This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement No 101120823

Advancing the physical intelligence and performance of roBOTs towards human-like bi-manual objects MANipulation

D2.4. MANiBOT system technical specifications and architecture

WP number and title	WP2 – MANiBOT requirements and technical specifications
Lead Beneficiary	CERTH
Contributor(s)	AUTH, TUDa, SSSA, UBU, UoB, TUW, THL, CIOP, ABB, FG, MASOUTIS
Deliverable type	Report
Planned delivery date	30/05/25
Last Update	30/05/25
Dissemination level	SEN

Disclaimer

Participant No	Participant organisation name	Short Name	Country
1	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS	CERTH	GR
2	ARISTOTELIO PANEPISTIMIO THESSALONIKIS	AUTH	GR
3	TECHNISCHE UNIVERSITAT DARMSTADT	TUDa	DE
4	SCUOLA SUPERIORE DI STUDI UNIVERSITARI E DI PERFEZIONAMENTO S ANNA	SSSA	IT
5	UNIVERSIDAD DE BURGOS	UBU	ES
6	TECHNISCHE UNIVERSITAET WIEN	TUW	AT
7	TWI ELLAS ASTIKI MI KERDOSKOPIKI ETAIREIA	THL	GR
8	CENTRALNY INSTYTUT OCHRONY PRACY - PANSTWOWY INSTYTUT BADAWCZY	CIOP	PL
9	ASEA BROWN BOVERI SA	ABB	ES
10	FRAPORT ETAIRIA DIACHEIRISIS TON PERIFEREIAKON AERODROMION TIS ELLADAS ANONYMI ETAIREIA	FG	GR
11	SCHWARZ DIGITAL GMBH & CO. KG	SDI	DE
12	DIAMANTIS MASOUTIS AE SUPER MARKET	MASOUTIS	GR
13	UNIVERSITY OF BRISTOL	UoB	UK

Document Authors

Partner	Author
CERTH	Dimitrios Giakoumis, Andreas Kargakos, Ioannis Mariolis, Stefanos Papadam, Georgia Peleka, Evangelos Skartados, Christina Theodoridou, Petros Toupas, Dimitra Triantafyllou, Georgios Tsamis, Georgios Zampokas, Dimitra Zotou
AUTH, TUDa, SSSA, UBU, UoB, TUW, THL, CIOP, ABB, FG, MASOUTIS	CERTH acquired feedback via document templates from the partners regarding the information on the functional components they are developing and the system's physical architecture and SW/HW requirements. The consolidation of this feedback into the current deliverable has been performed by CERTH's authors.

Document History

Version	Date	Status	Description
0.1	03/09/24	Draft	Table of Contents
0.3	04/10/24	Draft	Initial version
0.7	25/10/24	Draft	Updated with T2.2 feedback
1.0	31/10/24	Final	After internal review
1.1	30/05/25	Final	Addressed the reviewer's comments by: 1) adding section 6.3 on handling requirements based on the pilot site descriptions and the physical properties of the items considered for robotic manipulations, 2) updating section 8.3.1.1 with new information on the robot design and its connection to the analysis of the requirements of section 6.3. Small additions were also made to section 1.3 and the introduction of section 6. The newly added or updated text is in blue font to facilitate the reader.

Definitions, Acronyms and Abbreviations

Acronyms and Abbreviations	Description
WP	Work Package
D	Deliverable
DSS	Decision Support System
EC	European Commission
EO	Earth Observation
EU	European Union
GPS	Global Positioning System
IMU	Inertial Measurement Unit
SFR	System Functional Requirement
SLAM	Simultaneous Localisation and Mapping
UR	User Requirements

Executive Summary

The present deliverable aims to provide a comprehensive overview of the overall architecture design for the MANiBOT system. The architecture has been developed based on user requirements and specific use cases outlined in the corresponding deliverable D2.2 - "MANiBOT User requirements and use cases".

To construct and elaborate on the anticipated MANiBOT architecture, a hierarchical methodology has been adopted. The level of detail increases progressively through the hierarchy. Initially, the MANiBOT system is dissected into conceptual modules, highlighting the key components of the system, including both software modules and their interdependencies. Following this, a use case-specific functional analysis is presented, determining the necessary processing actions required for the system to achieve the objectives of each sub-use case while defining the corresponding technical specifications.

After the conceptual analysis, a functional view of the system is detailed. This section breaks down the system's architecture into functional modules and components, providing comprehensive specifications regarding their functionalities and interactions. This structured approach facilitates the independent development of each module.

Given the complexity of the MANiBOT system, technical specifications must be assessed concerning performance, hardware, and software requirements based on identified user needs. The mapping of these specifications involves integrating functional requirements with the necessary hardware and software components while considering the performance benchmarks set by prioritized use cases and user requirements. The requirements mapping process results in a list of prioritized functional requirements that will guide the implementation of the MANiBOT system.

Building upon the high-level architecture of the MANiBOT system, its functionality is evaluated through dynamic modelling using SysML activity and sequence diagrams tailored for each sub-use case. Then, the Development View is presented, outlining how the architecture supports the development process, assigning responsibilities to partner organizations, and specifying software and technology constraints. Additionally, the Development View contains agreements on software configuration management. The Deployment View is thoroughly described afterwards, outlining the modules and their corresponding software/hardware requirements. This includes existing software tools and frameworks utilized by partners, such as ROS for simulation. The hardware requirements detail various sensors, robotic platforms, and manipulators selected to meet the demands of each sub-use case.

Our analysis concludes with the presentation of our implementation plan that defines the timeline for addressing the functional requirements throughout the project duration.

Finally, a series of Annexes is provided, detailing hardware and software requirements along with the mapping of system functional requirements to technical specifications.

1 Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Scope of the deliverable

This deliverable (D2.4 "MANiBOT system technical specifications and architecture") presents the final Conceptual Architecture and System Specifications for the MANiBOT solution. It includes a detailed Development and Deployment overview, along with a dynamic analysis of the MANiBOT system. The document provides a comprehensive perspective on the MANiBOT system, covering the software modules, the hardware requirements for each module, and the robots and additional hardware selected for each sub-use case.

The objective of this deliverable (D2.4 "MANiBOT solution technical specification and architecture") is to provide a comprehensive overview of the MANiBOT architecture, summarizing its technical and functional design. It outlines the fundamental capabilities of the MANiBOT system by thoroughly detailing each module.

The primary goals of this deliverable are:

- Presenting the design methodology for the MANiBOT Conceptual Architecture
- Defining the high-level functionalities of the MANiBOT system
- Conducting a technical specification analysis for each sub-use case
- Offering an overview of the development and deployment framework utilized
- Providing a dynamic analysis of the system for each sub-use case independently

1.2 Relation to other Activities and Deliverables

This deliverable heavily depends on the analysis of the user requirements and the MANiBOT use cases conducted by the efforts of Task T2.2 “MANiBOT framework requirements analysis and use cases definition” and presented in deliverable D2.2 “User requirements and use cases”. In that deliverable the use case scenarios have been defined with specific sub-use cases linked to prioritised user requirements. In the current deliverable we utilise this analysis to extract the system functional requirements, map them to the sub-use cases and extract the system’s technical specifications. We are also utilising the sub-use cases analysis with their defined steps to perform the dynamic analysis of the system, generating corresponding dynamic activity diagrams for each sub-use case.

On the other hand the analysis performed in this deliverable will be utilised by all technical tasks (WP3-WP5) developing the functional components of the MANiBOT system, as well as, the integration tasks (WP6) and the validation and deployment tasks (WP7). However, since we are applying an iterative approach to system design and development the presented architecture and specifications will further be revised and updated utilising the development and deployment feedback. The updated analysis will be tracked via internal reports and presented in future MANiBOT deliverables, such as D6.4 “Integrated bimanual mobile manipulator robot”.

1.3 Structure of the deliverable

The deliverable is structured as reported below:

Chapter 2 outlines the methodology used to determine the architecture of the MANiBOT system. It presents the fundamental principles guiding the definition of the architectural elements and highlights the key concepts followed in establishing the conceptual architecture. Additionally, it delivers detailed specifications for the software modules that make up the MANiBOT system and describes the methodology used to define the system's technical specifications.

Chapter 3 describes the conceptual architecture of the MANiBOT system, detailing the main components (building blocks) of the anticipated system as software modules and their interdependencies.

Chapter 4 introduces the functional analysis of the MANiBOT system, outlining the system's functional requirements as by mapping them to the identified sub-use cases of the system.

Chapter 5 offers a functional perspective on the introduced conceptual architecture, detailing the functional elements, their responsibilities, and their primary interactions with other components.

Chapter 6 outlines the technical specifications of the MANiBOT system, detailing the requirements for each individual hardware and software module, along with their corresponding performance requirements. Additionally, it presents the mapping of the sub-use cases, tracing them from their associated user requirements to the technical specifications. After the review, section 6.3 “Handling requirements in MANiBOT use cases” was added in order to describe the handling requirements of MANiBOT use cases based on the pilot site descriptions and the physical properties of the items considered for robotic manipulation.

Chapter 7 presents the dynamic analysis of the MANiBOT system, evaluating the actions performed by the system, the interactions among its modules, and the interactions between the system and the supervisor. These elements are analyzed within the context of each sub-use case and illustrated through activity diagrams.

Chapter 8 defines the Development View, illustrating how the architecture facilitates the development process. It identifies a responsible partner and any relevant software or technology constraints for each module. Additionally, the Deployment View is described, detailing the modules and their software/hardware requirements. It includes existing software tools or frameworks contributed by partners (e.g., ROS). Finally, the hardware requirements of the system are presented, encompassing a range of sensors, robotic platforms, and robotic manipulators selected to meet the demands of each sub-use case. After the review, in section 8.3.1.1 “MANiBOT Robotic Platforms”, the description of the robotic platform design was updated and connected to the requirements occurring from the analysis of the pilot sites’ environment and the physical properties of the items to be handled, made in section 6.3. Furthermore, information was added on the reachability and trajectory study conducted to ensure the suitability of the robotic arms for the MANiBOT use cases.

Chapter 9 provides a conclusion, summarizing the key findings and outcomes of the document.

2 Architecture & Technical Specifications Derivation Methodology

2.1 System Architecture Concepts and Design Principles

The design of the MANiBOT system's architecture adhered to the guidelines outlined in the "Software Systems Architecture" handbook [1]. The system's architecture is composed of multiple interconnected modules, encompassing software elements, their externally visible properties, and relationships. The design process focused on describing the system's components and their interactions to achieve the desired functionality. Following the basic design principles, the following guidelines should be followed:

- Modularity and Openness: The system architecture is designed to be modular and open, allowing both the MANiBOT system and its users to utilize the functional modules described in the architecture.
- Technology Independence: The architecture aims to be as technology-agnostic as possible, promoting the use of generic, standardized solutions that can be implemented using various key technologies, including open-source and commercial options.

To determine the software architecture of the MANiBOT system, a hierarchical approach was taken, as shown in Figure 1. The process begins with the MANiBOT concept and provides a preliminary analysis of its fundamental architectural elements in an abstract conceptual manner. The system's conceptual architecture outlines the hardware and software components that together form the foundational elements of the MANiBOT system.

Following this, the Functional View describes the system's functional structure by defining its components and illustrating how the required functions are carried out. It specifies and provides details on the associations, relationships, and connections between the system's components. Regarding hardware, the necessary physical interconnections between the hardware components and subsystems that make up the entire system are also defined.

Finally, based on the defined use cases (outlined in deliverable D2.2), a dynamic analysis of the MANiBOT system is presented to provide insights into the system's operation. This analysis demonstrates how the system functions in practice across various use cases and sub-use cases. It also offers an overview of each task related to the collaboration between the system and users, as well as details on timing and signal exchanges. Additionally, it illustrates the interactions and information flow among the system's architectural elements to present a complete workflow. Thus, the dynamic analysis serves as an evaluation mechanism for the functional components by applying them to specific sub-use cases within a scheduled plan.

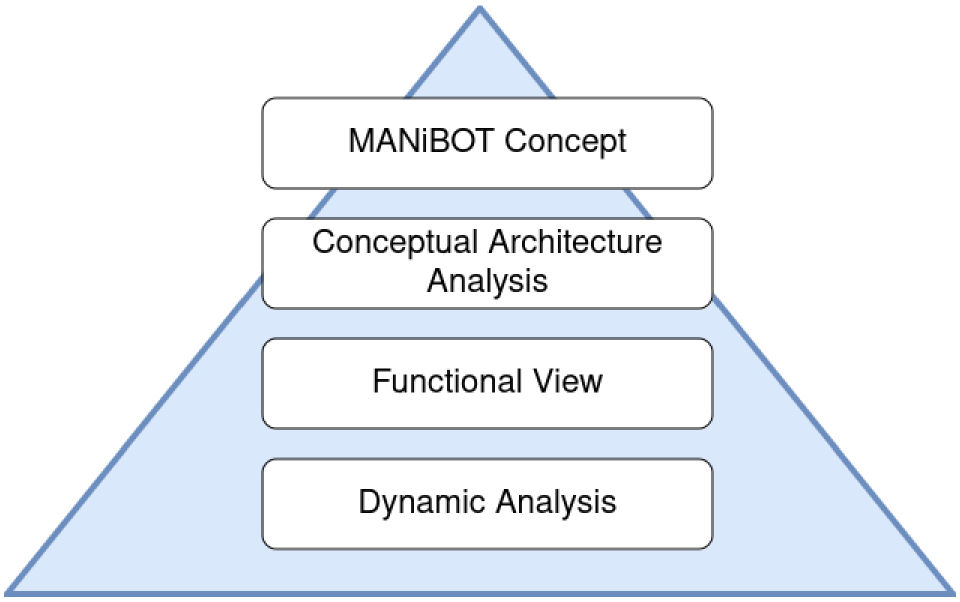


Figure 1 Hierarchical analysis of the MANiBOT architecture.

2.2 Architectural Development Process

This section outlines the methodology for designing the architectural components of the MANiBOT system, detailing their functionalities and interactions, from the conceptual architecture to the high-level functional view and finally to the system's dynamic structure. The process is described in the following steps:

- Step 1: The foundation for accurately determining the architecture is the definition of user requirements, along with the corresponding use cases and sub-use cases, all established and described in Deliverable D2.2.
- Step 2: Based on the user requirements and the specified use cases/sub-use cases, the MANiBOT system's technical specifications are defined. This involves a conceptual breakdown of the system into essential structures, considering hardware and software requirements while taking performance needs into account. The system's functional requirements act as an interface between user requirements and technical specifications.
- Step 3: A sequence of system functionalities, executed by the relevant software modules, fulfils specific tasks within a sub-use case. The functional view provides a detailed analysis of these software modules, including a comprehensive description of each functional component, their interdependencies, operational requirements, and indicative performance criteria.
- Step 4: Once the high-level architecture is established, the system's functionality is evaluated through a dynamic model using the use cases defined in Task T2.2. This involves assessing the system's functionality across different application scenarios.
- Step 5: Following the functional view, the development of the MANiBOT system's modules is planned, including an analysis of the computational resources and hardware required for deployment.

An overview of this methodology is illustrated diagrammatically in Figure 2.

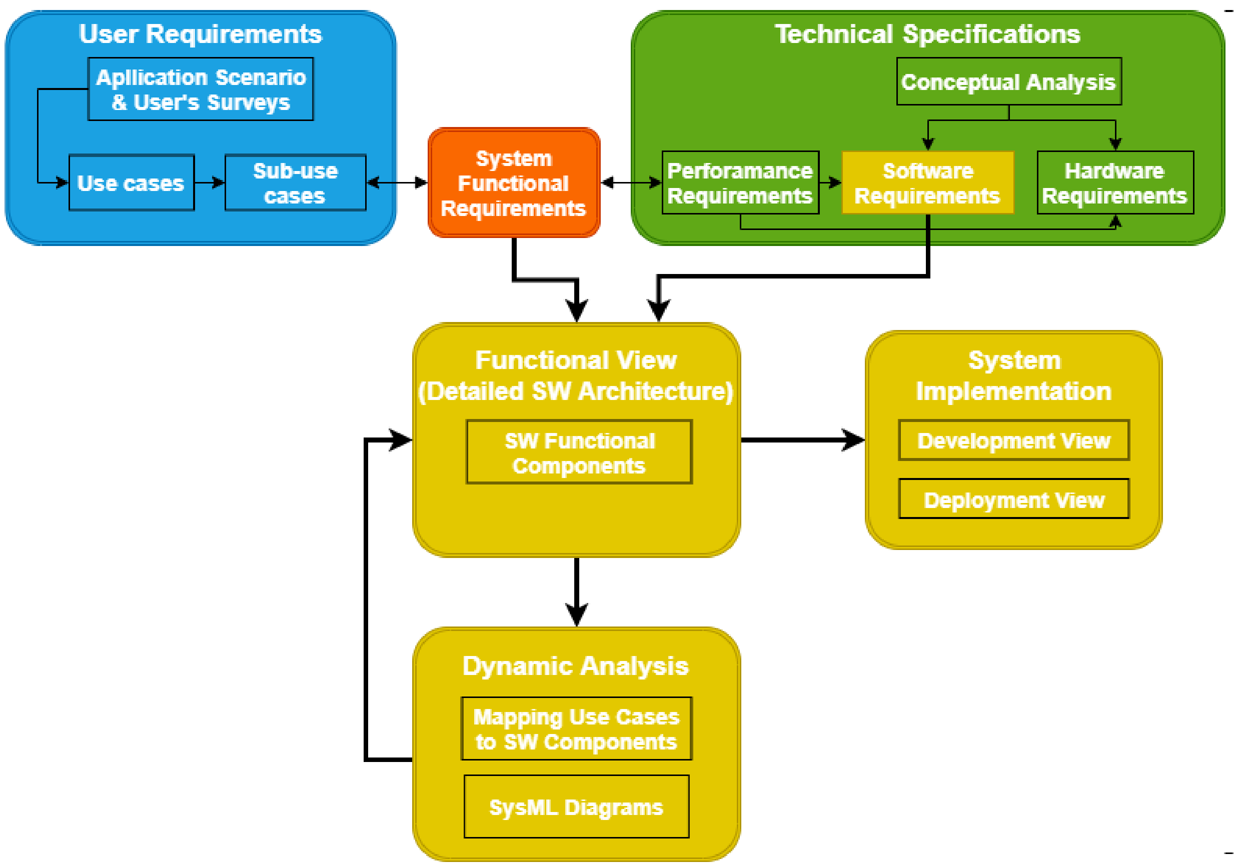


Figure 2 Diagram with the employed architectural design process.

