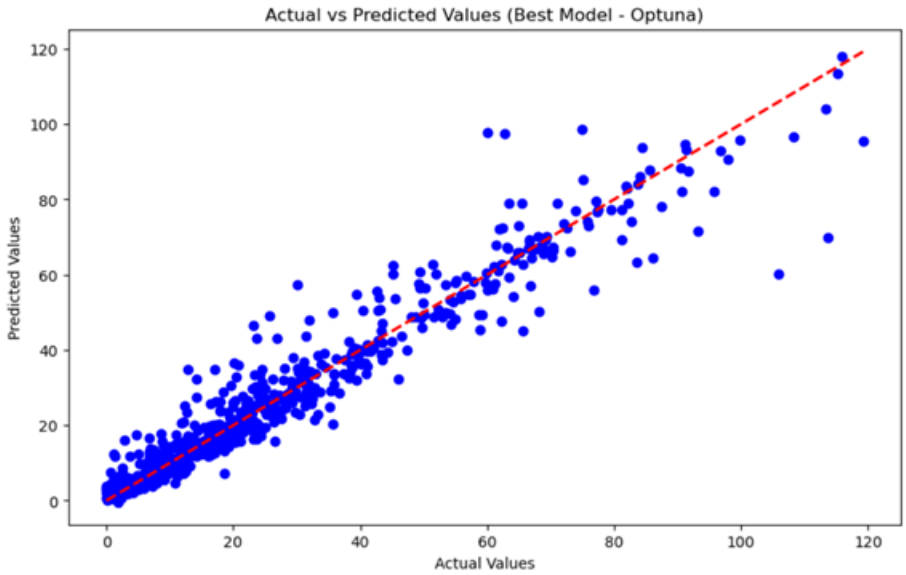

Figure 24: Results of the prediction accuracy for product quality by using the XGBoost (Optimized Optuna)

Moreover, the predictions achieved by the models will be provided to, e.g. WP6.2 the Digital Twin, which can be used as additional input data to be more precise in the forecasting job of the Digital Twin.

4.2.1.2. Marelec

The developed pipeline for enhancing the chicken packaging process and predicting the product quality of the chicken filet is structured into six steps which are displayed in Figure 25. In the modelling phase the pipeline is divided into two approaches: Anomaly Detection for product quality detection and Quality and Quantity Prediction.

The pipeline begins with the business understanding phase where the main objectives of the company are set. After that the data acquisition, understanding and preparation phases are followed by identify and analyzing and increasing the quality of the available data set.

During the data acquisition and understanding phase, it was noted that there were significant issues with the data quality, specifically with the absence of Quality B data and inconsistent data values, which could skew the accuracy of predictive models.

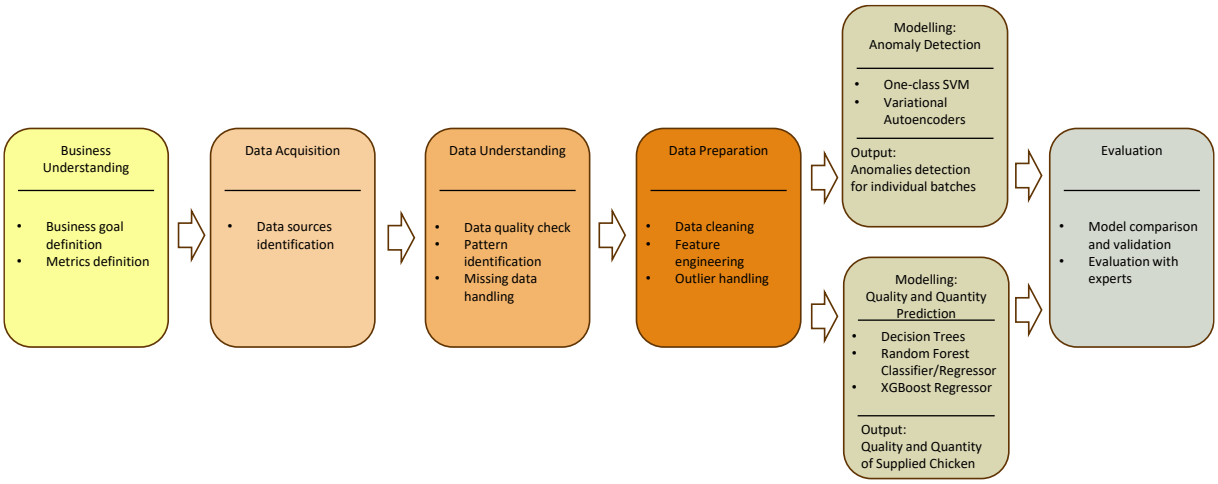

Figure 25: Procedure for the pipeline development for Marelec pilot

The anomaly detection in the modelling phase was specifically introduced to address these data quality challenges by developing an unsupervised learning model that discovers irregularities and outliers in the data. In the absence of Quality A/B data, in the modelling phase advanced anomaly detection techniques were utilized to identify patterns that deviate from the norm, which can often indicate subpar quality of chickens. Identifying these anomalies enables the identification of chicken batches that might require closer inspection, which can help maintain the integrity of quality assurance processes.

On the other hand, quality and quantity prediction models were developed with a thorough data preparation phase, ensuring that the data is clean and ready to be modelled upon. These models use validated and labelled data and operate in two phases. The first phase focuses on quality prediction: it utilizes the cleaned and validated data to develop predictive models that produce the quality of the chicken batch. The second phase, which is quantity prediction, builds upon the data to forecast the production volumes, specifically, the proportion of Quality A or Quality B chicken supplied to the packaging factory. This will enable management to take decisions on supplier continuity, inventory management and other operational aspects.

Data acquisition and data understanding

The dataset originates from the Marelec dataset, supplemented by new features generated from the existing data to enhance the analytical capabilities. The pilot provided datasets from different time periods:

- Dataset_1 contains data from August 2023 to January 2024.
- Dataset_2 contains data from July 2024 to November 2024.

Dataset_1 was provided with an MSSQL database. The primary source for model building has been the extracted data from that database. The main analytical view utilized, known as “BatchView” was provided directly by Marelec and comprises of key operational data. It comprises of 45 distinct features, that can be used for predictive pipelines. Despite its comprehensive nature, a significant challenge with this dataset is the high rate of missing data. Approximately 37% of the total entries

Figure 26: Supplier diversity in Dataset_1

in this dataset lack supplier information, which poses challenges for forecasting supply volumes from suppliers, and can also skew quality predictions.

Data Preparation

In the data preparation stage, a pipeline is developed to clean and prepare the data for modelling. This pipeline has the following phases:

- Missing feature handling: Columns that are completely empty are removed from the database. 'LabelPath' is a feature that had no entries in the database and was removed from the database.
- Irrelevant feature reduction: After consultation, irrelevant columns in the dataset were recognized.
- Feature Engineering: New features such as 'Kill to Production Days', 'Production to Delivery Days', 'Production to Expiry Days', 'Kill to Delivery Days' were calculated from the dates extracted after the Timestamp handling phase. These features are introduced in place of the timestamp features mentioned before, to give a clearer understanding of the time intervals that could impact the quality of the chicken batches. 'ProductionTime' was also used to derive the features of 'Season', categorizing each entry into Winter, Spring, Summer, or Autumn.
- Imputation of feature entries: Based on duplicate 'BatchID', missing suppliers are populated in the database.
- Deletion of entries: Entries which had missing information that could not be inputted were then deleted from the database to maintain data quality.
- Categorical data handling: All categorical data, such as supplier IDs and product types, were converted from numerical codes. This transformation is essential for machine learning algorithms which require numerical input. Mappings for these transformations were preserved in a JSON file to enable reverse transformations for interpretability.
- Final Dataset Preparation: Final checks are performed to verify data integrity and readiness. The dataframe comprising of 31 features is then saved in a parquet file for efficient storage and access.

For the quantity and quality prediction pipeline, an additional step is involved to classify lots based on the Production Grader used. Additionally, the Quality A/B weights are calculated based on different grade types.

During the weight calculation step for Dataset_1, it was observed that Quality B data was not correctly documented on certain days. This occurrence suggested possible issues in data recording. Such anomalies are critical as they could impact the accuracy of quality predictions and overall data integrity. This abnormality was communicated to the pilot company and a new data set was requested.

After the initial data cleaning and preparation, a Principal Component Analysis (PCA) is performed to further refine the dataset for modelling. PCA is a technique used to emphasize variation and bring out strong patterns in the dataset. This is achieved by transforming the original variables int a new set of variables, that are linear combinations of the original variables. This reduces the dimensionality of the dataset without losing much information. It is determined that including

11-12 components can capture approximately 90% of the cumulative explained variance, as shown in the explained variance graph below (Figure 27).