2.3 Methodology Overview for System Definition and Technical Specifications

To understand the role of D2.4 and its connection to D2.2, it is important to explain the initial stages of the MANiBOT system's development process, which include the following steps:

1. The User Needs identified through the analysis of the original project vision, user surveys, and user requirements led to the definition of the Use Cases and Sub-Use Cases, as documented in deliverables D2.2 and D2.4.
2. Based on the Sub-Use Cases, the Functional Design of the MANiBOT system was carried out, resulting in a set of Functional Requirements.
3. The Functional Requirements, along with the available technology base (e.g., technologies accessible to the Consortium, prior results) and the Design Constraints (e.g., environmental standards, regulatory/safety conditions), form the foundation for the Architectural Design of the hardware and software.
4. From the Architectural Design, the technical specifications for the MANiBOT system are established, covering performance, hardware, and software requirements.

2.3.1 Mapping of User Requirements to Technical Specifications

Following the methodology outlined in this section the performance requirements for the MANiBOT system were established through a process that maps user requirements to technical specifications. In this mapping process, each use case and sub-use case is initially analyzed in relation to the corresponding user requirements, which ultimately leads to the definition of the system's performance requirements. The subsequent step involved defining the technical specifications for the individual MANiBOT hardware components, taking into account both the system's performance needs and the anticipated requirements of the use cases. This approach provides a foundation for the technical developments needed to build the MANiBOT system from a hardware perspective. The overall workflow for determining the technical specifications is illustrated in Figure 3.

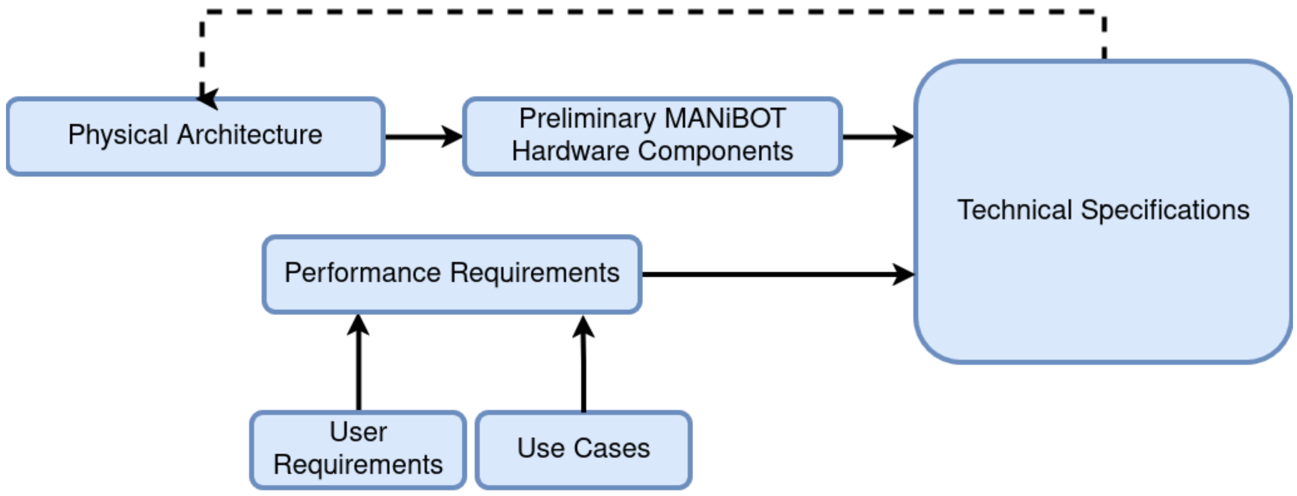


Figure 3 Workflow for the determination of Technical Specifications of the MANiBOT system.

2.3.2 Definition of the MANiBOT Physical Architecture

To establish the physical architecture of the MANiBOT system, the indicative hardware components (such as the robotic platform, robotic arms, RGB/RGBD sensor, etc.) are defined based on the performance requirements and the planned physical architecture. This involves determining how these hardware components can be integrated within the system's conceptual design, as well as identifying the interfacing options that can be used to enable communication among them.

2.3.3 Establishment of the Technical Specifications of the H/W components of MANiBOT

To create the final version of the MANiBOT system's conceptual physical architecture established, collaboration among the project partners was necessary to define the specifications for each hardware component. This involved identifying existing commercial components that can be used, as well as determination of the detailed specifications for components that need to be developed within the project.

3 Conceptual Architecture Analysis

3.1 The MANiBOT Conceptual Architecture

The MANiBOT project aims to develop a versatile, bi-manual mobile robotic system capable of executing a diverse range of manipulation tasks with human-like efficiency. This innovative system will be specifically designed for environments where the handling of various objects—potentially unknown in terms of their characteristics—must be performed with precision and adaptability. Central to the MANiBOT concept is a dual-arm robotic platform equipped with advanced multimodal perception systems that integrate vision, proximity, and tactile sensing for enhanced perception and manipulation.

This architecture enables the robot to understand its environment and recognize objects through sophisticated algorithms that fuse sensory information, allowing for rapid and effective manipulation of deformable and rigid objects alike. The platform will feature a suite of manipulation primitives, including both prehensile and non-prehensile techniques, enabling the transfer and handling of objects of varying sizes, weights, and materials. The system's robust human-robot interaction (HRI) capabilities will also foster trust and efficiency in collaborative operations. By employing a diverse range of sensors and technologies the MANiBOT solution is equipped to perform complex manipulation tasks, pushing the boundaries of physical intelligence in mobile robotics. Figure 4 depicts the MANiBOT's conceptual architecture.

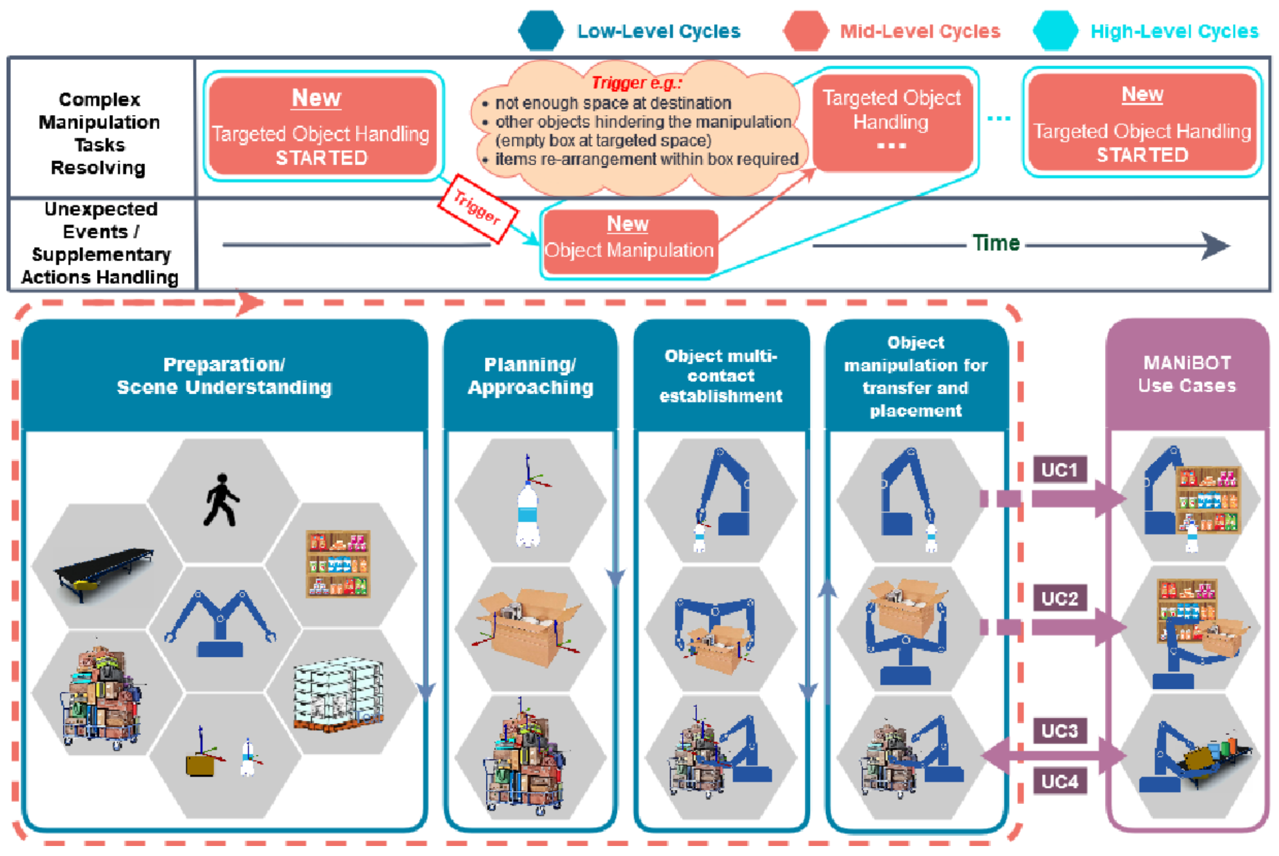

Low-Level Cycles
Mid-Level Cycles
High-Level Cycles
Complex Manipulation Tasks Resolving
Unexpected Events / Supplementary Actions Handling
New
Targeted Object Handling STARTED
Trigger e.g.:
not enough space at destination
other objects hindering the manipulation (empty box at targeted space)
items re-arrangement within box required
Targeted Object Handling
...
New
Targeted Object Handling STARTED
Trigger
New
Object Manipulation
Time
Preparation/ Scene Understanding
Planning/ Approaching
Object multi-contact establishment
Object manipulation for transfer and placement
MANiBOT Use Cases
UC1
UC2
UC3
UC4

Figure 4 Overview of MANiBOT's conceptual architecture

The core concept of MANiBOT revolves around adaptive, multi-level robot cycles designed to achieve versatile manipulation capabilities, depicted in Figure 4. At the foundational level are low-level cycles, each representing a distinct, fundamental step of the four-step interaction process—preparation, planning, contact establishment, and manipulation. Building upon these, mid-level cycles encompass sequences of low-level cycles, enabling the effective manipulation of a single object. High-level cycles then bring together multiple mid-level cycles to accomplish more complex tasks, such as handling several objects simultaneously.

These cycles are highly adaptive, responding dynamically to context. Adaptations occur across all levels, allowing MANiBOT to adjust both its perception—such as shifting attention toward specific elements in the environment or modifying modality weightings—and its action plans. For instance, within low-level cycles, the robot can adapt behaviors in response to new environmental data, while mid- and high-level cycles permit more substantial adjustments to facilitate complex multi-object manipulation. Through these layered, context-aware cycles, MANiBOT is equipped with a robust framework for flexible, human-like interactions across diverse scenarios. The MANiBOT conceptual architecture is structured around a series of multi-level cycles comprising four key steps:

Step (1): Preparation/Scene Understanding This initial stage involves safe, human-aware navigation of the robotic platform to the designated manipulation area as defined through the MANiBOT HRI interface. Once the robot is roughly positioned within the operational space, it begins an initial assessment of the target object and its surrounding environment. This assessment includes vision-based target recognition, pose estimation, and an understanding of the scene, encompassing the identification of both the starting and intended destination positions of the object. Furthermore, the robot assesses available pick and place affordances, which can inform complex multi-object handling, and predictive structural inference (e.g., object piles).

Step (2): Planning/Approach Building on Step (1), the robot determines a specific manipulation approach (e.g., handles-based, prehensile, or non-prehensile) and identifies optimal contact points. Throughout a vision-driven feedback loop, the robot repositions itself as needed, guiding the end effector into a pre-contact position while progressively incorporating proximity sensing for accurate object approach. The aim here is to enable swift yet safe movement, fostering both operational safety and human trust.

Step (3): Multi-Contact Establishment During this phase, the robot establishes the necessary contact points to execute the manipulation plan devised in Step (2). If deviations occur, the robot's cognitive functions adjust actions in response through a responsive perception-action feedback loop. Multi-modal perception here combines visual servoing with tactile and proximity sensing, shifting towards a greater reliance on tactile input for precise manipulation.

Step (4): Object Manipulation for Transfer and Placement This final step initiates the object transfer based on the plan from Step (1), which may be dynamically adapted as new data from tactile, visual, force/torque sensing, and other modalities indicate deviations. Here, the robot applies bi-manual actions, including pushing, toppling, and pulling, to complete the transfer process.

The robot's cognitive abilities allow it to fluidly transition among different cycles and adapt to increased task complexity, such as ensuring adequate placement affordance or adjusting object positions mid-transfer when necessary. For instance, the robot may need to re-arrange items on a supermarket shelf to create space for a new item or adjust a fallen item during a box transfer. The architecture's adaptive cycle structure empowers the robot to handle such intermediate tasks autonomously, ensuring robust and reliable task completion across diverse manipulation scenarios.

3.2 The Basic Conceptual Modules

3.2.1 Adaptive robot perception for object recognition and dynamic environment sensing module

The Adaptive Robot Perception for Object Recognition and Dynamic Environment Sensing module addresses the complex challenges of robotic perception in cluttered, constrained environments filled with diverse objects. For effective object manipulation, MANiBOT's perception system needs to recognize a wide range of objects which differ in size, material, and shape, while also including unseen items, and extract their manipulation affordance areas, even when they are stacked or packed closely together. Similar to human perception, understanding the physical properties of objects requires integrating visual cues (such as shape, size, and material) with tactile feedback (rigidity, weight) for comprehensive object characterization.

This adaptive perception system enables MANiBOT to dynamically adjust sensing modalities based on task requirements and environmental constraints. For instance, the robot may rely primarily on visual inputs for initial recognition and localization, identifying object category, 6-DOF pose, and relevant manipulation affordances, without requiring precise object models. When in proximity, MANiBOT enhances its perception with tactile feedback to infer additional physical properties, like weight and rigidity, optimizing handling based on each object's unique characteristics.

In cluttered settings, structural analysis supports MANiBOT's understanding of an object's body, pose, and manipulation affordances, while assessing the stability of surrounding items to ensure safe manipulation. Furthermore, MANiBOT employs multimodal sensing for dynamic environment awareness, integrating vision and proximity data for effective human detection, safe interactions, and obstacle avoidance. The system continuously adapts its focus on the most effective sensing input, prioritizing vision for long-range environment assessment and proximity feedback when approaching or interacting closely with objects.

A core feature of MANiBOT's perception system is its ability to learn from limited data, enabling rapid adaptation to new objects by associating them with existing knowledge. A continual learning framework will be developed, supporting this capability, managed through an advanced MLOps platform that respects key principles of Trustworthy AI, including security, privacy, and transparency.

3.2.2 Manipulation primitives, bimanual control and navigation module

The Manipulation Primitives, Bimanual Control, and Navigation module enables MANiBOT to operate effectively in dynamic, human-populated environments, addressing key challenges in safety, human comfort, and adaptive handling. Human-aware navigation relies on real-time perception data to dynamically adjust robot trajectories, considering both human comfort zones and obstacles in the environment. Safety is further reinforced by accounting for infrastructure restrictions (e.g., restricted airport zones) during path planning.

Upon reaching the task area, MANiBOT encounters additional complexities due to cluttered and confined spaces, with objects of varying sizes, deformability, and fragility that may be stacked, confined in containers, or surrounded by other items. This diversity requires a robust set of manipulation strategies beyond traditional pick-and-place. To accommodate these demands, MANiBOT's manipulation module integrates compliant, multimodal controllers with bimanual handling capabilities, allowing the robot to execute a range of manipulation primitives, including non-prehensile actions like pushing, even in the presence of uncertainties. The bimanual control framework enables MANiBOT to learn and robustly execute a sequence of manipulation actions tailored to the complex tasks of shelf restocking or baggage handling. Leveraging learning-by-demonstration, the robot replicates human-like trajectories that respect imposed constraints and employ passive impedance-force controllers to achieve adaptive grasping and contact control. Visual and tactile feedback is integrated to enhance grasp stability and adjust forces as needed, supporting reliable execution of manipulation goals.

For small items in cluttered spaces, push-grasping strategies will be included, whereas a stable hold of large objects will be established by bimanual grasping. For sizable items, bimanual control facilitates non-prehensile manipulation, such as pushing an open box filled with items along a narrow shelf or positioning luggage on a cart. This approach also enables MANiBOT to reorient items, for example, by rotating luggage on a conveyor to reveal a tag, using opposing points of contact. These capabilities ensure that MANiBOT can adapt to a wide variety of manipulation scenarios while maintaining efficient, safe, and human-comfortable operation.

3.2.3 Robot cognition and HRI module

The Robot Cognition and Human-Robot Interaction (HRI) module of MANiBOT is designed to create an adaptive framework for understanding and interacting with the various environments and humans inside them. Central to this module is the continuous modeling of the robot's surroundings, enabling it to interpret crucial information about objects and their relationships. By developing semantic and adaptable representations, the robot enhances its navigation and manipulation capabilities, enabling more intuitive interactions with humans.

To enhance coordination in unstructured and semi-structured settings, the module will implement a hierarchical approach that organizes robotic behaviors across multiple levels of abstraction. This allows for a sophisticated orchestration of actions, where high-level goals are broken down into manageable sub-tasks that guide motion generation and decision-making processes. This approach aims to overcome traditional challenges associated with aligning cognitive models to complex geometric scenes, thus improving the robot’s responsiveness and operational efficiency.

Additionally, the module emphasizes bimanual coordination, focusing on synchronizing the robot's dual arms for enhanced dexterity. By encoding arm-specific affordances and actions within a spatiotemporal context, the robot can perform tasks more coherently and efficiently. An active semantic graph will represent possible sub-tasks related to object interactions, evolving from observed human behaviors to create a knowledge base that informs decision-making in real time.

To bridge the gap between high-level planning and execution, the module explores natural language representations of tasks, making them more intuitive for human operators. By integrating large language models (LLMs) with the robot's cognitive architecture, the system will enable seamless communication, allowing the robot to interpret natural language commands and adapt its behavior accordingly. This enables a collaborative environment where human and robot actions align for improved productivity and safety.

3.2.4 MANiBOT bi-manual robot platform and cognitive mechatronics module

The MANiBOT Bi-manual Robot Platform and Cognitive Mechatronics (WP6) focuses on developing advanced hardware and sensing capabilities that enhance the robot's interaction with complex environments and improve safety in human-robot collaboration. A primary objective is the integration of proximity sensors that create a "sensing aura" around the robot, enabling it to anticipate collisions and adjust movement speeds based on human presence. By optimizing sensor coverage and processing, the proximity system will allow faster, more responsive actions, ensuring compliance with safety regulations while enhancing procedural efficiency.

Complementing the proximity sensors, MANiBOT aims to implement high-resolution tactile sensing, essential for refined manipulation tasks requiring human-like precision. MANiBOT will leverage deep learning models trained in simulation to create tactile sensing policies that can transfer seamlessly to real-world environments. This open-source tactile technology, combined with dexterous control, enables MANiBOT to advance manipulation capabilities, particularly for in-hand object handling that requires fine touch. The use of standardized platforms will facilitate access and scalability of these technologies across project partners, addressing the limited availability of tactile-equipped hardware.

The bimanual mobile robot platform will be developed, aiming to revolutionize robotic manipulation through a focus on safety and adaptability in diverse environments. This platform will feature two manipulators designed to work collaboratively, enhancing the robot's ability to perform complex tasks while ensuring safe interaction with humans and objects. By integrating advanced sensing capabilities, the robot will be equipped to perceive its surroundings effectively, allowing it to navigate and manipulate objects with precision and reliability. A key aspect of this platform is its ability to handle a variety of tasks, including the transfer of large or heavy items, through innovative mechanisms that promote efficiency and safety. Additionally, the robot will incorporate solutions for maintaining energy autonomy, enabling prolonged operation without interruptions.

4 Use Case-specific Functional Analysis of the System

4.1 Use Case Analysis

As outlined in deliverable D2.2, each use case has been subdivided into scenario-oriented sub-use cases, detailing the tasks the MANiBOT system is expected to perform. Additionally, user requirements have been aligned with one or more sub-use cases, as shown in Table 1. Each sub-use case is specifically characterized by the system functional requirements, which define the actions the system or robot must perform or the processing needed to complete the task.

As noted in D2.2, following the grouping of User Requirements based on deployment priorities, the relevant Use Cases selected for validation and deployment were adjusted accordingly. The chosen use cases for validation and deployment, based on user feedback and requirement priorities, have a High deployment priority, while other use cases considered for validation under conditional circumstances have Medium or LOW deployment priority.

Table 1: Mapping of User Requirements to Sub-use Cases

High-level Use Case	Sub-Use Case ID and Title	Deployment Priority level	Related requirements
UC1.Product replenishment in retail stores (MASOUTIS)	**SubUC1.1.** Movement from resting position/charging station to the target replenishment cart	*HIGH*	UR1,UR3,UR5, UR6,UR7,UR20, UR24,UR27,UR 28,UR29,UR30, UR32,UR33,UR 34,UR35,U36
	SubUC1.2. Identification of the products in the replenishment cart	*HIGH*	UR2,UR8,UR12, UR13,UR20,UR 21,UR23,UR24, UR25,UR27,UR 28,UR29,UR30, UR32,UR33,UR 34,UR35,UR36
	SubUC1.3. Identification of the right position for the product on the shelf	*HIGH*	UR3,UR4,UR5, UR6,UR7,UR8, UR12,UR13,UR 20,UR22,UR23, UR24,UR26,UR 27,UR28,UR29, UR30,UR2,UR3 2,UR33,UR34,U R35,UR36
	SubUC1.4. Replenishment of the shelves	*HIGH*	UR3,UR5,UR6, UR7,UR8,UR9, UR10,UR11,UR 12,UR13,UR14, UR15,UR16,UR 17,UR20,UR21, UR24,UR25,UR 26,UR27,UR28, UR29,UR30,UR

High-level Use Case	Sub-Use Case ID and Title	Deployment Priority level	Related requirements
			31,UR32,UR33,UR34,UR35,UR36
	SubUC1.5. Movement back to the resting position/charging station	*MEDIUM*	UR3,UR5,UR6,UR7,UR18,UR19,UR20,UR27,UR28,UR29,UR30,UR32,UR33,UR34,UR35,UR36
UC2. Product replenishment in retail stores (SDI)	**SubUC2.1.** Movement from resting position/charging station to target destination for replenishment	*HIGH*	UR03,UR06,UR07,UR24,UR29,UR30,UR32,UR35,UR36,UR54,UR55,UR56,UR57,UR59,UR61,UR62,UR64,UR65,UR68,UR83,UR84, UR85
	SubUC2.2. Identification of the products on mixed pallet	*HIGH*	UR03,UR06,UR07,UR09,UR21,UR24,UR29,UR30,UR32,UR35,UR36,UR41,UR42,UR45,UR46,UR54,UR55,UR56,UR57,UR 59,UR60,UR61,UR62,UR63,UR64,UR65,UR68,UR83,UR84,UR85
	SubUC2.3. Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	*HIGH*	UR06,UR07,UR21,UR24,UR29,UR30,UR32,UR35,UR36,UR37,UR38,UR39,UR43,UR54,UR55,UR56,UR57,UR59,UR60,UR61,UR62,UR63,UR64,UR65,UR68,UR69,UR70,UR71,UR75,UR83,UR84,UR85
	SubUC2.4. Preparation of the products before replenishment	*MEDIUM*	UR03,UR06,UR07,UR13,UR21,UR23,UR24,UR

High-level Use Case	Sub-Use Case ID and Title	Deployment Priority level	Related requirements
			26,UR29,UR30,UR32,UR35,UR36,UR54,UR55,UR56,UR57,UR59,UR60,UR61,UR62,UR63,UR64,UR65,UR66,UR67,UR68,UR74,UR77,UR78,UR79,*UR89,UR81,UR82,UR83,UR84,UR85*
	SubUC2.5. Product replenishment (applying FIFO principle and overstock management)	*HIGH*	UR03,UR06,UR07,UR11,UR12,UR21,UR24,UR26,UR28,UR31,UR29,UR30,UR32,UR35,UR36,UR37,UR40,UR44,UR45,UR47,UR48,UR49,UR50,UR51,UR52,UR53,UR54,UR55,UR56,UR57,UR59,UR60,UR61,UR62,UR64,UR65,UR68,UR69,UR70,UR71,UR72,UR73,UR74,UR76,UR77,UR78,UR79,UR80,UR81,UR83,UR84,UR85
	SubUC2.6. Movement back to the resting position/charging station	*HIGH*	UR03,UR06,UR07,UR24,UR29,UR30,UR32,UR35,UR36,UR54,UR55,UR56,UR57,UR59,UR61,UR62,UR64,UR65,UR68,UR83,UR84,UR85
UC3. Baggage loading and unloading from conveyor belts to carts (FG)	**SubUC3.1.** Preparing for loading	*HIGH*	UR1, UR2, UR3, UR4,UR5,UR15, UR19,UR21,UR22

High-level Use Case	Sub-Use Case ID and Title	Deployment Priority level	Related requirements
	SubUC3.2. Recognizing Baggage for assigned flight	*HIGH*	UR6,UR7,UR8, UR23,UR24,UR 25
	SubUC3.3. Loading of allocated baggage on the cart	*HIGH*	UR5,UR6,UR9, UR11,UR13,UR 14,UR16,UR17, UR18,UR19,UR 23,UR27,UR28, UR29,UR32
	SubUC3.4. Movement back to resting position/charging point	*LOW*	UR1,UR2, UR31
UC4. Baggage loading and unloading from carts to conveyor belts (FG)	**SubUC4.1.** Movement from resting position/charging station to the target position	*MEDIUM*	UR1, UR2, UR5, UR19, UR21, UR22
	SubUC4.2. Preparing for unloading	*HIGH*	UR2, UR3, UR4, UR15, UR19, UR21, UR22
	SubUC4.3. Unloading from Cart to Conveyor belt	*HIGH*	UR5, UR6, UR7, UR8, UR9, UR10, UR11, UR12, UR13, UR14, UR16, UR18, UR23, UR24, UR25, UR27, UR28, UR29, UR32
	SubUC4.4. Moving to next Baggage cart	*MEDIUM*	UR1, UR2, UR5, UR17, UR19
	SubUC4.5. Movement back to resting position/charging point	*LOW*	UR1, UR2, UR17, UR31

4.2 Functional Design of the System

System functional requirements should be established with reference to the robotic functionalities that should be achieved with either software or hardware solutions in order to assign technical specifications to sub-use cases. As a result, every sub-use case has been examined to produce a list of tasks that must be completed. This section provides a summary of all system functional requirements pertaining to particular sub-use cases. Translating user requirements and use cases into precise technical specifications is the primary goal of system functional requirements. Additionally, it must be noted that the hardware and software requirements for each sub-use case, the technical specifications, are the ones that will be developed and implemented throughout the MANiBOT project.

In order to identify the robot functionalities that will unavoidably be repeated in numerous sub-use cases, the list of identified system functional requirements was initially kept as abstract as possible. As a result, each system functional requirement can be used in multiple sub-use cases, as indicated in the following table (Table 2).

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement No 101120823

Table 2: Summary of the identified Perception/Cognition system functional requirements and their respective sub-use cases

No. /ID	System Functional Requirement	Related Sub-Use Cases	Feasibility/ Relevance
SFR01	Recognize the product to be replenished	SubUC-1.2 (Step 1): Identification of the products in the replenishment cart SubUC-1.3 (Step 1): Identification of the right position for the product on the shelf SubUC-2.2 (Step 1): Identification of the products on mixed pallet SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	High
SFR02	Recognize expired and damaged products	SubUC-1.2 (Step 1): Identification of the products in the replenishment cart SubUC-1.3 (Step 1): Identification of the right position for the product on the shelf SubUC-1.4 (Step 2): Replenishment of the shelves SubUC-2.4 (Step 2): Preparation of the products before replenishment	Medium
SFR03	Recognize the replenishment cart	SubUC-1.1 (Step 2): Movement from resting position/charging station to the target replenishment cart	Medium
SFR04	Recognize dimensions and orientation of each product per box	SubUC-1.2 (Step 2): Identification of the products on mixed pallet	High
SFR05	Recognize dimensions and orientation of boxes per pallet	SubUC-2.2 (Step 3): Identification of the products on mixed pallet	High
SFR06	Recognize the type of box	SubUC-2.2 (Step 2): Identification of the products on mixed pallet	Medium
SFR07	Recognize electronic shelf labels and extract content	SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan) SubUC-2.5 (Step 2): Product replenishment (applying FIFO principle and overstock management)	Low
SFR08	Recognize the remaining space on shelves	SubUC-2.5 (Step 3): Product replenishment (applying FIFO principle and overstock management)	High
SFR09	Detect the appropriate baggage cart	SubUC-3.3 (Step 3): Loading of allocated baggage on the cart SubUC-4.2 (Step 2): Preparing for unloading SubUC-4.4 (Step 2): Moving to next Baggage cart	Medium
SFR10	Detects moving baggage on the conveyor belt	SubUC-3.2 (Step 1): Recognizing Baggage for assigned flight	High
SFR11	Recognize the front side (i.e. customer facing side) for each box	SubUC-1.4 (Step 5): Replenishment of the shelves	Medium

SFR12	Detects each baggage separately identifying its shape and dimensions	SubUC-3.3 (Step 1): Loading of allocated baggage on the cart SubUC-4.3 (Step 2): Unloading from Cart to Conveyor belt	High
SFR13	Recognize how many boxes of a specific product item can fit on the shelf	SubUC-1.4 (Step 4): Replenishment of the shelves SubUC-2.5 (Step 4): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR14	Recognize when a baggage cart is full or empty	SubUC-3.3 (Step 6): Loading of allocated baggage on the cart SubUC-4.3 (Step 7): Unloading from Cart to Conveyor belt	High
SFR15	Recognize damaged luggage for the arrivals	SubUC-3.2 (Step 2): Recognizing Baggage for assigned flight SubUC-4.2 (Step 2): Preparing for unloading	Medium
SFR16	Recognize the luggage material (soft or hard)	SubUC-3.2 (Step 4): Recognizing Baggage for assigned flight SubUC-4.3 (Step 2): Unloading from Cart to Conveyor belt	Medium
SFR17	Ability to determine the optimal quantity of products for placement at the forefront of the shelf	SubUC-1.4 (Step 4): Replenishment of the shelves	Medium
SFR18	Recognize misplaced products on the shelves	SubUC-1.4 (Step 2): Replenishment of the shelves SubUC-2.5 (Step 1): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR19	Recognize the free space at the target destination (baggage cart or belt) to place the luggage	SubUC-3.3 (Step 3): Loading of allocated baggage on the cart SubUC-4.3 (Step 5): Unloading from Cart to Conveyor belt	High
SFR20	Performs quality check on products (flowers are not withered, fruits and vegetables are not deviated)	SubUC-2.4 (Step 1): Preparation of the products before replenishment	Low
SFR21	Recognizes the location for the baggage placement	SubUC-3.3 (Step 3): Loading of allocated baggage on the cart SubUC-4.3 (Step 5): Unloading from Cart to Conveyor belt	High
SFR22	Recognize the proximate area and shelves and compare the predefined assortment planogram with the actual placement of the ESLs	SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	Low
SFR23	Recognize boxes' arrangement/placement on pallet (top to bottom)	SubUC-2.2 (Step 1): Identification of the products on mixed pallet SubUC-2.5 (Step 1): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR24	Recognize product assortment on shelves according to ESLs and per product defined area (by assortment plan)	SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	Medium
SFR25	Identify the correct destination on the shelf (through ESL) given the	SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	High

	prior recognition of the mixed pallet and the respective product on the mixed pallet		
SFR26	Inspect the state and replenishment feasibility of each package (per product)	SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan) SubUC-1.3 (Step 1): Identification of the right position for the product on the shelf	Medium
SFR27	Recognize the mixed pallet	SubUC-2.2 (Step 1): Identification of the products on mixed pallet SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	High
SFR28	Recognize and track humans trajectory	SubUC-1.1 (Step 2): Movement from resting position/charging station to the target replenishment cart,SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.1 (Step 2): Movement from resting position/charging station to target destination for replenishment SubUC-2.6 (Step 2): Movement back to the resting position/charging station, SubUC-3.1 (Step 3): Preparing for loading SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.2 (Step 4): Preparing for unloading SubUC-4.4 (Step 1): Moving to next Baggage cart SubUC-4.5 (Step 2): Movement back to resting position/charging point	Medium
SFR29	Detects proximity of moving baggage tractor with loaded carts	SubUC-4.2 (Step 1): Preparing for unloading	Medium
SFR30	The robot decides which baggage to remove with least disturbance to surrounding baggage	SubUC-4.3 (Step 1): Unloading from Cart to Conveyor belt	High
SFR31	Platform 2D/3D localization	SubUC-1.1 (Step 2): Movement from resting position/charging station to the target replenishment cart SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.1 (Step 2): Movement from resting position/charging station to target destination for replenishment SubUC-2.6 (Step 2): Movement back to the resting position/charging station SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.4 (Step 1): Moving to next Baggage cart SubUC-4.5 (Step 2): Movement back to resting position/charging point	High
SFR32	Platform Path planning ability	SubUC-1.1 (Step 2): Movement from resting position/charging station to the target replenishment cart	High

		SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.1 (Step 2): Movement from resting position/charging station to target destination for replenishment SubUC-2.6 (Step 2): Movement back to the resting position/charging station SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.4 (Step 1): Moving to next Baggage cart SubUC-4.5 (Step 2): Movement back to resting position/charging point	
SFR33	Platform human-aware navigation	SubUC-1.1 (Step 2): Movement from resting position/charging station to the target replenishment cart SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.1 (Step 2): Movement from resting position/charging station to target destination for replenishment SubUC-2.6 (Step 2): Movement back to the resting position/charging station SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.4 (Step 1): Moving to next Baggage cart SubUC-4.5 (Step 2): Movement back to resting position/charging point	High
SFR34	Ability to move on the pavement area alongside the conveyor belt	SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.4 (Step 1): Moving to next Baggage cart SubUC-4.5 (Step 2): Movement back to resting position/charging point	High
SFR35	Ability to step up to the pavement area (height 15cm) around the conveyor belt (with or without the use of a ramp)	SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.5 (Step 2): Movement back to resting position/charging point	Low
SFR36	Navigate avoiding restricted areas	SubUC-1.1 (Step 2): Movement from resting position/charging station to the target replenishment cart SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.1 (Step 2): Movement from resting position/charging station to target destination for replenishment SubUC-2.6 (Step 2): Movement back to the resting position/charging station SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.4 (Step 1): Moving to next Baggage cart SubUC-4.5 (Step 2): Movement back to resting position/charging point	Medium
SFR37	The robot selects a desired position regarding the baggage cart	SubUC-3.1 (Step 2): Preparing for loading SubUC-4.2 (Step 3): Preparing for unloading	Medium

SFR38	Grasp products from boxes	SubUC-1.4 (Step 6): Replenishment of the shelves	High
SFR39	Ability to pick and place single items	SubUC-2.5 (Step 4): Product replenishment (applying FIFO principle and overstock management)	High
SFR40	Ability to pick up boxes with max weight up to 10 kg	SubUC-2.5 (Step 3): Product replenishment (applying FIFO principle and overstock management)	High
SFR41	Ability to pick up boxes without damaging them	SubUC-2.5 (Step 3): Product replenishment (applying FIFO principle and overstock management)	High
SFR42	Place the products in the correct positions	SubUC-1.4 (Step 5): Replenishment of the shelves SubUC-2.4 (Step 2): Preparation of the products before replenishment SubUC-2.5 (Step 4) Product replenishment (applying FIFO principle and overstock management)	High
SFR43	Place the products that expire earlier in the front of the shelf ahead of newly received items	SubUC-1.4 (Step 5): Replenishment of the shelves SubUC-2.5 (Step 4): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR44	Place the products in a specified orientation, ensuring the front side of the product faces the customer	SubUC-1.4 (Step 5): Replenishment of the shelves	Medium
SFR45	Place momentarily the boxes on its equipped table or another interim storage position	SubUC-1.4 (Step 3): Replenishment of the shelves SubUC-2.5 (Step 3): Product replenishment (applying FIFO principle and overstock management) SubUC 3.3 (Step 2): Loading of allocated baggage on the cart SubUC 4.3 (Step 5): Unloading from Cart to Conveyor belt	High
SFR46	Manipulate and replenish tear strip boxes, regular product boxes, wrapped boxes, closed top boxes, and brown cardboard boxes	SubUC-2.5 (Step 2): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR47	Ability to close and open nested shelves, fridges, or freezing units	SubUC-2.5 (Step 2): Product replenishment (applying FIFO principle and overstock management)	Low
SFR48	Ability to grab newspapers and magazines enclosed in brown boxes and place them in storage positions	SubUC-2.5 (Step 2): Product replenishment (applying FIFO principle and overstock management)	Low
SFR49	Ability to open closed boxes and stock their content in the corresponding shelves	SubUC-2.5 (Step 2): Product replenishment (applying FIFO principle and overstock management)	Low
SFR50	Grasp the luggage that is made from soft material gently	SubUC-4.3 (Step 3): Unloading from Cart to Conveyor belt SubUC-3.3 (Step 4): Loading of allocated baggage on the cart	Medium