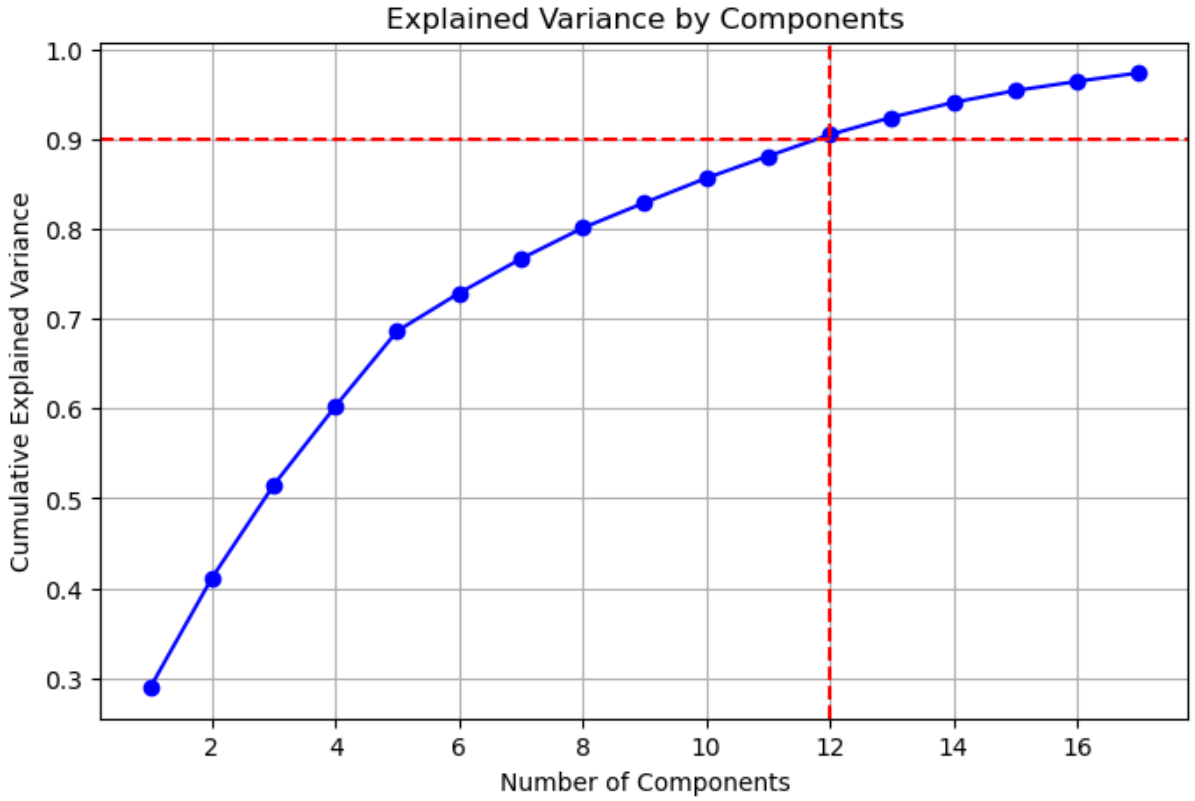

Figure 27: Explained Variance Graph for Principal Component Analysis

Retaining this number of components simplifies the dataset but also maintains enough complexity to capture essential information. From the processed set of 31 features from the anomaly detection pipeline, PCA compresses the dataset into 12 principal components, which explain 90% of the cumulative explained variance. This compressed data is then saved in a Parquet file format for future use.

Modelling

As highlighted in the Data Understanding chapter, Dataset_1, which contains the key information provided by Marelec, has incorrectly recorded weight data. As a result, trained machine learning models, i.e. Quality and Quantity Prediction pipeline, could not be employed. However, outlier detection could be used which can identify potential anomalies in the data entries. These anomalies could pinpoint the lots or batches of chicken which are of subpar quality i.e. Quality B, thereby requiring inspection.

The decision to focus on outlier detection does not reduce the importance of quality and quantity predictions but highlights a pivot to utilize the available data in the most effective way. By targeting anomalous data, a secondary system could be established and specifically designed to identify potential quality issues within the production process. Based on the provision of the new corrected dataset from Marelec, this anomaly detection framework will be validated, and can also be used to enhance the selected predictive models.

Anomaly detection

Two models were developed for the anomaly detection framework:

One Class Support Vector Machine:

Support Vector Machines (SVM) is a traditional machine learning technique which is used for classification or regression tasks. One-Class SVM is a type of SVM which is primarily used for anomaly detection. The model tries to learn what “normal” data looks like and then flags the data points that deviate from this norm. The model uses a Radial Basis Function (RBF) kernel, with a setting (gamma) to ensure that no single feature dominates the decision boundary of “normal” vs “outlier”. So, for instance, just because the chicken batch belonged to one supplier, it will not be favored or unfavored when deciding if it is subpar or not. Note that hyperparameter tuning allows it to select the number of outliers which could be detected. For instance, it can be chosen that 5% of the entries which do not comply with the norm can be identified as anomalies.

A train test split (70:30) is conducted with a random seed before implementing the model. In anomaly detection, the training data is assumed to be "normal". Testing the model then involves seeing how well it identifies data that fall outside the learned boundaries in the test set. After performing this split, the model is fit on the training data and evaluated on the test data to pinpoint the anomalies in the test dataset. Hyperparameters are chosen such that 1% of the dataset are recognized as outliers.

Variational Autoencoders:

Variational Autoencoders (VAEs) are a type of generative model that works by encoding data into a lower dimensional space and then decoding it back to the original space. VAEs are especially useful for anomaly/outlier detection because they are trained to minimize the difference between the input and the reconstructed data, which helps in identifying the outliers. Unlike traditional autoencoders, VAEs encode input data as distributions over latent spaces, thus providing a probabilistic element that enhances the ability to manage data variability. The encoder compresses the data into a latent space, and the decoder reconstructs the input from this compressed representation, where high reconstruction errors indicate anomalies in the data.

As with One Class SVMs, a train-test split is conducted where the training data is assumed to be “normal” by default. After performing the split, the model is fit on the training data and evaluated on the test data to pinpoint the anomalies in the test dataset. Hyperparameters are chosen such that 1% of the dataset are recognized as outliers.

In addition to the anomaly detection framework, the modelling phase incorporates specialized prediction pipelines for quality and quantity of the supplied chicken. As with any machine learning model, a train test split is performed to validate the models’ performance on unseen data. This split not only helps in fine-tuning the model parameters, but also tests their generalizability, which is important for deployment in a real-world environment.

Quality prediction

For predicting the quality of chicken batches, two models are developed:

Decision Tree Classifiers:

Decision Trees make predictions based on a series of rule-based decisions derived from the data features. They are particularly useful for interpretation, since each decision node in the tree represents a clear rule that contributes to the final prediction. For example, multiple features such as “Supplier”, “Weight”, “Season” could contribute to the decision behind whether a data entry is Quality A or Quality B. The target feature is Quality of the chicken.

Random Forest Classifiers:

Building on the Decision Tree model, Random Forest Classifiers use multiple trees to make decisions, thereby increasing prediction accuracy. This is an ensemble approach that reduces the likelihood of overfitting, by averaging multiple decision trees that individually consider random subsets of features and entries. The target feature is Quality of the chicken.

Quantity Prediction

Furthermore, for the quantity prediction the following models are selected and developed for the first tests.

XGBoost Regression:

XGBoost regression, or extreme gradient boosting regression, is a machine learning algorithm based on Decision Trees. It is an ensemble-based algorithm built on the gradient boosting framework, where each new tree is added to minimize the loss in training. The idea is that each successive tree predicts the mistakes of the previous trees, enabling the ensemble to learn from the mistakes of its previous trees. The quantity of supplied chickens (Quality A/B) is the target feature of our pipeline.

Random Forest Regression:

Similar to XGBoost, this is an ensemble-based method, which uses multiple decision trees to make predictions. The Random Forest Regressor averages the predictions of individual decision trees to improve the accuracy of the quantity forecasts. It is particularly useful for capturing non-linear relationships between the features of dataset.

Evaluation

In this chapter the different results for the Marelec pilot will be presented. First the anomaly detection results with two selected approaches will be described. After that the result of the quality and quantity prediction will be presented.

Results anomaly detection

One Class SVM Model

The One-Class SVM model identifies anomalies in the dataset. The model was configured to detect anomalies by classifying approximately 1% of the test dataset as outliers. This approach

helps pinpoint areas where intervention might be necessary, particularly with suppliers or during certain seasons that exhibit higher rates of anomalies. After validation with the experts the threshold of anomaly detection can be also adapted based on the performances in the real state scenario.

The results from the model reveal distinct patterns in the anomaly distribution across suppliers as seen in Figure 28.