SFR51	Ability to prepare the products/SKUs (e.g., removing cardboard or plastic foil toppers)	SubUC-2.4 (Step 1): Preparation of the products before replenishment	Low
SFR52	Place misplaced products in an interim storage position or to the cart or to the right shelf position	SubUC-1.4 (Step 3): Replenishment of the shelves SubUC-2.5 (Step 3): Product replenishment (applying FIFO principle and overstock management)	High
SFR53	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position	SubUC-1.4 (Step 7): Replenishment of the shelves SubUC-2.5 (Step 3): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR54	Place the products in the right position without using a planogram or consult it when it is available	SubUC1.3 (Step 2): Identification of the right position for the product on the shelf	Medium
SFR55	Place the luggage made of hard material below the luggage made of soft material	SubUC-3.3 (Step 4): Loading of allocated baggage on the cart SubUC-4.3 (Step 3): Unloading from Cart to Conveyor belt	Medium
SFR56	Place the luggage with a distance between them on the reclaim belt for the arrivals	SubUC 4.3 (Step 5): Unloading from Cart to Conveyor belt	Medium
SFR57	Place the luggage within the boundaries of the belt/cart	SubUC-3.3 (Step 4): Loading of allocated baggage on the cart SubUC-4.3 (Step 5, 6): Unloading from Cart to Conveyor belt	High
SFR58	Handle luggage only for the flight that has been allocated to	SubUC-4.2 (Step 2): Preparing for unloading SubUC-4.3 (Step 2): Unloading from Cart to Conveyor belt	Medium
SFR59	Ability to restock shelves with product/SKUs from the pallet (top to bottom)	SubUC-1.4 (Step 5): Replenishment of the shelves SubUC-2.5 (Step 2): Product replenishment (applying FIFO principle and overstock management)	High
SFR60	Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism	SubUC-1.4 (Step 3, 5, 6, 7): Replenishment of the shelves SubUC-2.5 (Step 3, 4): Product replenishment (applying FIFO principle and overstock management) SubUC-3.3 (Step 4): Loading of allocated baggage on the cart SubUC-4.3 (Step 5): Unloading from Cart to Conveyor belt	Medium
SFR61	Place empty (cardboard) boxes from replenished items into a bin cart located next to the pallet	SubUC-2.5 (Step 5): Product replenishment (applying FIFO principle and overstock management)	Low
SFR62	Manipulate the baggage to detect the bagtag	SubUC-3.2 (Step 5): Recognizing Baggage for assigned flight	High
SFR63	Ability to avoid collision with moving or static objects	SubUC-1.4 (Step 3, 5, 6, 7): Replenishment of the shelves	High

		SubUC-2.5 (Step 3, 4, 5): Product replenishment (applying FIFO principle and overstock management) SubUC-3.3 (Step 4): Loading of allocated baggage on the cart SubUC-4.3 (Step 3, 5): Unloading from Cart to Conveyor belt	
SFR64	Ability to avoid collisions with itself in bimanual tasks	SubUC-1.4 (Step 3, 5, 6, 7): Replenishment of the shelves SubUC-2.5 (Step 3, 4, 5): Product replenishment (applying FIFO principle and overstock management) SubUC-3.3 (Step 4): Loading of allocated baggage on the cart SubUC-4.3 (Step 3, 5): Unloading from Cart to Conveyor belt	High
SFR65	Detects the optimal contact points in boxes, bags, and products to grasp the item	SubUC-1.4 (Step 3,5,6): Replenishment of the shelves SubUC-2.5 (Step 3, 4): Product replenishment (applying FIFO principle and overstock management) SubUC-3.3 (Step 4): Loading of allocated baggage on the cart SubUC-4.3 (Step 2,3): Unloading from Cart to Conveyor belt	High
SFR66	Robot pauses its current task if employee or customer is too close to it	SubUC-1.1 (Step 2): Movement from resting position/charging station to the target replenishment cart SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.1 (Step 2): Movement from resting position/charging station to target destination for replenishment SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan) SubUC-2.4 (Step 1,2): Preparation of the products before replenishment SubUC-2.6 (Step 2): Movement back to the resting position/charging station SubUC-3.4 (Step 2): Movement back to resting position/charging point SubUC-4.1 (Step 2): Movement from resting position/charging station to the target position SubUC-4.4 (Step 1): Moving to next Baggage cart SubUC-4.5 (Step 2): Movement back to resting position/charging point	High
SFR67	Robots pauses and resumes tasks according to user's command	SubUC-1.1 (Step 1): Movement from resting position/charging station to the target replenishment cart SubUC-2.1 (Step 1): Movement from resting position/charging station to target destination for replenishment SubUC-2.6 (Step 2): Movement back to the resting position/charging station SubUC-3.1 (Step 1): Preparing for loading SubUC-4.1 (Step 1): Movement from resting position/charging station to the target position	Medium
SFR68	Ability to complete restocking tasks for shelves within a mobility radius (parameter with default value e.g., 5 meters) from the mixed pallet	SubUC-2.3 (Step 1): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan) SubUC-2.5 (Step 3, 4): Product replenishment (applying FIFO principle and overstock management)	Medium

SFR69	Decides from where it can leave the pavement track (either because it has been instructed to by the operator or if it needs to be charged)	SubUC-3.4 (Step 1): Movement back to resting position/charging point SubUC-4.5 (Step 1): Movement back to resting position/charging point	Medium
SFR70	User defining replenishment task and destination	SubUC-1.1 (Step 1): Movement from resting position/charging station to the target replenishment cart SubUC-2.1 (Step 1): Movement from resting position/charging station to target destination for replenishment	Medium
SFR71	User defining task and destination in the airport belts	SubUC-3.1 (Step 1): Preparing for loading SubUC-4.1 (Step 1): Movement from resting position/charging station to the target position	Medium
SFR72	User commanding through interaction interface the robot to move to specified location (shelf or belt)	SubUC-1.1 (Step 1): Movement from resting position/charging station to the target replenishment cart SubUC-2.1 (Step 1): Movement from resting position/charging station to target destination for replenishment SubUC-3.1 (Step 1): Preparing for loading SubUC-4.1 (Step 1): Movement from resting position/charging station to the target position	Medium
SFR73	Interaction interface notifies the user about the restocking task progress	SubUC-1.5 (Step 1): Movement back to the resting position/charging station SubUC-2.6 (Step 1): Movement back to the resting position/charging station	Medium
SFR74	Interaction interface notifies the user if the robot needs help with the identification of a product, bag, etc.	SubUC-1.2 (Step 1): Identification of the products in the replenishment cart SubUC-1.3 (Step 2): Identification of the right position for the product on the shelf SubUC-2.2 (Step 1): Identification of the products on mixed pallet SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	Medium
SFR75	Interaction interface notifies the user for help when a product falls on the floor	SubUC-1.2 (Step 1): Identification of the products in the replenishment cart SubUC-1.3 (Step 2): Identification of the right position for the product on the shelf SubUC-2.2 (Step 1): Identification of the products on mixed pallet SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	Medium
SFR76	Ability to give a signal (either audio or visual) when robot's workspace is invaded by a human	SubUC-1.3 (Step 1, 3): Identification of the right position for the product on the shelf SubUC-1.4 (Step 1): Replenishment of the shelves SubUC-2.3 (Step 1): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	High
SFR77	Interaction interface notifies the user when the replenishment process is complete	SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.6 (Step 2): Movement back to the resting position/charging station	Medium
SFR78	Interaction interface notifies the user if any of the processes failed	SubUC-1.1 (Step 1, 2): Movement from resting position/charging station to the target replenishment cart	Medium

		SubUC-1.2 (Step 1, 2): Identification of the products in the replenishment cart SubUC-1.3 (Step 2, 3): Identification of the right position for the product on the shelf SubUC-1.4 (All Steps): Replenishment of the shelves SubUC-1.5 (Step 2): Movement back to the resting position/charging station SubUC-2.1 (Step 2): Movement from resting position/charging station to the target destination for replenishment SubUC-2.2 (Step 2): Identification of the products on mixed pallet	
SFR79	Interaction interface notifies the user about the absence of planogram and difficulty in defining the right product position	SubUC1.3 (Step 2): Identification of the right position for the product on the shelf	Medium
SFR80	Interaction interface alerts when the bag it is manipulating is out of its capabilities (too heavy, too long, no handles, out of reach, etc.)	SubUC-3.3 (Step 4): Loading of allocated baggage on the cart SubUC-4.3 (Step 5): Unloading from Cart to Conveyor belt	Medium
SFR81	Interaction interface alerts when no bagtag found	SubUC-3.2 (Step 6): Recognizing Baggage for assigned flight	Medium
SFR82	The robot alerts the operator if no other cart is found, goes in standby mode and awaits input from the operator	SubUC-4.4 (Step 3) Moving to next Baggage cart	Medium
SFR83	Notifies the user through the UI for the table/interim storage state	SubUC-2.5 (Step 4): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR84	Robot stores/retrieves the bagtag information of the placed baggage in/from the database	SubUC-3.2 (Step 3): Recognizing Baggage for assigned flight SubUC-3.3 (Step 5): Loading of allocated baggage on the cart SubUC-4.3 (Step 4): Unloading from Cart to Conveyor belt	High
SFR85	Scans barcode and retrieves information of the paper label and checks if it corresponds to the product to be replenished	SubUC-1.3 (Step 2): Identification of the right position for the product on the shelf	Medium
SFR86	Retrieves information about the ESL tags	SubUC-2.3 (Step 2): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	Low
SFR87	If a mismatch identified, log the distance between the position of a product in the assortment plan and the position of the ESL	SubUC-2.3 (Step 1): Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	Low
SFR88	Keep count of the number of baggage loaded	SubUC-3.3 (Step 5): Loading of allocated baggage on the cart SubUC-4.3 (Step 4): Unloading from Cart to Conveyor belt	High

SFR89	Scans the bag tag (when placing the baggage) for BTRS (Baggage Tracking and Reconciliation System) purposes	SubUC-3.3 (Step 5): Loading of allocated baggage on the cart SubUC-4.3 (Step 4): Unloading from Cart to Conveyor belt	High
SFR90	Records and logs in the database the products that are not suitable for restocking (surplus, expired, damaged)	SubUC-1.4 (Step 9): Replenishment of the shelves SubUC-2.5 (Step 4): Product replenishment (applying FIFO principle and overstock management)	Medium
SFR91	Specific AI modules of the robot are updated while maintaining organisation and persons privacy	SubUC-1.2 (Step 1, 2): Identification of the products in the replenishment cart SubUC-1.3 (Step 2, 3): Identification of the right position for the product on the shelf SubUC-1.4 (All Steps): Replenishment of the shelves SubUC-2.2 (Step 2): Identification of the products on mixed pallet SubUC-3.3 (Step 6): Loading of allocated baggage on the cart SubUC-4.3 (Step 7): Unloading from Cart to Conveyor belt	High
SFR92	The outputs of specific AI modules of the robot are explained through the UI (User Interface)	SubUC-1.2 (Step 1, 2): Identification of the products in the replenishment cart SubUC-1.3 (Step 2, 3): Identification of the right position for the product on the shelf SubUC-1.4 (All Steps): Replenishment of the shelves SubUC-2.2 (Step 2): Identification of the products on mixed pallet SubUC-3.3 (Step 6): Loading of allocated baggage on the cart SubUC-4.3 (Step 7): Unloading from Cart to Conveyor belt	High

5 Functional View of the MANiBOT System

This chapter presents a functional view of the system architecture. The system's architecture specifies the components and their functionalities. A description of the system's functional structure is provided, demonstrating how the system performs the required functions. The functional structure usually consists of external entities, interfaces, connectors, and functional elements.

Established system components with specific capabilities that expose defined interfaces that enable them to be connected to other components are characterized as functional elements. Applications, collection of data, software modules, and subsystems are all regarded as functional components. Interfaces are specifications that specify the information exchange between two components. An interface, which is defined by the inputs, outputs, the operation offered, and the interaction required for the operation, allows another component to access the functions of another component. Software applications, hardware components, or any other entity with which the system communicates are examples of external entities. They are referred to as dependencies on other components or systems.

The MANiBOT system's functional components are introduced in the sections below, along with how they relate to one another. Their primary functions are explained, along with how they depend on other components. Figure 5 provides the functional architecture of the MANiBOT system. The conceptual architecture shown in Section 3 is converted into software modules and dependencies in this figure. Headed lines indicate the directed dependencies between the modules. The WP of each module has been indicated by colours, specifically:

- WP3 module are orange
-
- WP4 modules are green
- WP5 modules are blue
- WP6 modules are purple
- Modules that belong to more than 2 WP are blue without filled boxes.

The MANiBOT system consists of 19 functional components which are listed below:

- Visual Sensing Module (CERTH/TUW) – Section 5.1
    - Object Detection Module (T3.1)
    - 6DoF Object Pose Estimation Module (T3.1)
    - Human Pose Estimation and Tracking Module (T3.5)
- Tactile Sensing Module (UoB) (T3.4) – Section 5.2
    - Contact Pose and Force Estimation Module
    - Contact Detection Module
- Predictive Proximity Sensing Module (SSSA/CERTH) (T3.5) – Section 5.3
    - Human Detection Module
    - Human Intention Predictor
- Multimodal Sensing Orchestration Module (CERTH) (T3.6) – Section 5.4
    - Robot Platform Proprioceptive Data Analyzer
    - Robot Platform Exteroceptive Data Analyzer
    - Multimodal Data Orchestrator
    - Target Object Semantic Segmentation Module (T3.2)
    - Predictive Structural Inference Engine (T3.3)
- Visual Scene Understanding Module (CERTH) – Section 5.5
    - Target Object Semantic Segmentation Module (T3.2)

  - Predictive Structural Inference Engine (T3.3)
- Federated XAI Module (THL) (T3.7) – Section 5.6
  - Federated Learning Module
  - Explainable AI (XAI) Module
- Mobile bimanual low-level controller (UBU) (T6.4) – Section 5.7
  - Motion Control Module
  - Bimanual Control Module
  - Conveyor Belt Control Module
- Robot Navigation Module (CERTH) (T4.1) – Section 5.8
  - Localization Module
  - Global Planning Module
  - Local Planning Module
- Multi-modal adaptive control Module (AUTH) – Section 5.9
  - Bimanual Manipulation Primitives Controller (T4.4, T4.5)
  - Unimanual Manipulation Primitives Controller (T4.4, T4.5)
  - Grasping Controller (T4.5)
- Mobile Bimanual Coordination Module (AUTH) (T4.2) – Section 5.10
  - Dual Arm and Mobile Platform Coordinator
  - Performance Optimization Module
  - Collision Avoidance Module
- Contact Reaching Module (AUTH) (T4.3) – Section 5.11
  - Contact Selection Planner
  - Contact Reaching Controller
- Hybrid Control Module (AUTH) (T4.6) – Section 5.12
  - Hybrid Controller
- Adaptive Modelling Module (CERTH) (T5.1) – Section 5.13
  - Semantic 3D SLAM module
  - Operational Environments Adaptable Model Generator
  - Lifelong Semantic Adaptation Module
- Scene & Task Graph Generator (TUDa) (T5.2) – Section 5.14
  - Scene Graph Generator
  - Task Graph Generator
  - Grounding Module
  - Primitive Sequencing Module
- Learning to Plan from Demonstrations Module (TUDa) (T5.3) – Section 5.15
  - Hybrid RL High-level Planner
  - Low-level Primitives Adapter
- Adaptive Task Planner (TUDa) (T5.4) – Section 5.16
  - Manipulation Task Assessment & Adaptation Module
  - Recovery Strategies Enforcer
- Bi-manual Manipulations Orchestrator (TUDa) (T5.5) – Section 5.17
  - High-level Task Orchestrator
  - Natural Language Feedback Integrator
- MANiBOT HRI Interface (CERTH) (T5.6) – Section 5.18
  - HRI GUI
  - Back-end server
  - AR Module
  - Task Scheduling and Monitoring Module
- Data Management Module (CERTH/THL) – Section 5.19

- HRI Data Manager (T5.6)
- Robotic Platform Data Manager (T6.1)
- Federated Learning Data Manager (T3.7)

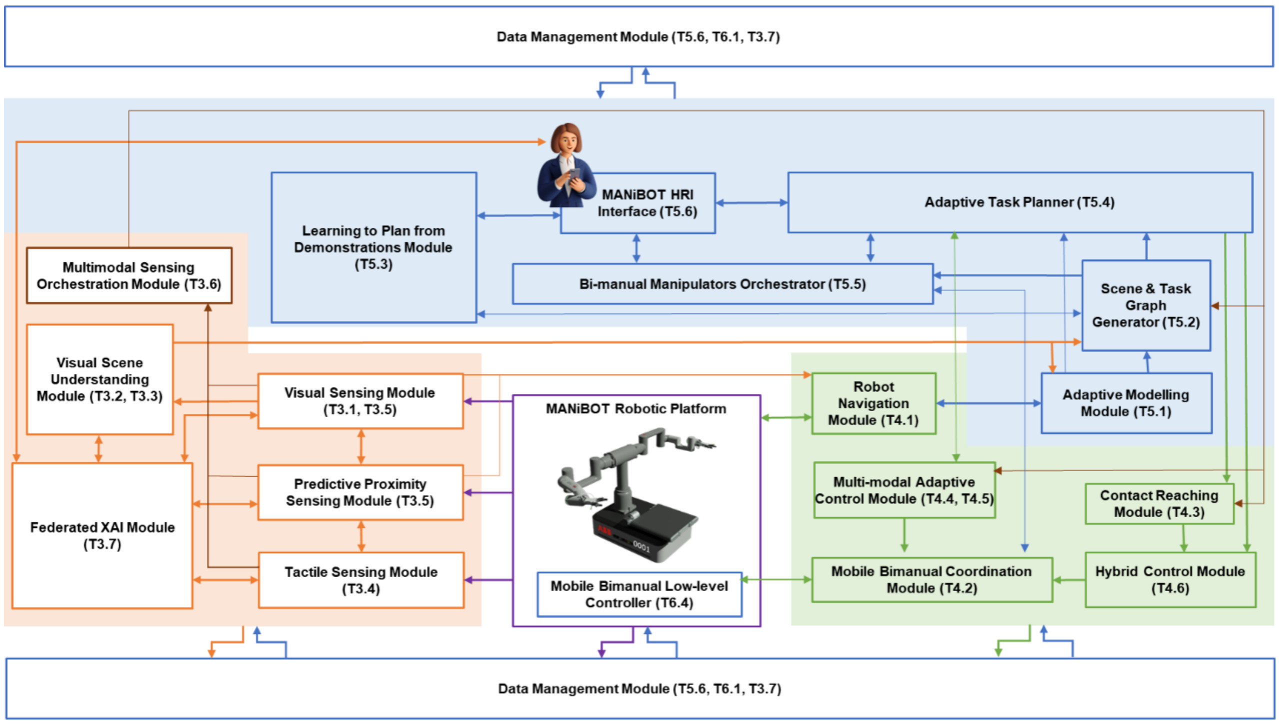

Data Management Module (T5.6, T6.1, T3.7)
MANiBOT HRI Interface (T5.6)
Adaptive Task Planner (T5.4)
Learning to Plan from Demonstrations Module (T5.3)
Multimodal Sensing Orchestration Module (T3.6)
Bi-manual Manipulators Orchestrator (T5.5)
Scene & Task Graph Generator (T5.2)
Visual Scene Understanding Module (T3.2, T3.3)
Visual Sensing Module (T3.1, T3.5)
MANiBOT Robotic Platform
Robot Navigation Module (T4.1)
Adaptive Modelling Module (T5.1)
Predictive Proximity Sensing Module (T3.5)
Multi-modal Adaptive Control Module (T4.4, T4.5)
Contact Reaching Module (T4.3)
Federated XAI Module (T3.7)
Tactile Sensing Module (T3.4)
Mobile Bimanual Low-level Controller (T6.4)
Mobile Bimanual Coordination Module (T4.2)
Hybrid Control Module (T4.6)
Data Management Module (T5.6, T6.1, T3.7)