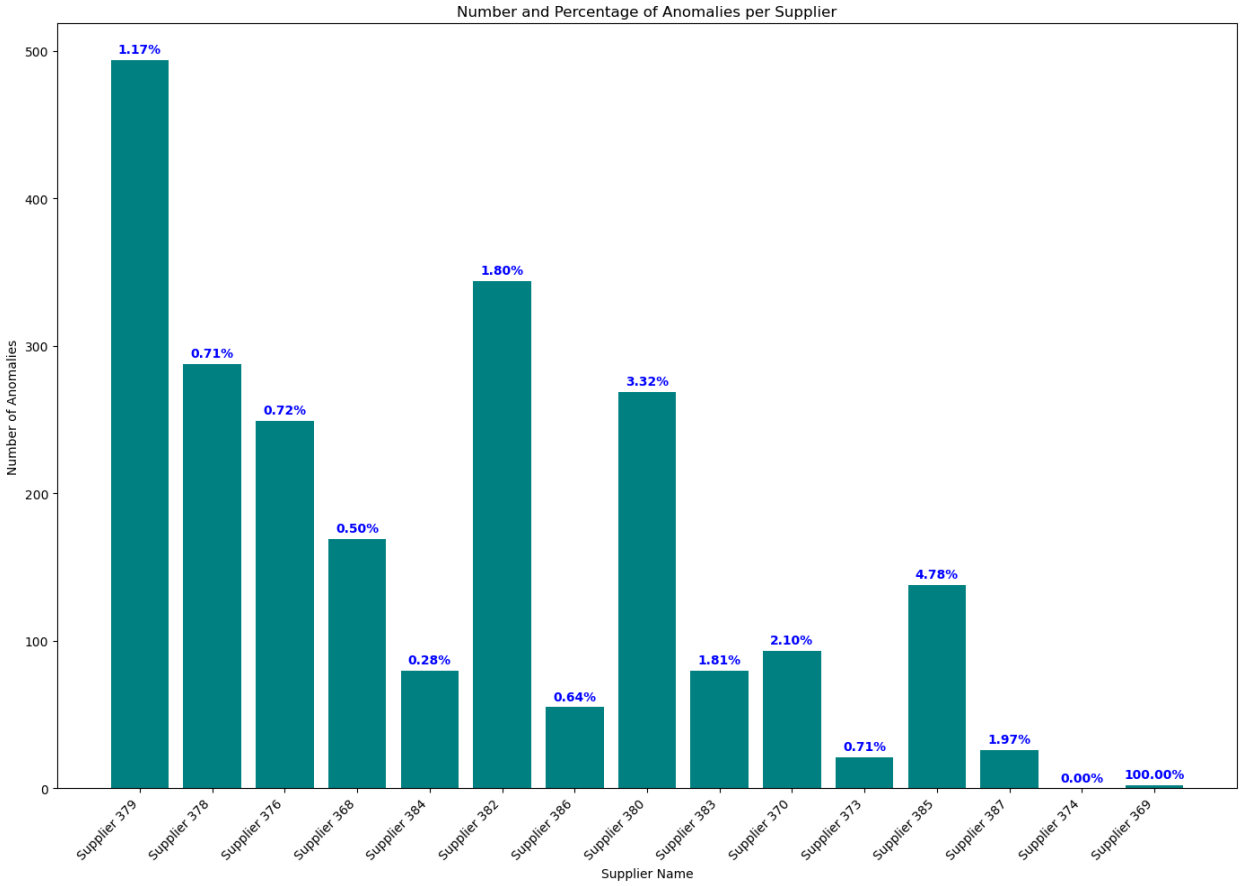

Figure 28: Diversity in anomalies per supplier – One Class SVM model

It is noteworthy that while Suppliers 379 and 382 have a higher count of anomalies, it is Suppliers 369, 380 and 385 that show a higher percentage of anomalies relative to the data entries they contributed. So, while suppliers may have more anomalies since their supply volume is high, other suppliers having a higher percentage of anomalies indicate more significant issues in the chicken quality or consistency of their supply.

Similarly, *Figure 29* displays the anomaly occurrences by season, showing a higher rate in winter (1.15%) compared to autumn (0.80%). This seasonal fluctuation could be influenced by factors such as operational changes or differing supply chain dynamics during these times.

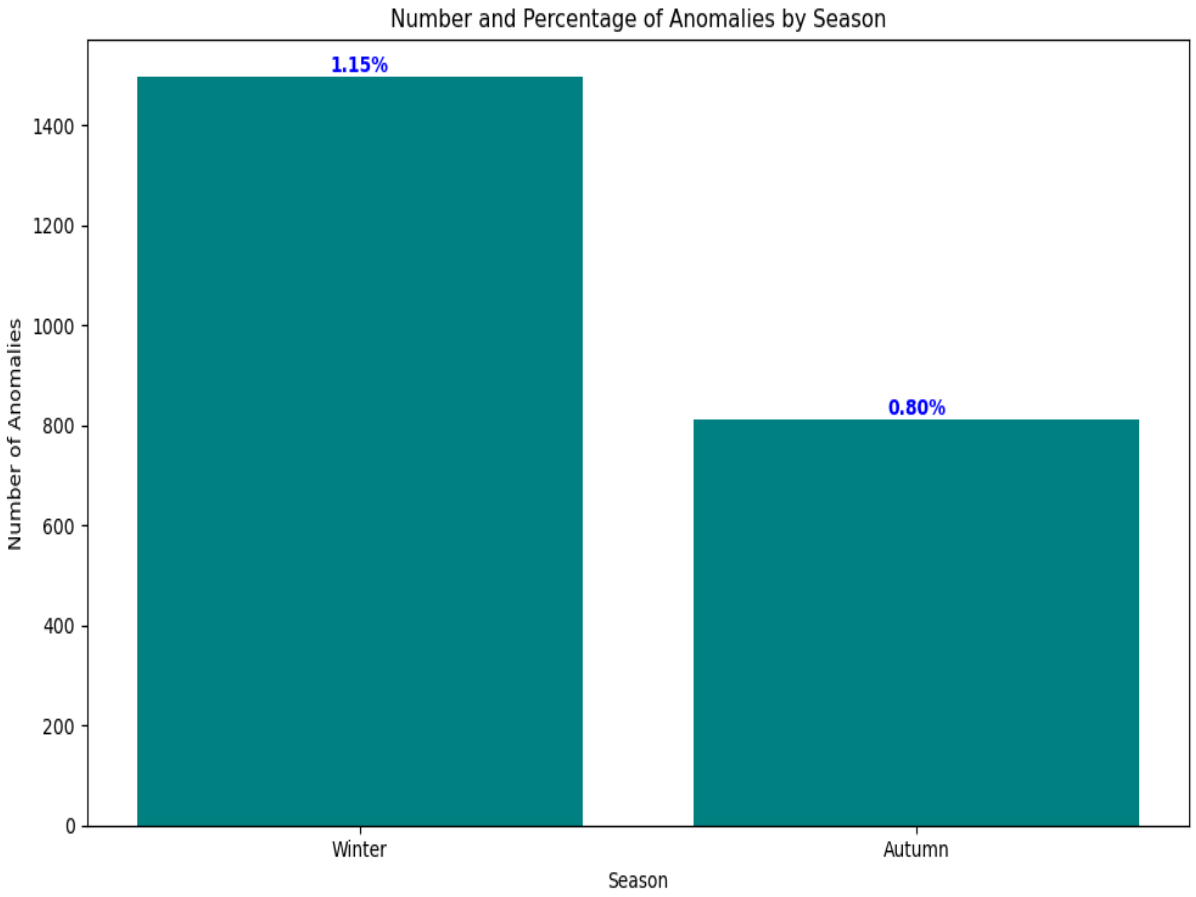

Figure 29: Diversity in Anomalies per Season – One Class SVM model

Variational Autoencoders (VAE)

Variational Autoencoders have a similar intent as One-Class SVMs, where the goal is to identify the outliers in the data. Like the One Class SVM model, the Variational Autoencoder model was configured to detect 1% of the dataset to be the anomalies.

The results from the model reveal interesting patterns in the anomaly distribution across suppliers as seen in *Figure 30*. The visualization indicates that Supplier 382 has a higher percentage of anomalies at 3.75% while supplier 379 has a lower rate of anomalies at 0.96%. This is starkly different from the One-Class SVM model, where anomalies are found for multiple suppliers.

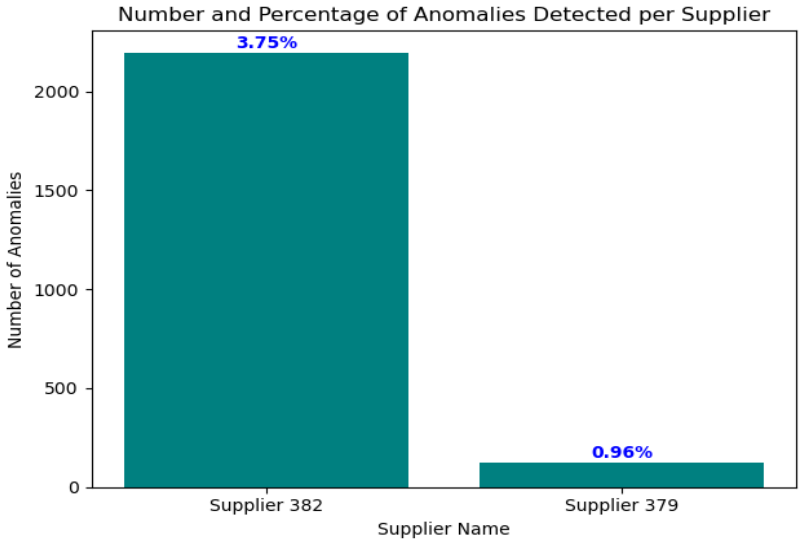

Figure 30: Diversity in Anomalies per Supplier-VAE Model

Figure 31 explores the seasonal distribution of anomalies detected by VAEs with Autumn showing a higher anomaly rate compared to Winter. The seasonal variation could be due to operational changes or supply chain dynamics.