Figure 5 MANiBOT Functional Architecture

5.1 Visual Sensing Module

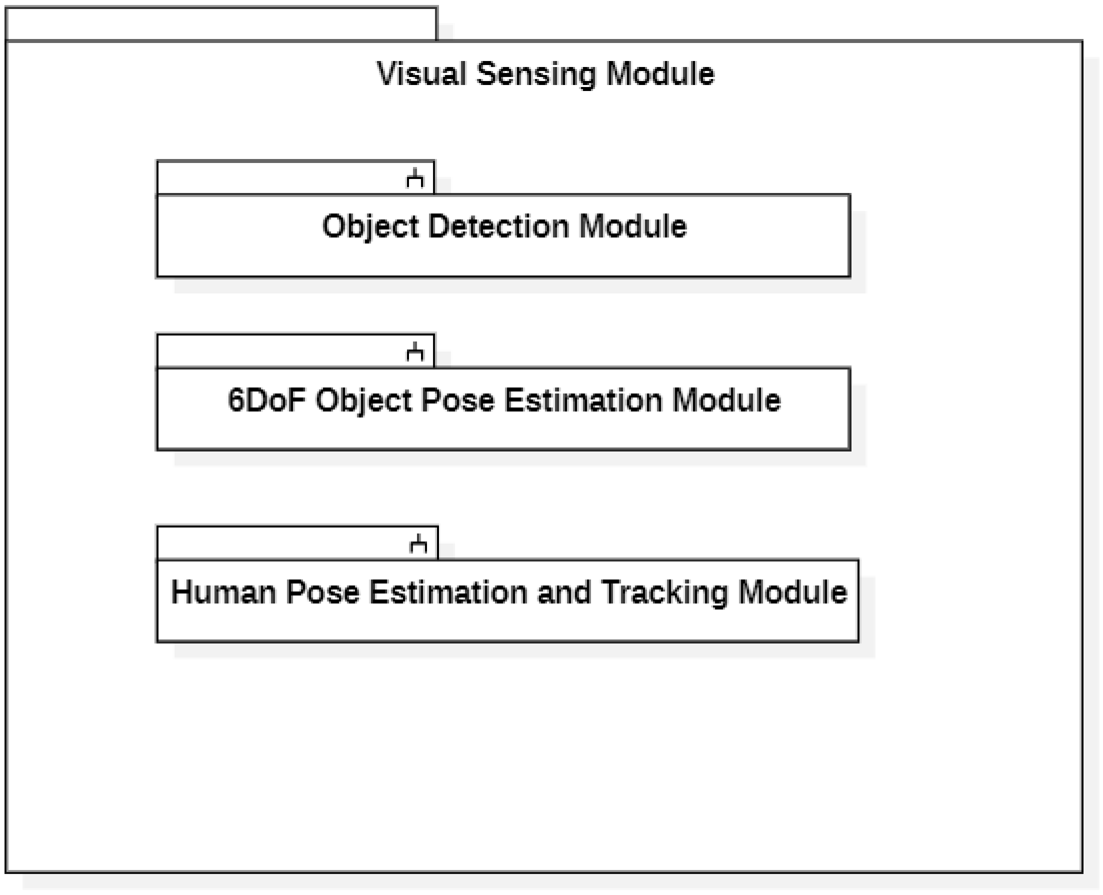
Description

The Visual Sensing Module utilizes the feedback from the optical sensors of the MANiBOT platform to enhance the robot perception of the working environment by applying to the acquired RGBD data machine-learning methods for real-time object detection and 6DoF pose estimation in cluttered scenes, while detecting and tracking the 3D pose of humans in the area near the robot.

Main Functionalities

The Visual Sensing Module is a perception module that allows the detection and identification of multiple unknown objects within an image, which enhances the MANiBOT platform's perception capabilities. This module precisely calculates each object's six degrees of freedom (6DoF) pose with respect to the camera's or robot's reference frame by using cutting-edge computer vision techniques. By providing the 3D pose of the human skeleton, including the positioning of joints and links, also within the camera's or robot's reference frame, the module provides thorough insights into human presence and interaction.

Internal Components


Visual Sensing Module
Object Detection Module
6DoF Object Pose Estimation Module
Human Pose Estimation and Tracking Module

The Visual Sensing Module consists of the following internal components:

Object Detection Module

- Object Detection Module utilizes deep neural networks that take as input pairs of RGB and Depth images and detects multiple instances of objects of interest in cluttered scenes (e.g. retail items on the racks of the supermarket).

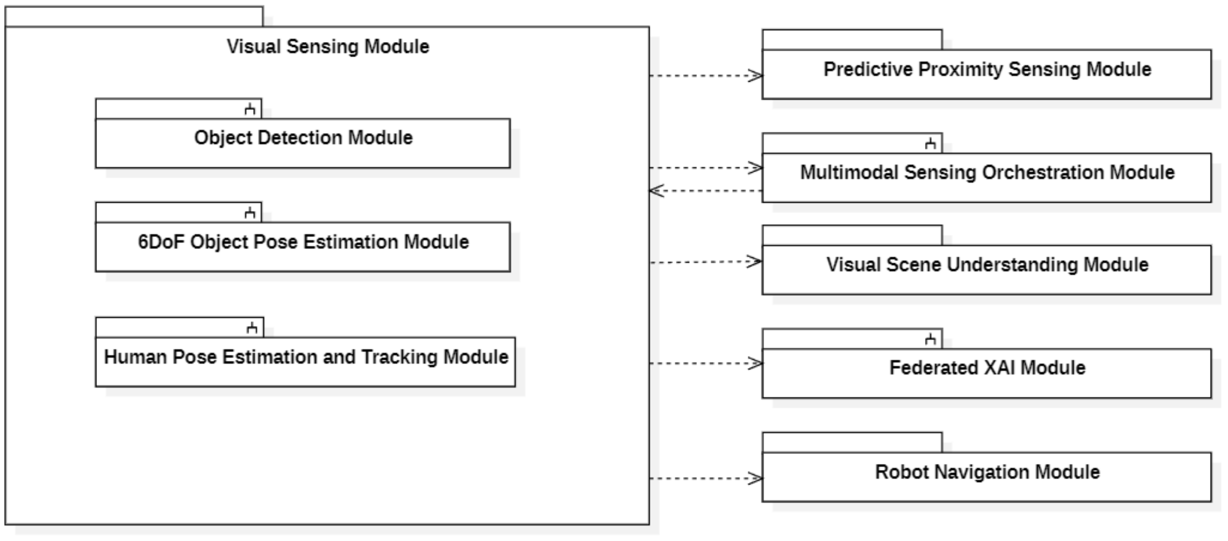
6DoF Object Pose Estimation Module

- 6DoF Object Pose Estimation Module is responsible for estimating the pose of the detected objects of interest in order to facilitate their manipulation by the robotic platform. The detected objects' pose will be defined using generic object models at category level.

Human Pose Estimation and Tracking Module

- Human Pose Estimation and Tracking Module will utilize RGBD information of the human workers near the robot in order to first detect and then track in real-time the pose of the workers in 3D space.

Dependencies to other components


Visual Sensing Module
Object Detection Module
6DoF Object Pose Estimation Module
Human Pose Estimation and Tracking Module
Predictive Proximity Sensing Module
Multimodal Sensing Orchestration Module
Visual Scene Understanding Module
Federated XAI Module
Robot Navigation Module

Visual Sensing Module interacts with the following components:

Predictive Proximity Sensing Module (To)

- This component expects input from the Visual Sensing Module in order to utilize the detected human pose for the human intention prediction.

Multimodal Sensing Orchestration Module (From/To)

- The interaction between the Visual Sensing Module and Multimodal Sensing Orchestration Module is essential in order to orchestrate the priorities between the different perception modalities.

Visual Scene Understanding Module (To)

- The Visual Sensing Module provides crucial information that contributes to the environment's semantic understanding. With the aid of this input, the Visual Scene Understanding Module is able to produce segmentation maps and evaluate the visual affordances of the objects it has detected, leading to a more thorough understanding of the objects' interactions and relationships with one another within the scene. Furthermore, the Visual Sensing Module's structural data facilitates the estimation of underlying structural relations between objects, which allows the Visual Scene Understanding Module to efficiently utilize machine learning techniques.

Federated XAI Module (To)

- The Federated XAI Module uses federated learning algorithms, which take input by detailed visual data generated by the Visual Sensing Module. By processing this visual data, the Federated XAI Module makes sure that users receive insights from the robot's visual perception in an understandable way, which promotes mutual trust and understanding.

Robot Navigation Module (To)

- The Robot Navigation Module receives vital spatial data from the Visual Sensing Module to assist in obstacle detection and avoidance. The navigation system can create an accurate representation of the surroundings utilizing this data. The Visual Sensing Module enables the Robot Navigation Module to adapt to dynamic changes in the environment, such as the presence of new objects or moving obstacles. This collaborative relationship ensures that the MANiBOT platform navigates safely and efficiently.

Addressed Requirements

Requirements ID	Description
SFR01	Recognize the product to be replenished
SFR02	Recognize expired and damaged products
SFR03	Recognize the replenishment cart
SFR04	Recognize dimensions and orientation of each product per box
SFR05	Recognize dimensions and orientation of boxes per pallet
SFR06	Recognize the type of box
SFR07	Recognize electronic shelf labels and extract content
SFR08	Recognize the remaining space on shelves
SFR09	Detect the appropriate baggage cart
SFR10	Detects moving baggage on the conveyor belt
SFR11	Recognize the front side (i.e., customer facing side) for each box
SFR12	Detects each baggage separately identifying its shape and dimensions
SFR13	Recognize how many boxes of a specific product item can fit on the shelf
SFR14	Recognize when a baggage cart is full or empty
SFR15	Recognize damaged luggage for the arrivals
SFR16	Recognize the luggage material (soft or hard)
SFR17	Ability to determine the optimal quantity of products for placement at the forefront of the shelf
SFR18	Recognize misplaced products on the shelves
SFR19	Recognize the free space at the target destination (baggage cart or belt) to place the luggage
SFR20	Performs quality check on products (flowers are not withered, fruits and vegetables are not deviated)
SFR21	Recognizes the location for the baggage placement
SFR22	Recognize the proximate area and shelves and compare the predefined assortment planogram with the actual placement of the ESLs
SFR23	Recognize boxes' arrangement/placement on pallet (top to bottom)
SFR24	Recognize product assortment on shelves according to ESLs and per product defined area (by assortment plan)
SFR25	Identify the correct destination on the shelf (through ESL) given the prior recognition of the mixed pallet and the respective product on the mixed pallet
SFR26	Inspect the state and replenishment feasibility of each package (per product)
SFR27	Recognize the mixed pallet
SFR028	Recognize and track humans trajectory
SFR029	Detects proximity of moving baggage tractor with loaded carts
SFR66	Robot pauses its current task if employee or customer is too close to it

5.2 Tactile Sensing Module

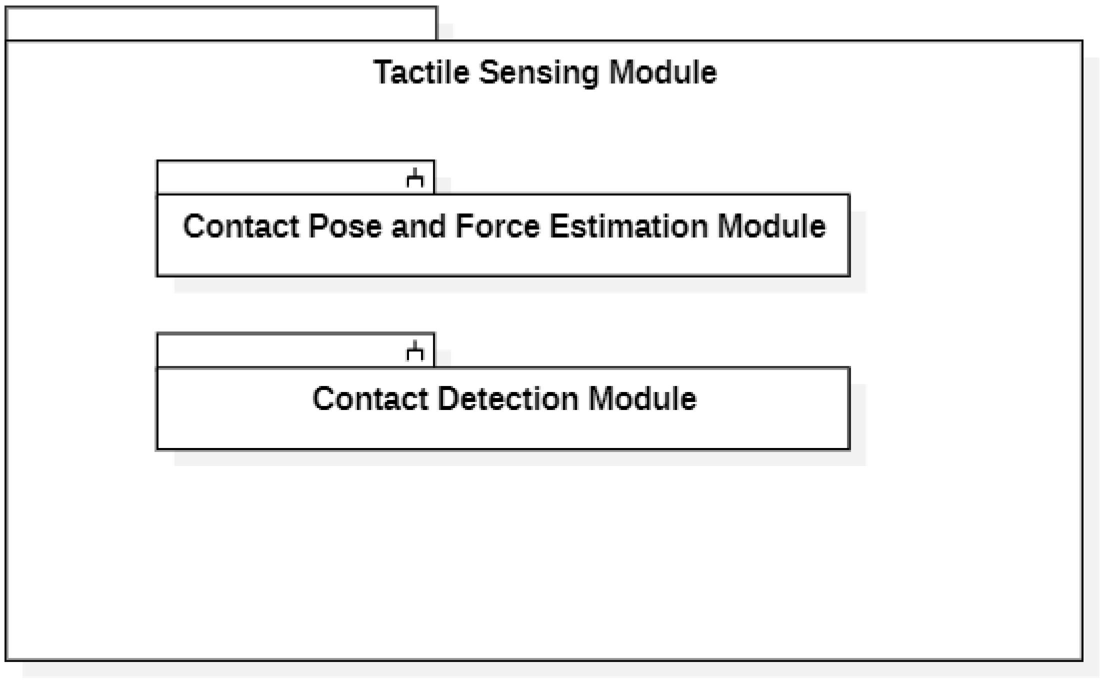
Description

The Tactile Sensing Module utilizes the feedback from the high-resolution optical tactile sensors of the MANiBOT platform to infer information on contact pose and force from supervised deep learning models, which may then be used to better inform grasping and manipulation protocol.

Main Functionalities

The Tactile Sensing module allows for contact force and pose to be inferred to a high accuracy. It can also be used for the inference of more fundamental states such as the presence of contact, or slip detection.

Internal Components



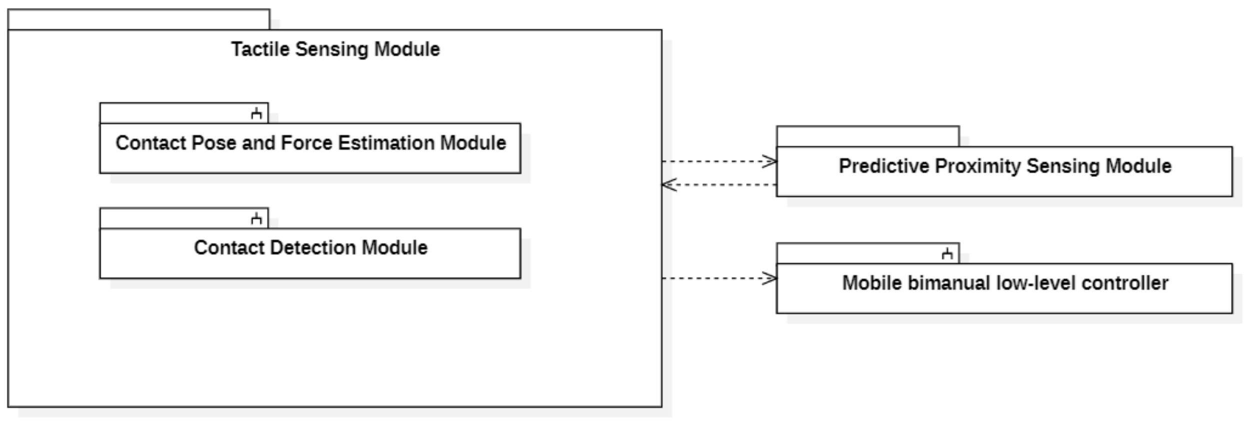
Contact Pose and Force Estimation Module

- This module takes an input of the current tactile image and infers contact pose and force from the output of a deep neural network.

Contact Detection Module

- Using an image similarity measure, this module compares the current tactile image to a base state tactile image to detect contact to high sensitivity and for relatively low computational cost.

Dependencies to other components


Tactile Sensing Module
Contact Pose and Force Estimation Module
Contact Detection Module
Predictive Proximity Sensing Module
Mobile bimanual low-level controller

Predictive Proximity Sensing Module (From/To)

- The tactile and proximity sensing modules are expected to work together to provide proprioceptive information specific to the grasping and handling of objects.

Mobile bimanual low-level controller (To)

- Since the tactile information will inform of any minor grasp adjustments which may need to be made as the object is handled, the controller responsible for moving the arm/gripper will require input from the tactile sensing module.

Addressed Requirements

Requirements ID	Description
SFR16	Recognize the luggage material (soft or hard)
SFR38	Grasp products from boxes
SFR39	Ability to pick and place single items
SFR40	Ability to pick up boxes with max weight up to 10 kg
SFR46	Manipulate and replenish product boxes
SFR47	Ability to close and open nested shelves, fridges, or freezing units
SFR48	Ability to grab newspapers and magazines enclosed in brown boxes and place them in storage positions
SFR49	Ability to open closed boxes and stock their content in the corresponding shelves
SFR50	Grasp the luggage that made from soft material gently
SFR51	Ability to prepare the products/SKUs (e.g., removing card box or plastic foil toppers)
SFR59	Ability to restock shelves with product/SKUs from the pallet (top to bottom)
SFR60	Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism
SFR62	Manipulate the baggage to detect the bagtag

5.3 Predictive Proximity Sensing Module

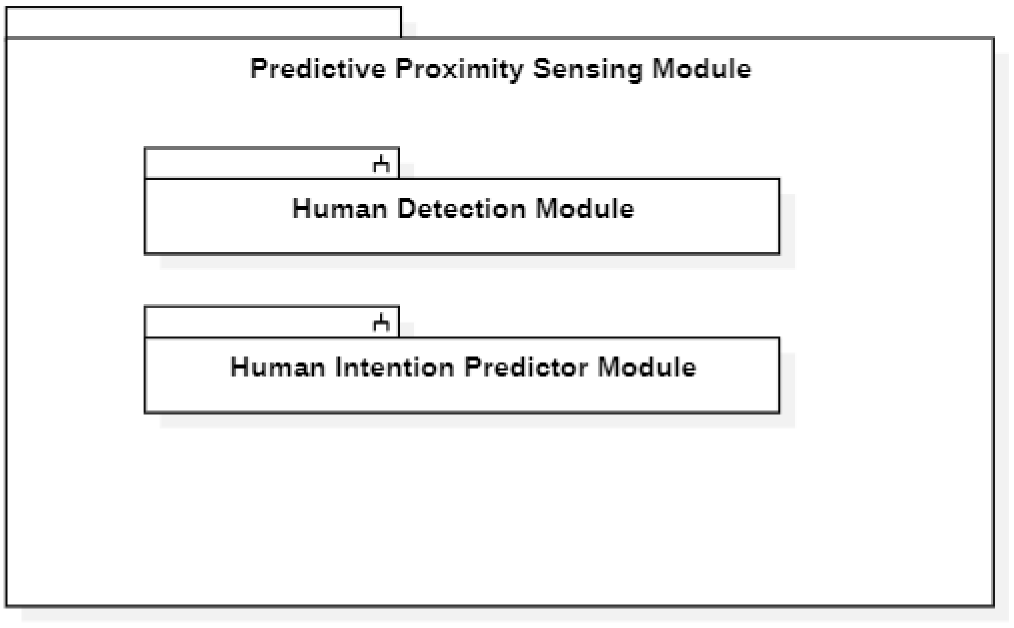
Description

The Predictive Proximity Sensing module of the MANiBOT platform uses data from proximity capacitive-based sensors (T6.2) to enhance robots' awareness about the surrounding environment. It allows, by analyzing the collected timeseries data and applying machine-learning techniques, to detect and track humans in real-time.

Main Functionalities

The module is responsible for detecting and tracking humans for safety reasons. In the first case, the module analyzes the signal retrieved from the sensor's array to classify it. In the second case, the module, powered by machine learning algorithms, performs dedicated mining of the signal and provides insights of humans' intentions.

Internal Components


Predictive Proximity Sensing Module
Human Detection Module
Human Intention Predictor Module

The Predictive Proximity Sensing Module consists of the following internal components:

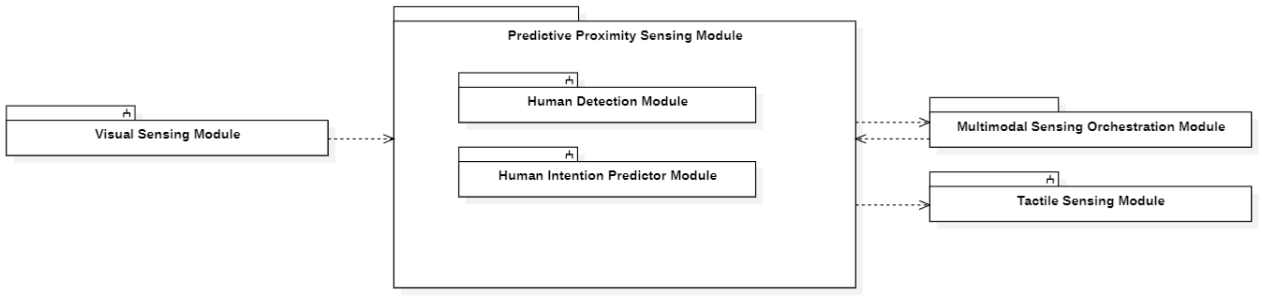
Human Detection Module

- The Human Detection Module utilizes deep neural networks that take as input multi-variate timeseries signal from the array of capacitive-based proximity sensors and classifies them as a human or no-human detection.

Human Intention Predictor Module

- The function of the Human Intention Predictor Module is to estimate the humans' intentions with the ultimate goal of enhancing collaboration between humans and robots.

Dependencies to other components:


Predictive Proximity Sensing Module
Human Detection Module
Human Intention Predictor Module
Visual Sensing Module
Multimodal Sensing Orchestration Module
Tactile Sensing Module

Visual Sensing Module interacts with the following components:

Visual Sensing Module (From)

- This component provides input to the Human Intention Predictor Module in order to utilize the detected human pose for empowering the human intentions prediction.

Multimodal Sensing Orchestration Module (From/To)

- The interaction between the Predictive Proximity Sensing Module and the Multimodal Sensing Orchestration Module is essential to orchestrate priorities between different perception modalities.

Tactile Sensing Module (To)

- The interaction between the Predictive Proximity Sensing Module and the Tactile Sensing Module could be useful to support, if combined, pre-shaping of manipulators and grippers before contact and tactile-based actions happen.

Addressed Requirements

Requirements ID	Description
SFR028	Recognize and track humans trajectory
SFR63	Ability to avoid collision with moving or static objects
SFR66	Robot pauses its current task if employee or customer is too close to it

5.4 Multimodal Sensing Orchestration Module

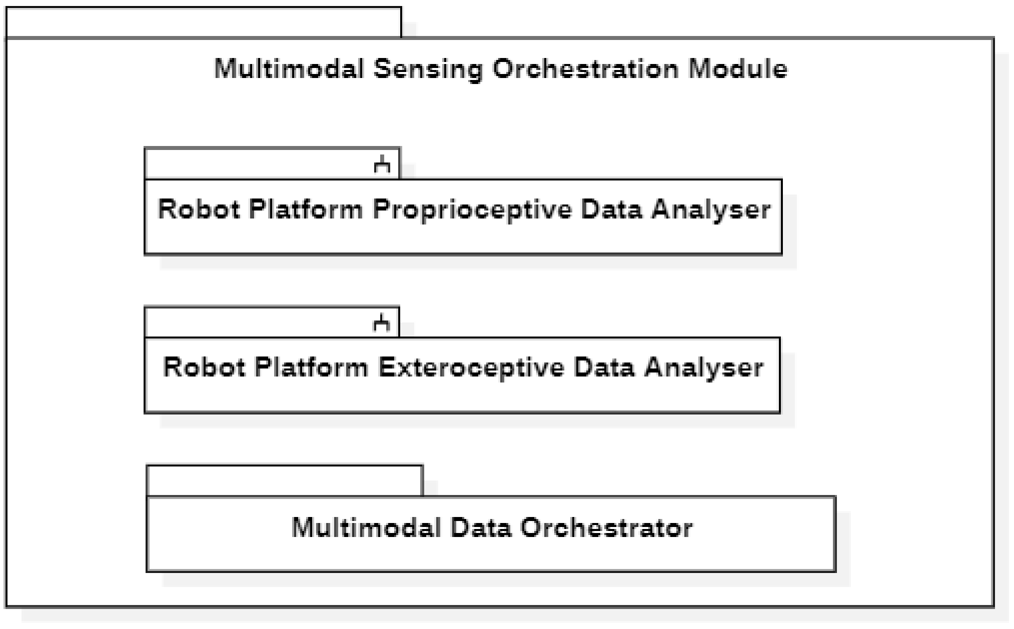
Description

The Multimodal Sensing Orchestration Module will rely on closed-loop real-time feedback from multiple sensors, including proprioceptive (such as IMU and force/torque sensors), and exteroceptive (like visual, tactile and proximity sensors) to continuously be aware of the object, task, environment and robot state. The data will be used to steer actions and affect behavior change by high-level adaptive sensing in the robot to meet perceptual targets. Combined, the importance and efficiency of each sensor modality will be dynamically weighted i.e., the most necessary ones get a priority in control loop which closely resembles to that of /as similar to person.

Main Functionalities

The module multiple sensors which specializes on different modalities such as visual, tactile, proximity, inertial, and force data for interactive closed-loop control in concurrency with its environment and area of focus. The overhead sensor's view is continuously adjusted based on the tasks that need to be performed. It maintains active surveillance of changing real time inputs and awareness of the task and objects to be manipulated. Synthesizing high-level commands involves the processing of most relevant sensor inputs in a stage-wise fashion from the use of vision toward touch of the object. Also, the system measures and suggests the internal forces and torques to be exercised towards safe and proper operation.

Internal Components


Multimodal Sensing Orchestration Module
Robot Platform Proprioceptive Data Analyser
Robot Platform Exteroceptive Data Analyser
Multimodal Data Orchestrator

The Multimodal Sensing Orchestration Module consists of the following internal components:

Robot Platform Proprioceptive Data Analyser

- Robot Platform Proprioceptive Data Analyser processes internal sensory data (e.g. IMU, force, torque) from the robot itself. It monitors the robot's joint positions, forces and torques to ensure stability, balance and safe operation. Provides feedback for controlling the robot's own movements and adjusting behavior based on internal conditions.

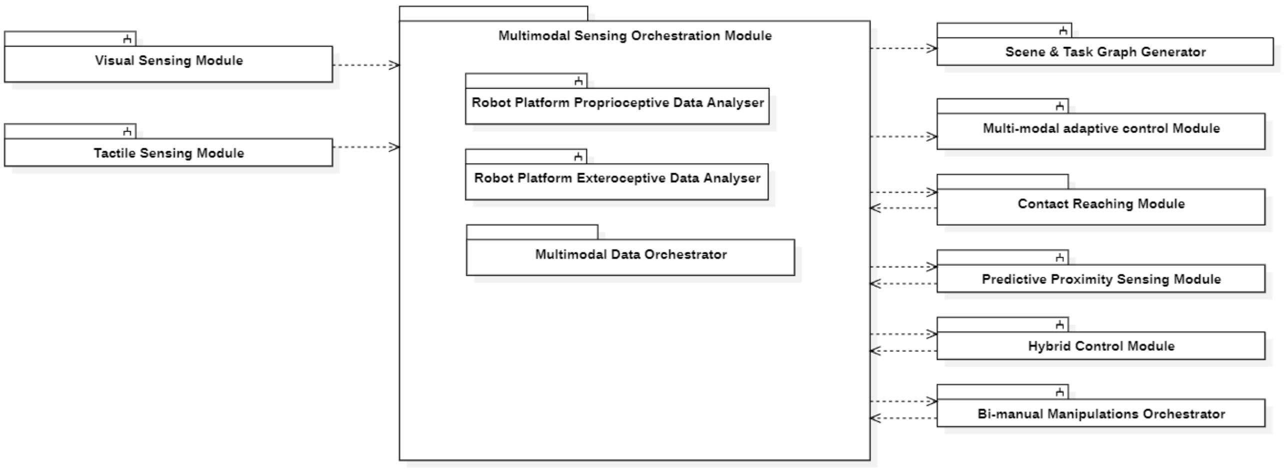
Robot Platform Exteroceptive Data Analyser

- Robot Platform Exteroceptive Data Analyser processes the external sensory data (e.g. visual, tactile, proximity) related to the robot's environment and objects. It ensures proper interaction with the environment, adjusting the robot's actions in response to real time environmental changes by tracking objects, measuring distances etc.

Multimodal Data Orchestrator

- The Multimodal Data Orchestrator integrates and prioritizes data from both proprioceptive and exteroceptive analyzers to guide the robot's decision-making process. IT dynamically assigns weights to different sensor inputs based on task relevance and context. It coordinates sensor data to optimize real-time control and action selection, ensuring smooth transitions and enabling adaptive behavior in complex tasks.

Dependencies to other components


Visual Sensing Module
Tactile Sensing Module
Multimodal Sensing Orchestration Module
Robot Platform Proprioceptive Data Analyser
Robot Platform Exteroceptive Data Analyser
Multimodal Data Orchestrator
Scene & Task Graph Generator
Multi-modal adaptive control Module
Contact Reaching Module
Predictive Proximity Sensing Module
Hybrid Control Module
Bi-manual Manipulations Orchestrator

Multimodal Sensing Orchestration Module interacts with the following components:

Visual Sensing Module (From)

- This module supplies the Multimodal Sensing Orchestration Module with the visual inputs. It ensures the system has the necessary visual information to detect and recognize objects and their features in the environment.

Multi-modal adaptive control Module (To)

- This component expects input from the Multimodal Sensing Orchestration Module to plan the manipulation scheme. It adapts the robot's actions dynamically based on the sensor inputs to optimize task execution.

Tactile Sensing Module (From)

- This component provides tactile inputs to the Multimodal Sensing Orchestration Module. It captures real-time touch and pressure data, ensuring precise handling during physical interactions with objects.

Predictive Proximity Sensing Module (From/To)

- This module will collaborate with Multimodal Sensing Orchestration Module to enable adaptable safety during robot operation. It allows the robot to anticipate potential collisions and adjust its movements accordingly.

Contact Reaching Module (From/To)

- The safe and efficient object manipulation is ensured with the cooperation of this component with the Multimodal Sensing Orchestration Module. It guarantees proper contact with objects while maintaining stability and precision during manipulation.

Hybrid Control Module (From/To)

- The Multimodal Sensing Orchestration Module and the **Hybrid Control Module** components are interacting in order to prioritize the tasks that need to be executed. This coordination ensures task sequences are handled efficiently, adapting to changing environmental conditions.

Scene & Task Graph Generator (To)

- This component expects input from the Multimodal Sensing Orchestration Module to determine how the manipulation will be achieved. It formulates the base-level actions required for mobile manipulation, ensuring smooth coordination of robot movements.

Bi-manual Manipulations Orchestrator (From/To)

- The collaboration between Bi-manual Manipulations Orchestrator Module and Multimodal Sensing Orchestration Module is crucial for ensuring safe and effective manipulation of target objects. It synchronizes the actions of both arms for complex bi-manual tasks.

Addressed Requirements

Requirements ID	Description
SFR16	Recognize the luggage material (soft or hard)
SFR028	Recognize and track humans trajectory
SFR38	Grasp products from boxes
SFR39	Ability to pick and place single items
SFR40	Ability to pick up boxes with max weight up to 10 kg
SFR46	Manipulate and replenish product boxes,
SFR38	Grasp products from boxes
SFR39	Ability to pick and place single items
SFR40	Ability to pick up boxes with max weight up to 10 kg
SFR46	Manipulate and replenish product boxes
SFR62	Manipulate the baggage to detect the bagtag
SFR66	Robot pauses its current task if employee or customer is too close to it

5.5 Visual Scene Understanding Module

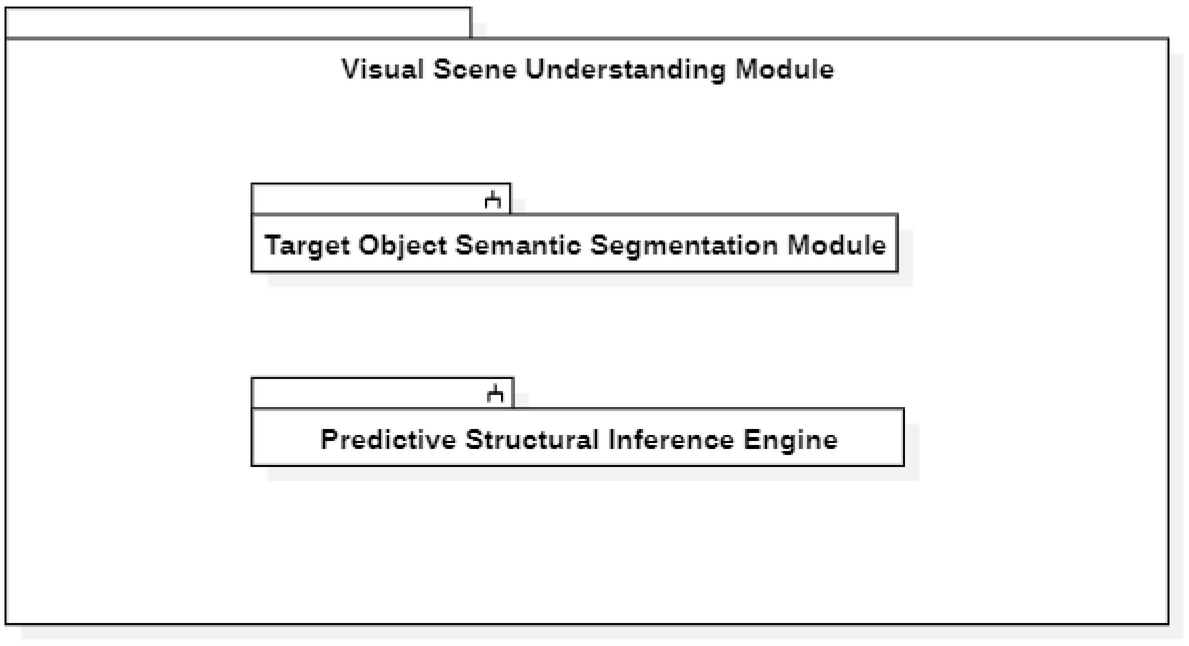
Description

The Visual Scene Understanding Module utilizes the feedback from the optical sensors of the MANiBOT platform to achieve a) understanding of semantic information of target objects including segmentation maps, visual affordances and b) estimation of the underlying structure and structural relations between objects in scenes, using machine learning techniques involving processing of visual data.

Main Functionalities

The module provides segmentation and visual affordance maps for various manipulation actions, and structure graphs containing the structural relations between objects which are viewed by the camera sensors. This graph contains connections between objects, representing the structural dependencies between them.

Internal Components


Visual Scene Understanding Module
Target Object Semantic Segmentation Module
Predictive Structural Inference Engine

The Visual Scene Understanding Module consists of the following internal components:

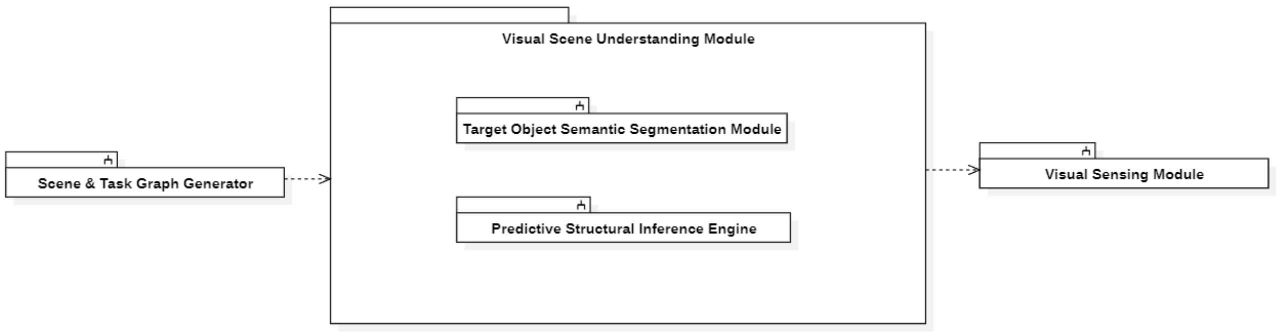
Target Object Semantic Segmentation Module

- Target Object Semantic Segmentation Module is in charge of estimating the affordance grounding of detected target objects and determining their semantic structure to optimize them for robotic manipulation.