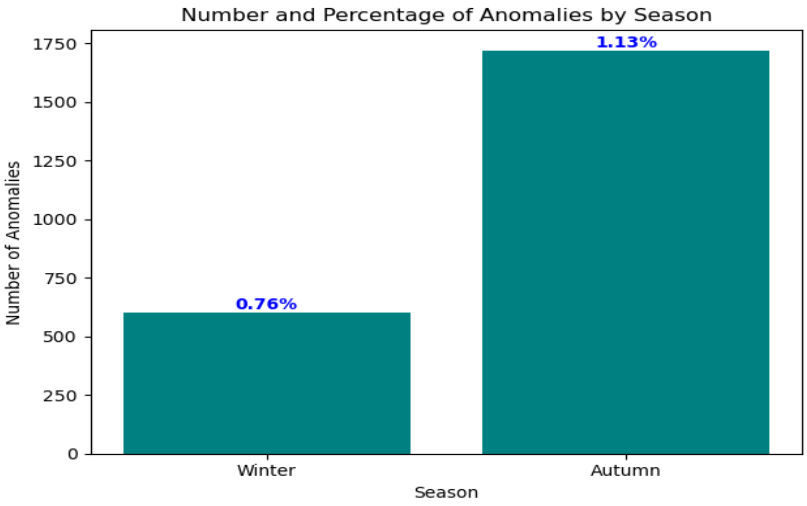

Figure 31: Diversity in Anomalies per Season-VAE Model

These insights into suppliers, seasons, batches or products are crucial for pinpointing specific areas where anomalies are more frequent, so that Marelec can tailor the quality control measures more effectively.

The outputs from both models (One Class SVM and Variational Autoencoders) will undergo validation once the updated dataset with correctly labelled quality tags is available. This will allow for a more accurate assessment of the models' precision in identifying genuine anomalies and adjusting the models accordingly. Additionally, the results will be validated with domain experts from Marelec. These experts will assess whether the identified anomalies correspond to actual issues within the supply chain process, providing a layer of practical validation. The actual result of the models can be seen in Table 17 below:

Dataset ID	No. of used Data Count	Feature Count	No. of used Features	Pre-processing pipeline	Model	Target Output	No. of anomalies detected
Dataset_1	773.339	45	31	Anomaly detection: Without PCA	One-Class SVM	~1% Outliers	2321
Dataset_1	773.339	45	31	Anomaly detection: Without PCA	Variational Autoencoders	~1% Outliers	2308

Table 17: Results of the anomaly detection models

Results product quality and quantity prediction

For the quality prediction pipeline, Decision Trees and Random Forest classifiers will be evaluated once the labelled dataset is available. The same will happen with the quantity prediction pipeline, where XGBoost and Random Forest regressors will be used and evaluated. Due to delays in the data recording and provision of Dataset_2 by the pilot, the data could not be pre-processed and inserted into the pipeline in time for the completion of the report. The preparation and adaptation of the new data set is ongoing and the first results for the quality and quantity forecast are expected at the beginning of 2025. The model results will then be validated directly with Marelec's experts.

Additional to that, the achieved model predictions will also be provided with the other developed solutions like WP 6.2 to support the forecasting process.

4.2.2. Future Steps

Below the next steps and tasks for the different pilots are defined.

Pilot: Sant'Orsola and Marelec

The next steps for the development of the solution for the product quality and quantity prediction will be the testing of additional algorithms and their performance reviewed in tandem with the experts from the pilots. Moreover, the model performance will be validated with the pilot experts and the results will be tested in real state scenario.

Concurrently, the design of the user interface to efficiently present the results to the workers and aid them in their daily tasks, will be optimized. Based on the validation results and the user feedback the solutions will be optimized by integration of upcoming suggestions.

Pilot: MultiScan

Moreover, based on the delay at the provision of a dataset from MultiScan, the work on the rapid reconfiguration solution for the MultiScan pilot can be continued. Therefore, a conceptualization of the MultiScan solution can be developed, to optimize batch handling by calculating the ideal

sequence of batches to reduce machine set up time. Moreover, another step includes the preparation of the supplied dataset. Alongside this, different machine set-up types need to be identified and prepared, paving the way for the subsequent development of a machine learning pipeline. After that, optimization algorithms will be selected, modelled, and analyzed to identify the most efficient handling of batches. The next task includes testing the selected algorithms and evaluating their performances, providing crucial insights into the effectiveness of the developed model.

4.2.3. Functional and Implementation Viewpoint

In these sections, the overview of the functional and implementation viewpoints of the Production Traceability Solution task are represented.

4.2.3.1. Functional Viewpoint

In these sections, the overview of the functional and implementation viewpoints of the Rapid Reconfiguration Solution task are represented.

4.2.3.2. Functional Viewpoint

In the rapid reconfiguration solution different machine learning pipelines are under development to predict the product quality and quantity in an early stage to increase transparency and provide decision in the production planning and order management process. Different data sources like historical production data or open weather API data are used to feed the models. The models want to predict the estimated product quality and quantity for each supplier and their specific field or farm and for each product variety.

Data structure of Rapid Reconfiguration solution

Format	Input / Output	Example
. json/.csv	Input	Acquiring historical production data, open API data, creation of new data based on given information from the pilot experts.
.json/.txt/.csv	Output	Product quality in classes and product quantity in kg

Table 18 – Data structure

Software requirements

Software Component	Description / Role	Required Version / Configuration	Dependencies

Windows OS	Operating system needed to use the tool	Windows 10 Pro	N/A
Python	Programming language	Latest	N/A
Visual Studio Code	Editor	Latest	N/A
JavaScript	Programming language	Latest	N/A
Chrome	Browser	N/A	N/A

Table 19 – Software Requirements

Objects

Front end: The front end will provide the model predictions to the user. The front-end development will start as one of the next steps in the year 2025.

Back end: The machine learning pipeline builds the back end and consists of several steps like data acquisition, data preprocessing and model training. The model results will be displayed over the front end to the user.

Database: The data sets will be provided for the machine learning pipelines to train the selected models.

4.2.3.3. Implementation Viewpoint

The implementation status of the rapid reconfiguration solution is described in the following tables.

Implementation Components	Description
Data Collecting	Data was acquired over several data sources like historical production data including supplier information, open API weather data and additional data based on expert information.
Data Pre-Processing	The provided data was pre-processed, missing values were handled, new features were generated, and important features identified.
Modelling	Selected models were trained on the pre-processed data. Hyperparameter tuning was implemented to increase the model performance.
Data Storage	The model results will be saved in a data storage.

Table 20 – Implementation Components

Below the different implementation components will be described in the given tables.

Implementation component	Data collecting

Description of the implementation component	The acquired data was collected and fed to the pipeline. Based on expert interview further data was added to the pipeline
Used technology	Python,
Technical Description of the Component	*Dependencies*
	Development Language: Python
	Interfaces
	User Interface: Visual studio code

Table 21 – Data collecting

Implementation component	Data pre-processing
Description of the implementation component	The different pre-processing steps were implemented to increase the data quality. Missing data was handled. New synthetic datasets were tested to increase the data variety.
Used technology	Python
Technical Description of the Component	*Dependencies*
	Development Language: Python,
	Interfaces
	User Interface: Visual studio code

Table 22 – Data pre-processing

Implementation component	Modelling
Description of the implementation component	Different models were implemented to predict the product quality and quantity based on the given data sets. Through different performance metric the model results can be compared.
Used technology	Python, REACT
Technical Description of the Component	*Dependencies*
	Development Language: Python
	Interfaces
	User Interface: Visual studio code

Table 23 – Modelling

Implementation component	Data storage
Description of the implementation component	Serves to store the model results.
Used technology	Python
Technical Description of the Component	*Dependencies*
	Development Language: Python Libraries: Json, Txt, CSV

Table 24 – Data storage

4.3. Transferability of the solution

The developed solution so far is predicting the product quality and quantity of the given products in the use cases. The products vary based on the pilots from raspberries to chickens. Both pilots have different data sets. Moreover, preprocessing steps and models used for predicting the product quality and quantity have a high variety. Based on that, transferring these solutions to other different food products in some case major adaptions of the pipelines need to be made. This work strongly depends on the given data volume, data format, and data quality and the selected product.