Predictive Structural Inference Engine

- Predictive Structural Inference Engine Module is responsible for estimating the structural relations between the detected objects of interest in order to facilitate an understanding of the underlying scene structure and dynamics. The structural relation information will be represented as a directed graph structure.

Dependencies to other components


Scene & Task Graph Generator
Visual Scene Understanding Module
Target Object Semantic Segmentation Module
Predictive Structural Inference Engine
Visual Sensing Module

The Visual Scene Understanding Module interacts with the following components:

Scene & Task Graph Generator (From)

- The Visual Scene Understanding Module will provide **Scene & Task Graph Generator** with semantic, affordance and structural information of the scene, to be used for planning the manipulation actions.

Visual Sensing Module (To)

- The Visual Scene Understanding Module will receive detected object location and pose information from Visual Sensing Module to facilitate the affordance grounding and structure relation estimation of the given objects.

Addressed Requirements

Requirements ID	Description
SFR21	Recognizes the location for the baggage placement
SFR030	The robot decides which baggage to remove with least disturbance to surrounding baggage
SFR59	Ability to restock shelves with product/SKUs from the pallet (top to bottom)
SFR62	Manipulate the baggage to detect the bagtag
SFR65	Detects the optimal contact points in boxes, bags and products to grasp the item

5.6 Federated XAI Module

Description

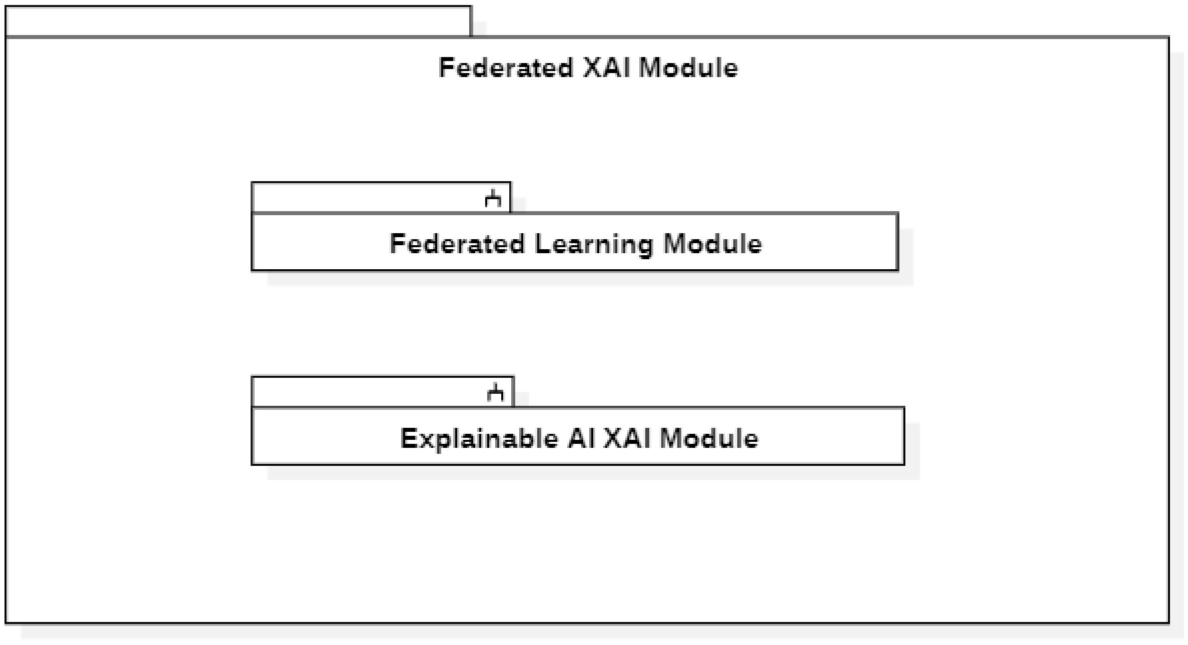
This module offers federated learning and Explainable AI capabilities to the MANiBOT platform and specifically to the perception modules in an efficient manner, while ensuring privacy, transparency, and trust for the end users.

Main Functionalities

Its main functionalities include:

- **Federated learning:** This allows training machine learning models across multiple decentralized entities, such as devices/robots or servers, without centralizing data, ensuring privacy.
- **Explainability:** This allows interpreting/explaining machine learning model inference results, enhancing their transparency, reliability, and trustworthiness.

Internal Components



The Federated XAI Module consists of the following internal components:

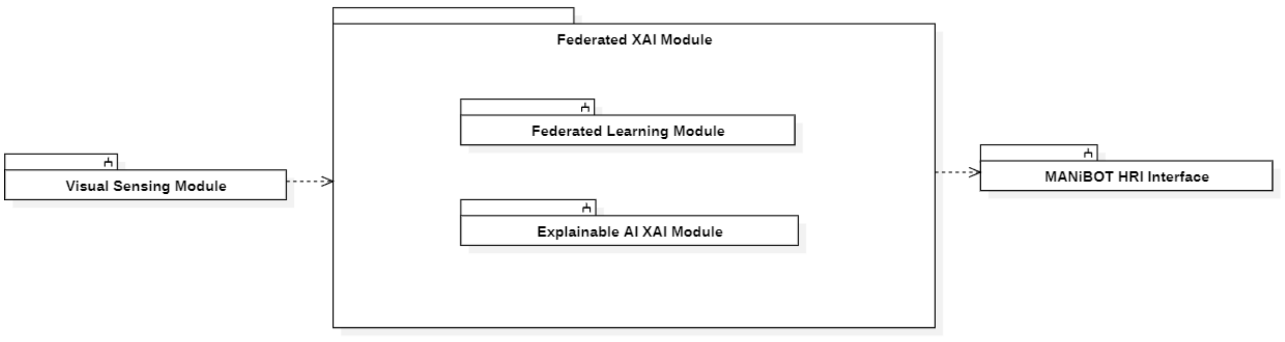
Federated Learning Module

- This module facilitates collaborative model training across distributed devices/robots while preserving data privacy. Its main functionalities include (i) distributed training by conducting model training locally on edge devices without the need for centralized data collection, (ii) data privacy preservation by keeping data on local devices and only sharing model updates, and (iii) model aggregation by combining locally trained models into a global model using established or novel aggregation techniques such as FedAvg and FedProx.

Explainable AI (XAI) Module

- This module aims to enhance transparency and interpretability in machine learning models. The module builds upon existing interpretability techniques by utilizing established methods such as SHAP, LIME and LRP to provide insights into model predictions and offers model explanation by generating human-understandable explanations for model decisions, enabling users to comprehend the rationale behind predictions.

Dependencies to other components



The Federated XAI Module interacts with the following components (WIP):

Visual Sensing Module (From)

- This component needs the deep learning models to be created and developed for the visual sensing module for:
    - Providing a federated mechanism for training of the perception modules.
    - Providing insights for how the perception models work through explainability.

MANiBOT HRI Interface (To)

- This component will provide the HRI interface the outcome of the explainable AI module.

Addressed Requirements

Requirements ID	Description
SFR91	Specific AI modules of the robot are updated while maintaining organisation and persons privacy
SFR92	The outputs of specific AI modules of the robot are explained through the UI (User Interface)

5.7 Mobile bimanual low-level controller

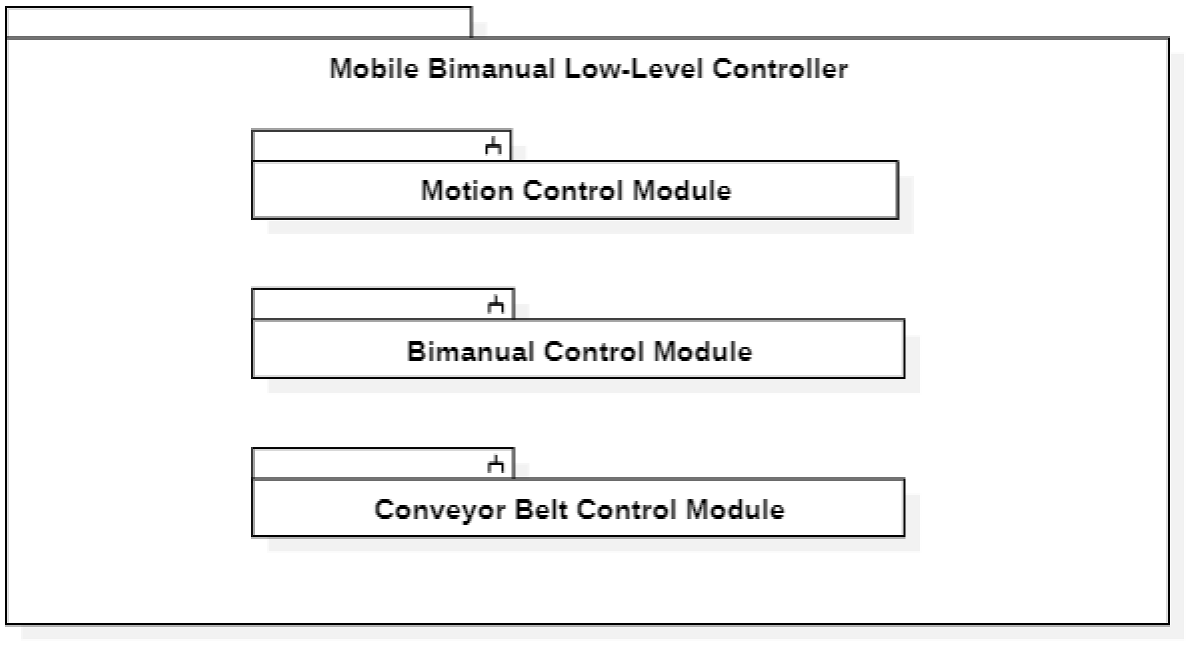
Description

The low-level controller for the mobile bimanual robot manages and coordinates the actions required to control the movement of the mobile platform, the robotic arms and the conveyor belt. It translates high-level instructions into precise commands for the actuators, ensuring stability, accuracy, and synchronization in manipulation and movement tasks. This controller is designed to respond to sensor signals in real-time, dynamically adjusting the robot's movements based on environmental conditions and specific tasks.

Main Functionalities

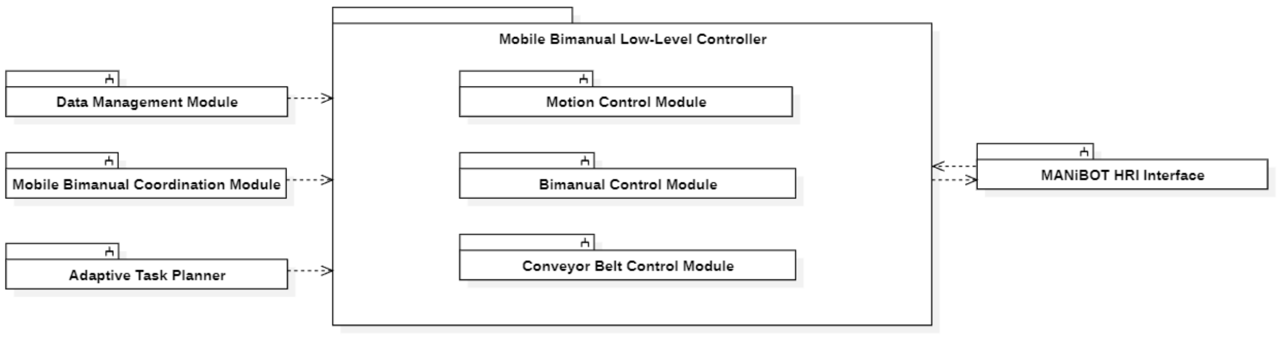
- **Mobile Platform Movement Control:** At a low level, the platform is controlled through longitudinal and angular velocity commands, and it will also provide feedback on the AGV's current state, including actual longitudinal and angular velocities.
- **Bimanual Coordination:** Synchronizes the movements of both robotic arms and linear actuators for complex manipulation tasks, such as grasping and moving objects of different sizes and weights. The arms receive and returns joint coordinates as feedback on the robot's overall state. There will be two separate interfaces: one for Robot 1 and another for Robot 2, each handling its respective arm.
- **Conveyor Belt Control on the Platform:** Controls the conveyor belt system via the position of two actuators. It will also provide status information, including potential errors or operational states, ensuring smooth coordination between the actuators and the belt's functionality.

Internal Components


Mobile Bimanual Low-Level Controller
Motion Control Module
Bimanual Control Module
Conveyor Belt Control Module

- **Motion Control Module:** Responsible of motor control to ensure a smooth speed tracking of the mobile platform.
- **Bimanual Control Module:** Manages the coordination of both robotic arms and linear actuators, ensuring precise synchronization for joint manipulation tasks as executed by the higher level control modules.
- **Conveyor Belt Control Module:** Controls the two actuators responsible for the position, and movement of the conveyor belt on the platform. This subsystem monitors the belt's operation, providing feedback on its position, speed, and any error that may occur during operation.

Dependencies to other components



- **Data Management Module (From):** The controller depends on data provided by proximity, tactile, and vision sensors to adjust movements and ensure safe manipulation.

- **Mobile Bimanual Coordination Module (From):** Receives global task instructions and translates them into low-level commands. The high-level system handles overall planning, while the low-level controller executes specific actions.
- **Adaptive Task Planner (From)**: Relies on the battery and charging management system to ensure the robot has enough power to complete its tasks without interruptions.
- **MANiBOT HRI Interface (To/From):** Although not directly related, the controller interacts with interfaces that allow operators to monitor and adjust the robot's actions in real-time.

Addressed Requirements

Requirements ID	Description
SFR64	Ability to avoid collisions with itself in bimanual tasks
SFR66	Robot pauses its current task if employee or customer is too close to it
SFR67	Robots pauses and resumes tasks according to user's command

5.8 Robot Navigation Module

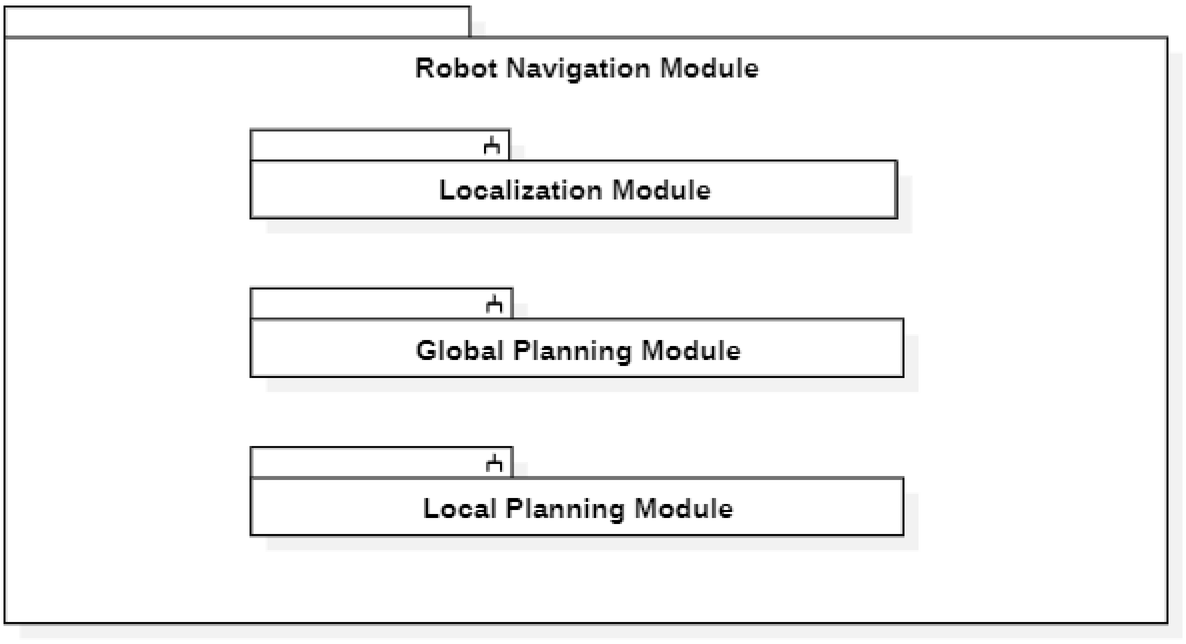
Description

The Robot Navigation Module creates the MANiBOT robot's navigation framework and provides a safe human aware, obstacle-free navigation.

Main Functionalities

This module's primary function is to generate a human-aware, obstacle-free path that should be respectfully followed. The global trajectories that are generated encounter constant updates to prevent any potential freezing issues with the robot. The module creates obstacle-free local trajectories that enable the robot to follow a predetermined global path.

Internal Components



The Robot Navigation Module consists of the following internal components:

Localization Module

- The module provides centimeter-level accuracy and robustness in dynamic environments by combining landmark-based localization with VSLAM. The purpose of this module is to give the platform a precise localization system so that the robot can navigate safely.

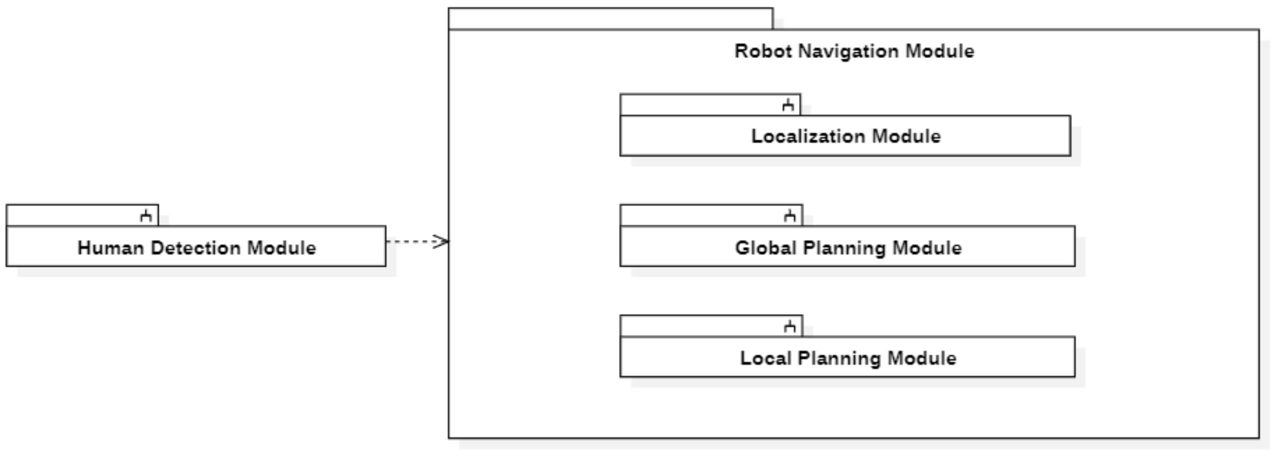
Global Planning Module

- The module provides a path followed by the platform and infers trajectories that consider human comfort and that the robot should kindly follow. The route that the global planner designs takes into account the possibility of human existence, fixed physical obstacles and possible restricted areas in the environments. When no feasible path exists, the global planner also reroutes.