Furthermore, if the products are closely connected to the pilot products like raspberries, it could be possible that only minor adaptation needs to be done to predict also the product quality and quantity of strawberries or gooseberries. But to clarify that further development in the next project period needs to be done to provide a better estimation of that.

5. AGILEHAND Automated Production Line Operations Optimization

5.1. Overview

The solutions presented in this section enable automation and increased profitability in food sorting tasks across various industries, leveraging advanced sensing technologies to classify products based on their physical properties.

The optimization problem centers on maximizing the allocation of products into different quality tiers within specified batch tolerances, typically allowing for 5-10% deviations depending on customer requirements and pricing structures.

Utilizing data obtained through initial analysis performed by the Smart Sensing Suite, dynamic control strategies are defined for the food classification task. This suite provides comprehensive batch status information prior to classification, allowing for the adjustment of pre-set parameters and classification strategies to suit each specific batch. The classification algorithm is designed to maximize and optimize batch tolerances, thereby increasing supplier profitability while maintaining product quality and adhering to customer specifications.

These solutions are implemented in accordance with the IEC-62264 standard, ensuring proper hierarchy and database management. The proposed system architecture employs Node-RED microservices and PostgreSQL databases for seamless communication with various industrial sensors and actuators.

This flexible setup offers high automation capabilities and can be adapted to different food processing sectors, from fruit and vegetable sorting to seafood classification. By leveraging multispectral imaging and machine learning algorithms, this versatile system can be calibrated to detect and analyze key quality indicators specific to each product type. Whether assessing the ripeness of berries, the freshness of fish, or the quality of citrus fruits, the system provides a robust framework for optimizing food sorting processes across diverse applications in the food industry.

5.2. Design Status

The operations for which the PRAgile solution design has been proposed include the sorting of different foods for subsequent grading into quality grades provided by industrial partners. Therefore, the main operation on which we focus is the classification of the same before the configuration of the batch.

To perform the operation, we must consider all the variables that define the degree of quality of a given food. Therefore, we need to store all the variables (Color, Size, Defects...) that are considered necessary and all the industrial protocols used in the industrial sensors that collect this information (OPC-UA, TCP/IP, MQTT...). The proposed technology for this monitorization is based on a Docker running the services on an Ubuntu 22.04 VM. This deployment allows seamless integration with the classification equipment from the industrial partners.

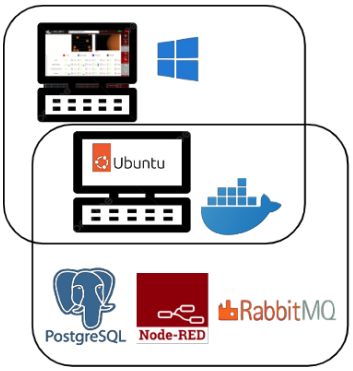

Figure 32: Ubuntu VM running inside Windows VM, hosting Docker containers for PostgreSQL, Node-RED, and RabbitMQ

The variable monitoring starts once the docker is up and running all the services. The goal is to constantly monitor the values affecting the classification parameters and the result classification. This data monitoring will enable the training, testing and validation of models that help to optimize and maximize the classification tolerances proposed.

All these variables allow us to obtain a histogram that defines the total result of the classification of an inspected lot according to its quality grade and the selected classification parameters as can be seen in the following Figure.

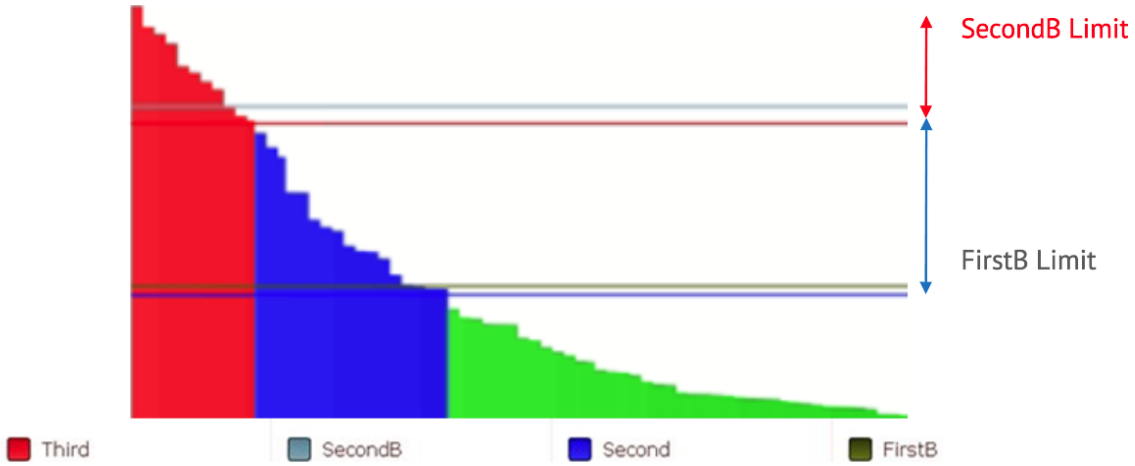

Figure 33: Histogram of quality parameters for the inspected lot with definition of quality categories

By monitoring and defining new variable limits that determine the quality grade of a given batch, the allowed failure tolerance can be maximized, as shown in the following figure, where the limits have been artificially modified to optimize the maximum tolerance allowed by the customer.

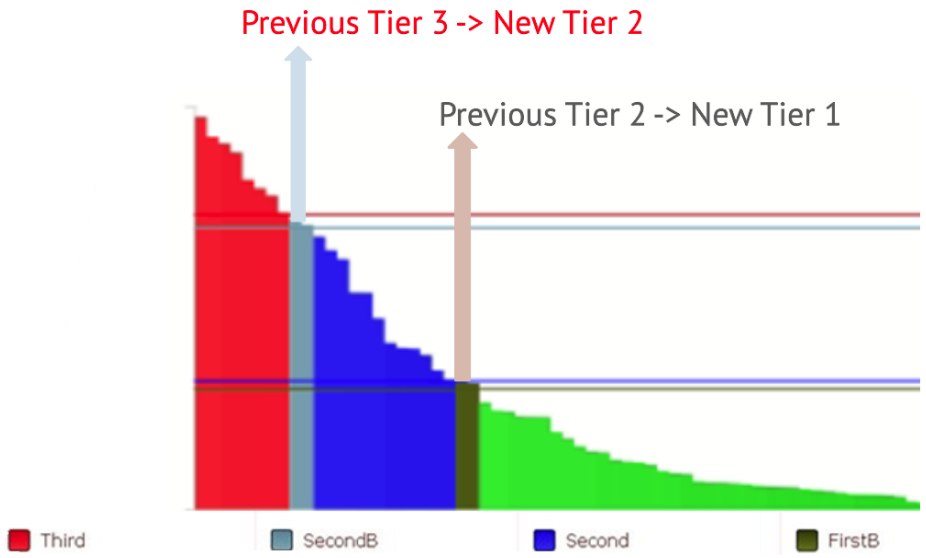

Figure 34: Histogram of quality parameters for the Virtual Tier categories

5.2.1. Current Implementation

5.2.1.1. Multiscan

The deployment of the solution in the case of fruit sorting in MultiScan will allow us to adjust the maximum allowed batch tolerances. For this purpose, we monitor **95 variables** according to the OPC-UA protocol. The classification variables consist of 6 parameters to be monitored (Color, WhiteRotten, Scars, Decay, Greenish, and Defect and Diameter). The monitored physical properties are based on Multispectral images that determine the previously mentioned properties on a range from 0 to 100. Multispectral imaging techniques combine the use of infrared (IR), near-infrared (NIR), and other spectral bands to accurately identify potential defects or anomalies that degrade the product quality. These quantified variables are monitored and stored according to MultiScan quality guidelines.

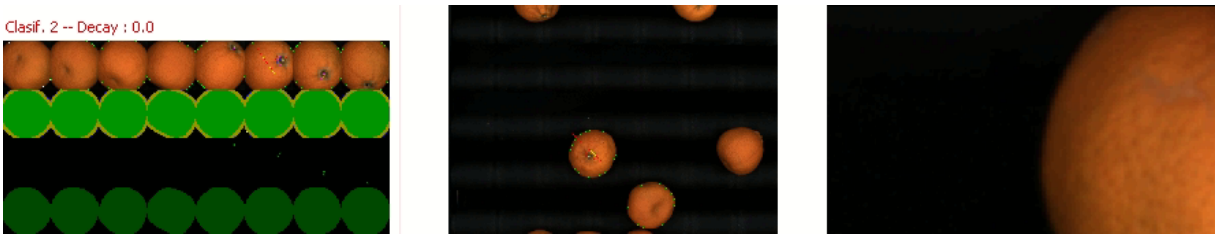

Figure 35: Multispectral Imaging samples on Multiscan Classifier

Each of these classification variables has two values: a Working Value that provides the current value of the limit (in the range of 0 to 100) and an Overwrite Value, which indicates whether the limit is active (Value=0) and influences the classification or not.

The variable information that is being read can be seen in the right part of the screen in the

Node-RED UI, that provides the values of each variable in real-time, as well as the changes in its values over the batch classification order.

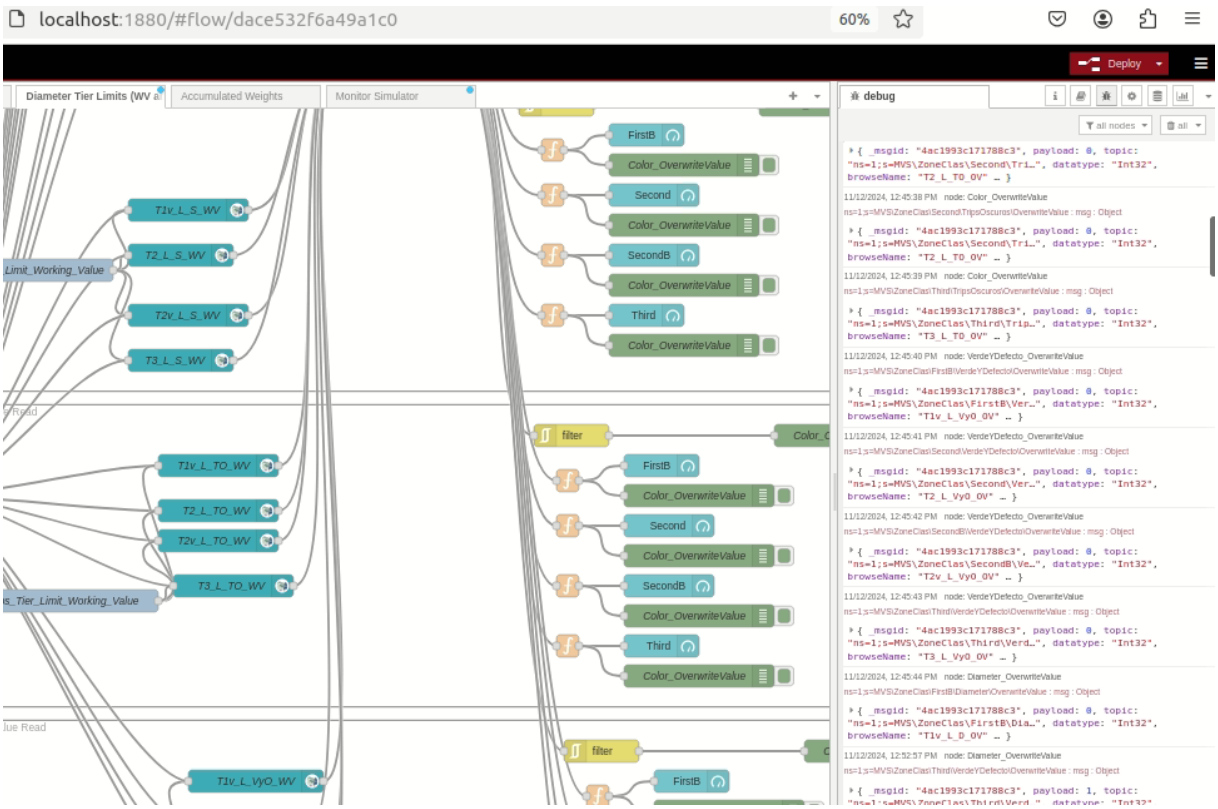

Figure 36:OPC-UA data Collection pipeline based on Node-RED

Therefore **60 variables** (for reading/writing purposes) correspond to the MultiScan simulator data shown in the figure below:

Figure 37: Multiscan Equipment Tier Limit variables

In Node-RED microservice, it can be verified that these values are synchronized in the Dashboard configured within the same, shown in the following Figure for user purposes.

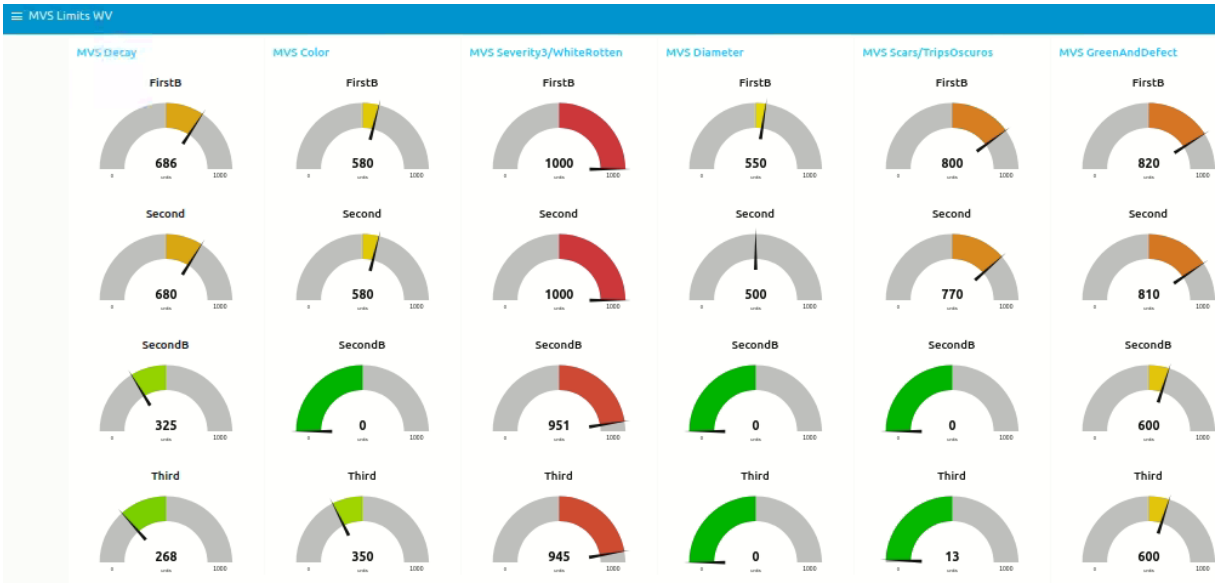

Figure 38: Node-RED Dashboard UI for Tier Limit variables

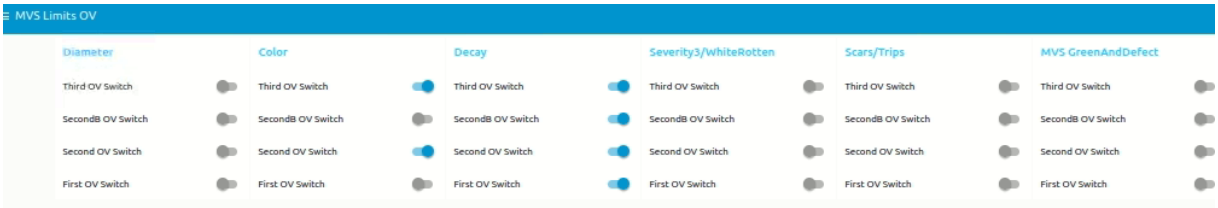

Figure 39: Node-RED Dashboard limit active/inactive

In addition to these variables, the accumulated weight of each classificator is monitored (for reading purposes). These variables provide the accumulated kilograms classified per each of the data classes that impact the imitation strategy to be followed by our algorithm.

This data is provided in % for the overall accumulated weight of the batch in the Simulator. Providing

Figure 40: Accumulated Tier Weights Distribution per Batch

In Node-RED microservice, the information regarding the accumulated weight is provided in kg per Tier (5 Total Accumulated Weights) and per Tier and Category (30 in total, 5 per each of the 6 Categories). The **Total Accumulated Weight for the 2** virtual categories, **SecondB and FirstB** are the objective functions intended to be **optimised based on the previous 60 variables.**

To enable a correct monitorization of all the variables OPC-UA node library that allows us to connect to each of the sensors for monitoring or writing will be used. Several operations can be performed by implementing these nodes, such as variable writing that changes the expulsion and classification values.

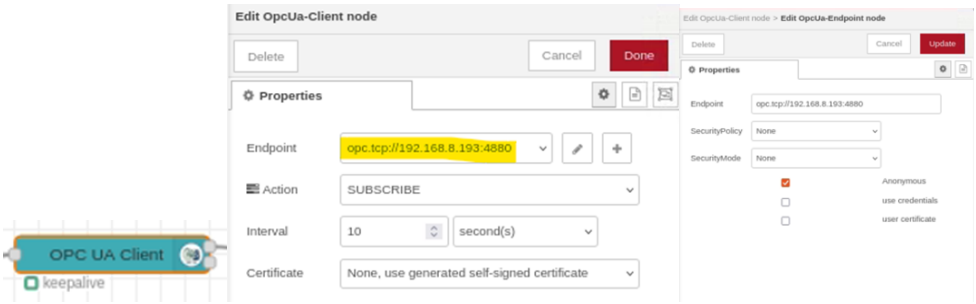

Figure 41: OPC-UA Node Confguration

In addition to the OPC-UA Client Nodes, OPC-UA item Nodes are used to properly monitor each of the 95 variables. The Item Node parameters are provided in a query from the PostgreSQL database according to the IEC-62264 standard defined.

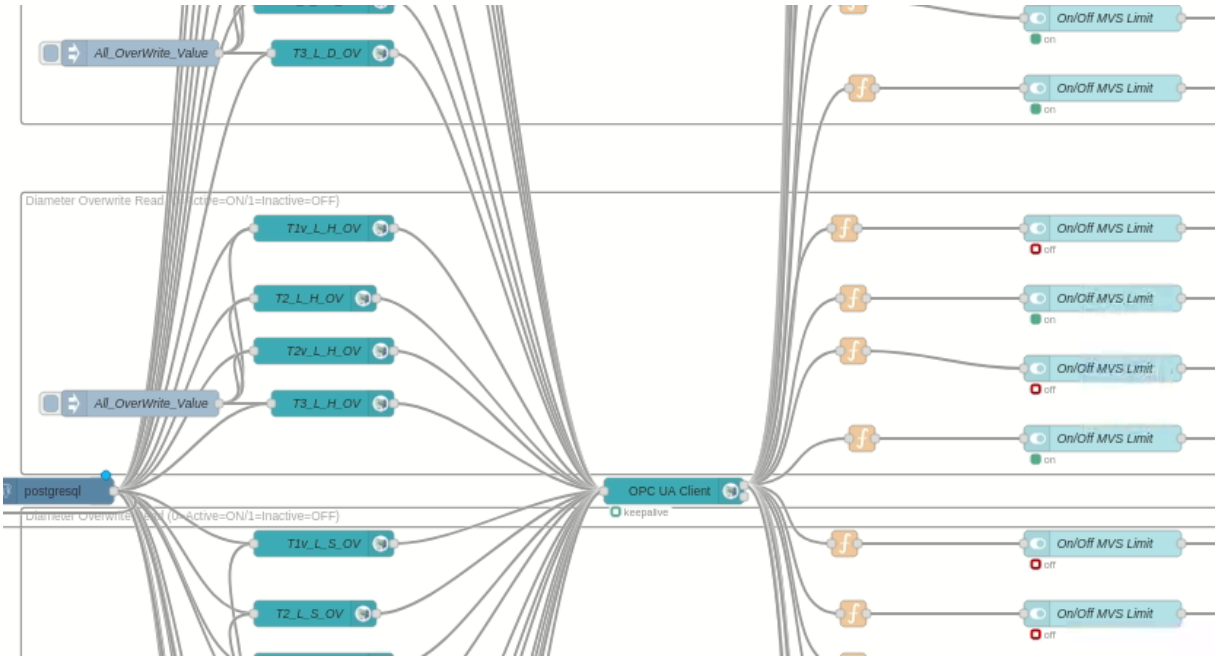

Figure 42: OPC-UA Monitoring Pipeline

5.2.2. Future Steps

In the upcoming months, rapid reconfiguration methods for item classification will be assessed. Two approaches have been previously selected for this purpose. The Parametric Optimization Model focuses on the minimization of an *f(ClassLimit)* function that adjusts the class limit tolerances based on techniques such as Grid Search, Bayesian Optimization or CMA-ES algorithms for fine-tuning.

The second approach would be based on a predictive Model with Inverse Optimization, employing random Forest or Extreme Gradient Boosting model (XGBoost) to infer accumulated weight percentages per Tier and optimize the limits inversely.

The primary objective is to fine-tune, assess and train these models to check whether their performance in real-time applications is feasible, as well as compare their suitability on each of the proposed scenarios.

5.2.3. Functional and Implementation Viewpoint

These sections present an overview of the functional and implementation viewpoints of the Automated Production Line Operations Optimization task.

5.2.3.1. Functional Viewpoint

Our inspection solution is based on Node-RED microservice running on a local Docker container. The microservice is configured to work with OPC-UA protocol for the equipment hardware sensors and MQTT protocol for the Smart Sensing Suite. The OPC-UA node id information is gathered from the PostgreSQL database according to the IEC-62264 standard.

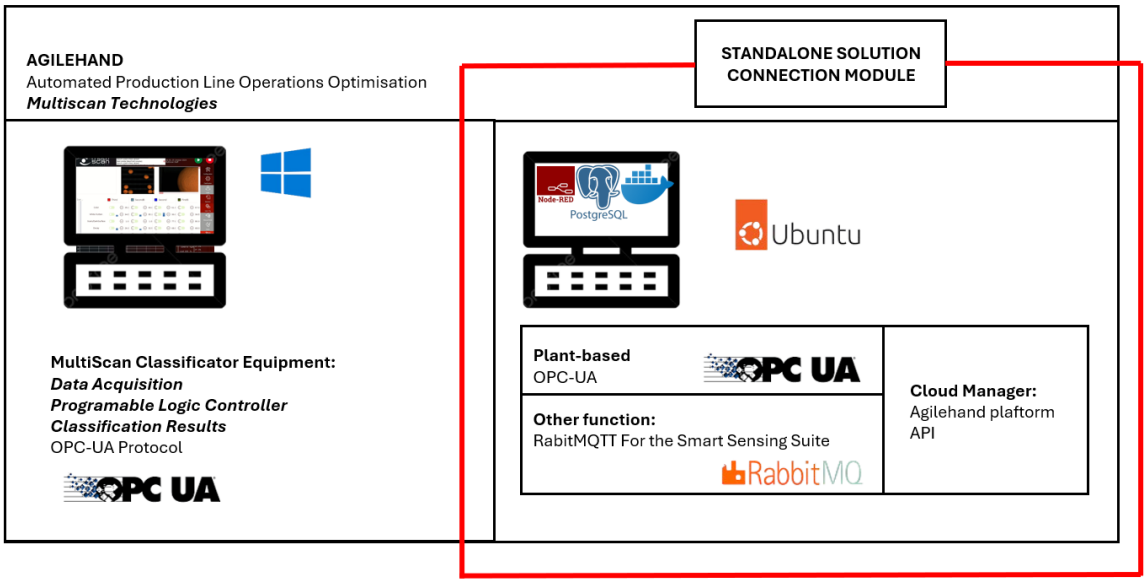

Figure 43: Standalone Solution Node-RED Module with Multiscan Use Case

The functional viewpoint of the Automated Production Line Operations Optimization suite is presented in Figure 43. This solution collects data from the company classification system, including classification parameters based on physical properties, thanks to Multispectral imaging. The batch is classified according to the equipment results, and the Tier groups are established accordingly. Our standalone solution monitors the variability for each of the Tier classes. It optimizes in real-time the results for Tier 1 and Tier 2 maximization, improving the overall benefits over the inspected batch.

Data structure of Automated Production Line Operations Optimization

Format	Input / Output	Example
.json (OPC-UA)	Input	Acquiring data from the equipment classifier (e.g., accumulated weight, active/inactive limits, limit values)
.json (OPC-UA)	Output	Modified Limit Values for Tier 1 and 2 maximizations

Table 25: Data Structure

Software requirements

Software Component	Description / Role	Required Version / Configuration	Dependencies
Linux OS	Operating system needed to use the tool	Ubuntu 22.04	N/A
Visual Studio Code	Editor	Latest	N/A
JavaScript	Programming language	Latest	N/A
Node.js	Programming language	Latest	N/A
Mozilla Firefox	Browser	N/A	N/A
Mosquitto MQTT	Back-end system that coordinates messages between different clients	Latest	N/A
postgreSQL	Local database that stores the node id to monitor in the suite	Latest	N/A
Docker	Collaborative containerization software for developers	Latest	N/A

Table 26: Software Requirements

5.2.3.2. Implementation Viewpoint

The implementation architecture provides an overview of the implementation components needed, which are detailed in the following tables.

Implementation Components	Description
Data Collecting	Acquires input data from the company's classification equipment (Multiscan Hardware and PLC systems) that gathers all relevant information related to product classification parameters.
Data Processing	Serves as a data processing step to filter and calculate only relevant information for the solution. For example, accumulated weight variation and Tier percentage based on batch processing status.

Data Post-Processing	Serves to obtain the optimal solution from the previous step and set/write on the classification limit values to its optimal values. This task is performed dynamically with the batch evolution.
Data Storage	Serves to store the classification results for each batch, according to the IEC-62264 standard.

Table 27: Implementation Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required for implementation of each implementation component.

Implementation Components	Data Pre-Processing
Description of the implementation component	Acquiring data from the company's classification equipment (Multiscan Hardware and PLC systems) that gathers all relevant information related to product classification parameters.
Used technology	NodeJS, Javascript, Node-RED
Technical Description of the Component	*Dependencies*
	Language Development: NodeJS, JavaScript Libraries: node-red-contrib-postgresql, node-red-dashboard and node-red-contrib-opcua
	Interfaces
	User Interface: Node-RED UI and Dashboard Network/Protocols: OPC-UA/MQTT

Table 28: Data Collecting

Implementation component	Data Pre-Processing
Description of the implementation component	Serves as a data processing step to filter and calculate only relevant information for the solution. For example, accumulated weight variation and Tier percentage based on batch processing status.
Used technology	NodeJS, Javascript, Node-RED
Technical Description of the Component	*Dependencies*
	Language Development: NodeJS, JavaScript Libraries: node-red-contrib-postgresql, node-red-dashboard and node-red-contrib-opcua
	Interfaces
	User Interface: Node-RED UI and Dashboard Network/Protocols: OPC-UA/MQTT

Table 29: Data Processing

Implementation component	Data Post-Processing
Description of the implementation component	Serves to obtain the optimal solution from the previous step and set/write on the classification limit values to its optimal values. This task is performed dynamically with the batch evolution.
Used technology	NodeJS, Javascript, Node-RED
Technical Description of the Component	*Dependencies*
	Language Development: NodeJS, JavaScript Libraries: node-red-contrib-postgresql, node-red-dashboard and node-red-contrib-opcua
	Interfaces
	User Interface: Node-RED UI and Dashboard Network/Protocols: OPC-UA/MQTT

Table 30: Data Post-Processing

Implementation component	Data Storage
Description of the implementation component	Serves to store the classification results for each batch, according to the IEC-62264 standard.
Used technology	JavaScript
Technical Description of the Component	*Dependencies*
	Language Development: JavaScript Libraries: Node-RED-contrib-postgresql

Table 31: Data Storage

5.3. Transferability of the solution

The application of multispectral imaging for fruit classification can be transferred to other sectors such as raspberries (Santorsola) and frozen fish (Produmar) processing. Here's how this knowledge could be transferred:

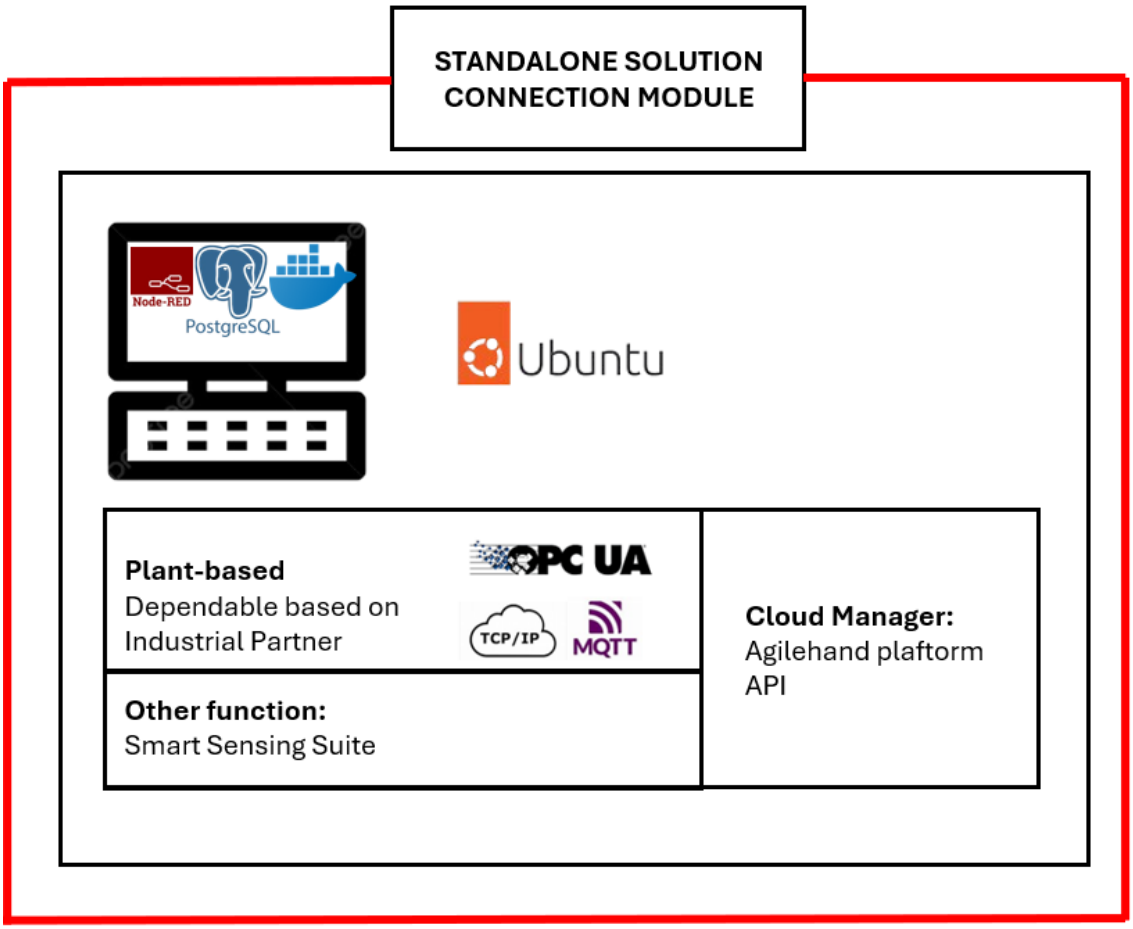

Figure 44:Standalone Solution Node-RED Module

Raspberry Classification

Multispectral imaging can be used to classify raspberries based on:

- Ripeness: By analyzing spectral signatures in the visible and near-infrared regions, the system can determine the ripeness stage of raspberries.
- Quality assessment: The technology can detect bruises, mold, or other defects that may not be visible to the naked eye.
- Size and shape: Multispectral imaging can measure the dimensions and uniformity of raspberries for grading purposes.
- Sugar content: The system can estimate total soluble solids, indicating sweetness.

Frozen Fish Processing

For frozen fish processing, multispectral imaging or segmentation models can be applied to:

- Species identification: Different fish species have unique spectral signatures that can be used for automated sorting.
- Freshness evaluation: The system can detect changes in the fish's chemical composition that indicate freshness levels.
- Quality grading: By analyzing color, texture, and other surface characteristics, the system can grade fish quality.
- Defect detection: Multispectral imaging can identify freezer burn, dehydration, or other quality issues in frozen fish.
- Size and weight estimation: The technology can be used to sort fish by size and estimate weight without physical contact.

In both cases, the system would need to be calibrated with specific wavelengths relevant to the properties being measured for raspberries and frozen fish when using Multispectral imaging. Machine learning algorithms, such as Support Vector Machines (SVM) or Neural Networks, can be trained on multispectral data to classify the products into different quality tiers, similar to the citrus fruit classification system. In case RGB cameras or other sensors are used, standard segmentation models can be used for physical properties assessment such as Color, Diameter, and/or Perimeter properties, allowing its process optimization. This approach would be compatible with the solution deployment in a different scenario where the hardware is not based on multispectral imaging.

6. Conclusion

This document provides a detailed account of the progress and outcomes of WP6 of AGILEHAND, emphasizing its contribution to innovative solutions for production traceability, data-driven digital twins, and intelligent production system configuration.

In summary the achievements can be described in four points:

Pilot Project Advancements: Across the Multiscan, Sant'Orsola, Produmar, and Marelec pilots, the project has made significant strides in developing integrated traceability systems. These advancements enhance productivity and efficiency, particularly in managing soft and deformable products.

Technological Breakthroughs: Adoption of an IEC-62264-compliant database model has ensured robust data orchestration and system interoperability. Key tools such as PostgreSQL and Node-RED have been effectively leveraged to handle data management and real-time processing, respectively.

Future Focus: Building on current progress, future efforts will target improvements in traceability solutions, the refinement of digital twin technologies, and the optimization of automated production processes. These steps are essential for ensuring the scalability and adaptability of the solutions in diverse operational contexts.

Industry Impact: Designed for transferability, the Y project solutions have the potential to address use cases beyond the initial pilots, promoting broader adoption. By aligning with food safety and quality standards, the project underscores its relevance to the industry while supporting regulatory compliance.

The WP6 of AGILEHAND project is successfully paving the way for more agile and efficient production environments. Through its innovative technological contributions, it addresses critical challenges in food handling and processing, ensuring a substantial positive impact on productivity, compliance, and scalability across the industry.

7. References

[1] "Regulation - 178/2002 - EN - EUR-Lex." Accessed: Dec. 23, 2024. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2002/178/oj

[2] K. L. Hulebak and W. Schlosser, "Hazard Analysis and Critical Control Point (HACCP) History and Conceptual Overview," *Risk Analysis*, vol. 22, no. 3, pp. 547–552, Jun. 2002, doi: 10.1111/0272-4332.00038.

[3] "ISO 22000:2018 - Food safety management systems — Requirements for any organization in the food chain." Accessed: Dec. 23, 2024. [Online]. Available: https://www.iso.org/standard/65464.html

[4] "IEC 62264-1:2013 - Enterprise-control system integration — Part 1: Models and terminology." Accessed: Dec. 23, 2024. [Online]. Available: https://www.iso.org/standard/57308.html