Local Planning Module

The local planner provides local paths to the platform that follow the global path created by the global planner. The local planner module avoids static and dynamic obstacles by using LiDAR data, dynamic costmaps with varying resolutions and real-time traversability estimation. It also takes into account each infrastructure's limitations and regulations.

Dependencies to other components



Robot Navigation Module interacts with the following components:

Human Detection Module (From)

- The Human Detection Module utilizes RGBD information to detect and estimate the pose and the possible trajectories from the humans. The Robot Navigation Module uses the human poses estimated by this module to infer global and local human-aware trajectories.

Addressed Requirements

Requirements ID	Description
SFR031	Platform 2D/3D localization
SFR032	Platform Path planning ability
SFR033	Platform human-aware navigation
SFR034	Ability to move on the pavement area alongside the conveyor belt
SFR035	Ability to step up to the pavement area (height 15cm) around the conveyor belt (with or without the use of a ramp)
SFR036	Navigate avoiding restricted areas
SFR037	The robot selects a desired position regarding the baggage cart

5.9 Multi-modal adaptive control Module

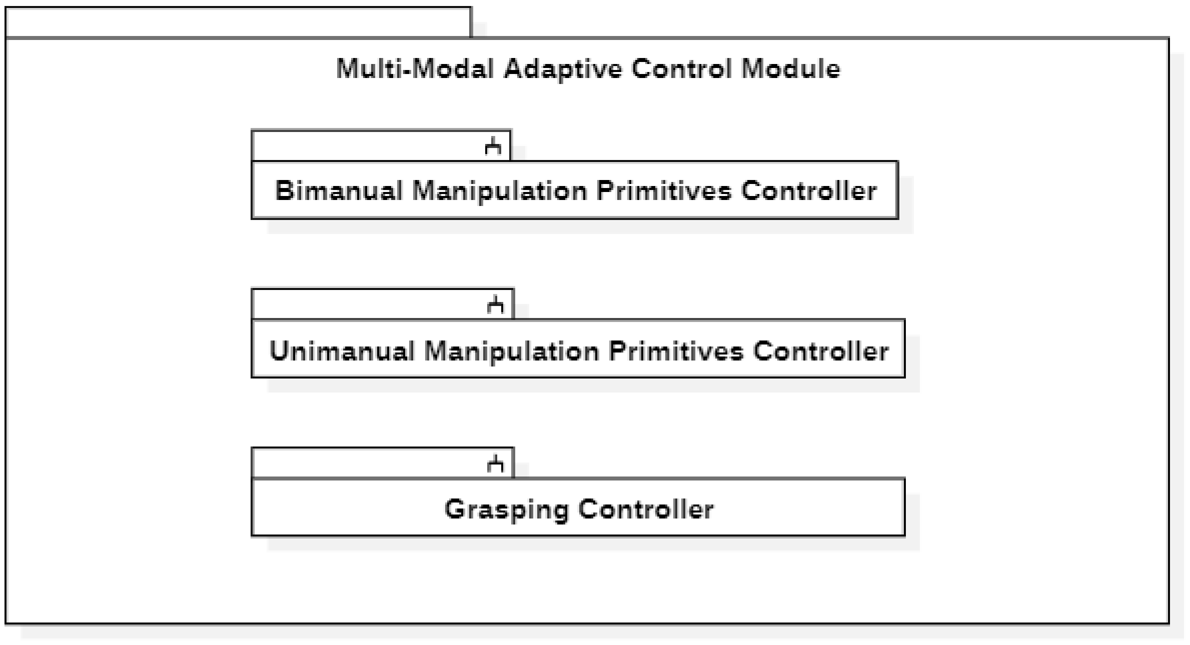
Description

The Multi-modal Adaptive Control Module forms the core of the MANiBOT's control architecture, enabling sophisticated manipulation capabilities. This module adapts to various environmental conditions and task requirements by employing advanced control strategies that integrate sensory feedback to optimize manipulation and grasping actions.

Main Functionalities

The module primarily enhances the robot's ability to perform complex manipulation tasks through adaptive control mechanisms. It supports dynamic adaptation of manipulation strategies based on real-time sensory data, ensuring high precision and reliability.

Internal Components


Multi-Modal Adaptive Control Module
Bimanual Manipulation Primitives Controller
Unimanual Manipulation Primitives Controller
Grasping Controller

The Multi-modal Adaptive Control Module consists of the following internal components:

Bimanual Manipulation Primitives Controller

- The Bimanual Manipulation Primitives Controller is responsible for the generation and execution of manipulation tasks requiring both arms, e.g., the manipulation of luggage or large containers in the supermarket. It is related to both T4.4 and T4.5.

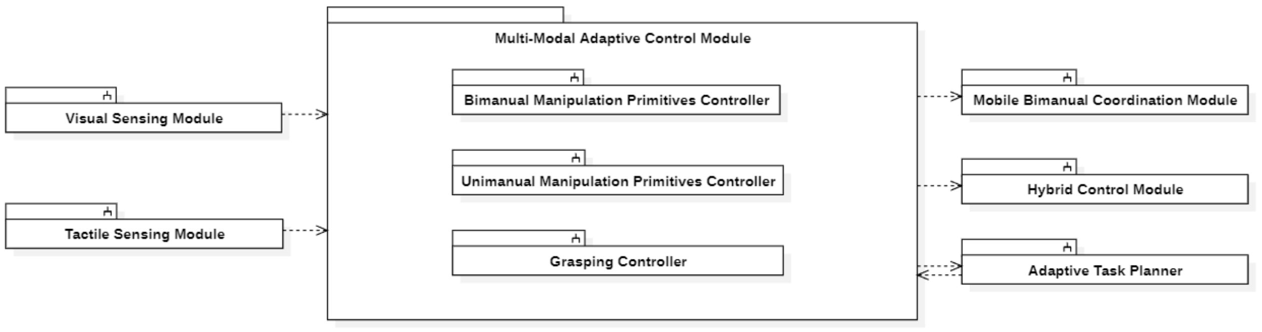
Unimanual Manipulation Primitives Controller

- The Unimanual Manipulation Primitives Controller is responsible for the generation and execution of manipulation tasks requiring just one arm or the synchronous movement of the two arms. Such tasks include the manipulation of small items in the supermarkets. It is related to both T4.4 and T4.5.

Grasping Controller

- The Grasping Controller is responsible for the execution of the grasping primitive. It executes bot unimanual grasps of small objects as well as bimanual grasps of large objects. It also responsible for the grasping of luggage handles.

Dependencies to other components


Visual Sensing Module
Tactile Sensing Module
Multi-Modal Adaptive Control Module
Bimanual Manipulation Primitives Controller
Unimanual Manipulation Primitives Controller
Grasping Controller
Mobile Bimanual Coordination Module
Hybrid Control Module
Adaptive Task Planner

Visual Sensing Module (From)

The 6-dof pose of the objects is required to execute all primitives. For the grasping controller, information about grasping affordances and luggage handle location is necessary.

Tactile Sensing Module (From)

- Information about the contact normal vector and force is required for the successful execution of the manipulation primitives

Mobile Bimanual Coordination Module (To)

- The generated control signals will be passed to the Mobile Bimanual Coordination Module, to execute those trajectories respecting the workspace constraints utilizing the redundant degrees of freedom.

Hybrid Control Module (To)

- The Hybrid Control Module receives the signal generated by this module to successfully guarantee the smooth transition between sequential primitive motions.

Adaptive Task Planner (From / To)

- The Adaptive Task Planner will command the beginning of each primitive motion, to achieve the manipulation task. It will also need to receive a signal of each low-level primitive's successful completion.

Addressed Requirements

Requirements ID	Description
SFR038	Grasp products from boxes
SFR039	Ability to pick and place single items
SFR040	Ability to pick up boxes with max weight up to 10 kg
SFR041	Ability to pick up boxes without damaging them
SFR042	Place the products in the correct positions
SFR043	Place the products that expire earlier in the front of the shelf ahead of newly received items
SFR044	Place the products in a specified orientation, ensuring the front side of the product faces the customer
SFR045	Place momentarily the boxes on its equipped table or another interim storage position
SFR046	Manipulate and replenish product boxes
SFR050	Grasp the luggage that made from soft material gently
SFR051	Ability to prepare the products/SKUs (e.g., removing cardbox or plastic foil toppers)
SFR052	Place misplaced products in an interim storage position or to the cart or to the right shelf position
SFR053	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position
SFR054	Place the products in the right position without using a planogram or consult it when it is available
SFR055	Place the luggage made of hard material below the luggage made of soft material
SFR056	Place the luggage with a distance between them on the reclaim belt for the arrivals
SFR057	Place the luggage within the boundaries of the belt/cart
SFR058	Handle luggage only for the flight that has been allocated to
SFR059	Ability to restock shelves with product/SKUs from the pallet (top to bottom)

SFR060	Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism
SFR061	Place empty (cardboard) boxes from replenished items into a bin cart located next to the pallet
SFR062	Manipulate the baggage to detect the bagtag

5.10 Mobile Bimanual Coordination Module

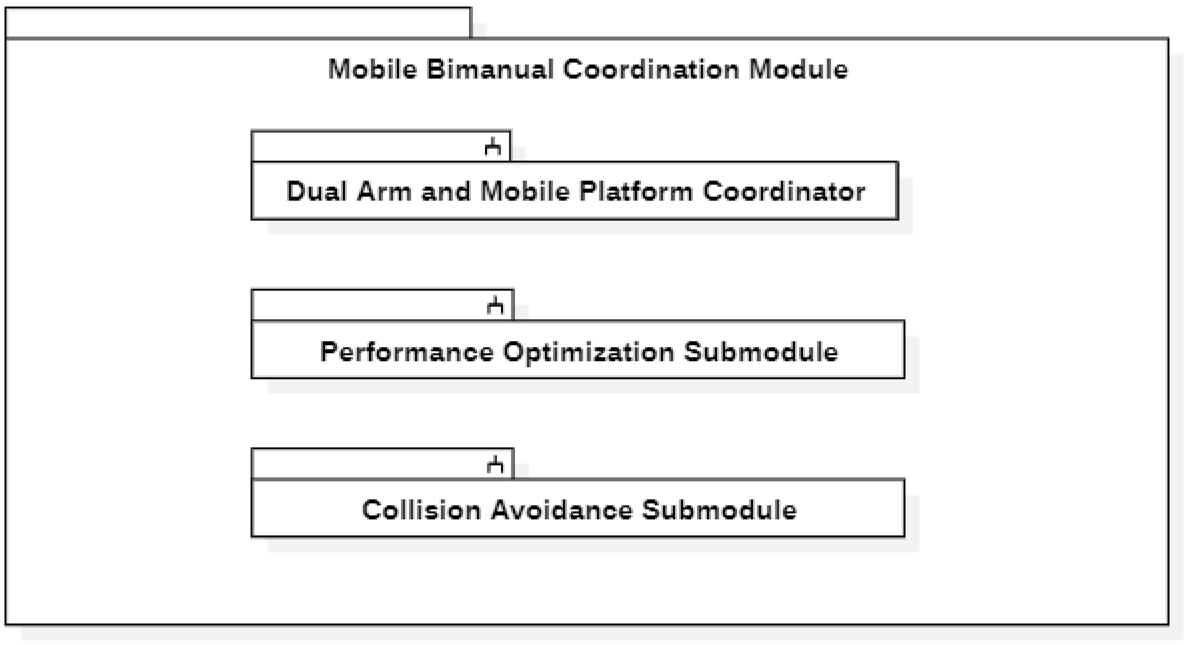
Description

The Mobile Bimanual Coordination Module is responsible for the coordination of the motion of the two arms and the mobile platform, and the exploitation of the redundancy of the robotic system to improve performance in bimanual tasks and avoid collisions with itself and the environment while also satisfying the main end-effector task objectives.

Main Functionalities

The Mobile Bimanual Motion Coordination Module functionalities include the coordination of the two arms and the mobile platform in bimanual tasks, the optimization of the dexterity of the end-effectors with respect to the task, the self and external obstacle collision avoidance.

Internal Components


Mobile Bimanual Coordination Module
Dual Arm and Mobile Platform Coordinator
Performance Optimization Submodule
Collision Avoidance Submodule

The Mobile Bimanual Motion Coordination Module consists of the following internal components:

Dual Arm and Mobile Platform Coordinator

- Dual arm and mobile platform coordinator implement control procedures that coordinate the motion of the two arms and the mobile platform during the end-effector tasks objectives execution (i.e., reaching, pushing, manipulation) to accurately perform the given task.

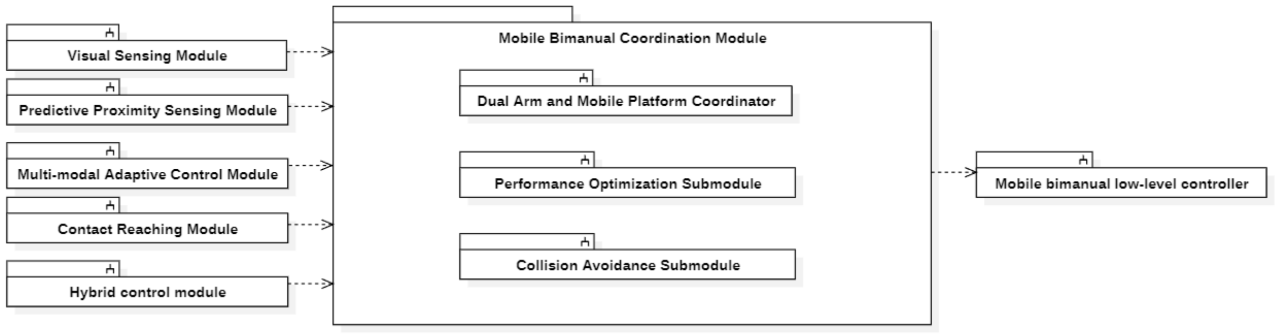
Performance Optimization Submodule

- The performance optimization submodule optimizes the configuration of the two arms and the mobile platform to improve the dexterity of the robotic system in terms of task-related optimization of performance metrics.

Collision Avoidance Submodule

- The collision avoidance submodule is responsible to guarantee that the robot performs the reference task without colliding with itself and with external obstacles utilizing proprioception, visual and tactile feedback available to the robotic system.

Dependencies to other components


Visual Sensing Module
Predictive Proximity Sensing Module
Multi-modal Adaptive Control Module
Contact Reaching Module
Hybrid control module
Mobile Bimanual Coordination Module
Dual Arm and Mobile Platform Coordinator
Performance Optimization Submodule
Collision Avoidance Submodule
Mobile bimanual low-level controller

Visual Sensing Module (From)

- The Mobile Bimanual Motion Coordination Module expects input from the Visual Sensing Module in the form of primitive shapes in order to utilize them for collision avoidance purposes.

Predictive Proximity Sensing Module (From)

- The Mobile Bimanual Motion Coordination Module expects input from the Predictive Proximity Sensing Module in order to utilize the proximity sensors for collision.

Multi-modal Adaptive Control Module (From)

- The Mobile Bimanual Motion Coordination Module expects input from the Multi-modal Adaptive Control Module in order to realize the desired motion to perform object manipulation while coordinating the different robot embodiments and satisfying the safety and performance constraints.

Contact Reaching Module (From)

- The Mobile Bimanual Motion Coordination Module expects input from the Contact Reaching Module in order to reach and establish contact with the object of interest while coordinating the different robot embodiments and satisfying the safety and performance constraints.

Hybrid control module (From)

- The Mobile Bimanual Motion Coordination Module expects input from the Hybrid control module in order to realize the desired trajectory to perform the desired behavior while coordinating the different robot embodiments and satisfying the safety and performance constraints.

Mobile bimanual low-level controller (To)

- The Mobile Bimanual Motion Coordination Module outputs the reference joint and mobile platform velocities that is received by this component which is then commanded to the robotic system.

Addressed Requirements

Requirements ID	Description
SFR063	Ability to avoid collision with moving or static objects
SFR064	Ability to avoid collisions with itself in bimanual tasks

5.11 Contact Reaching Module

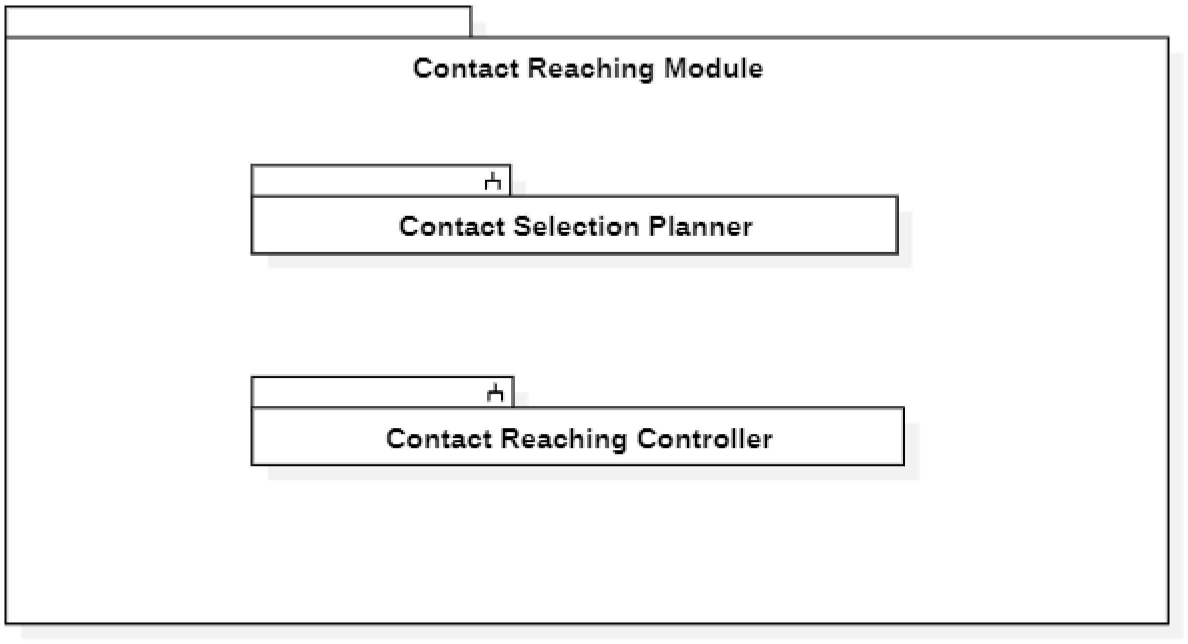
Description

To initiate the execution of each manipulation primitive, the robot must initially select the appropriate contact points with the object and then establish contact. This module, developed by AUTH, is responsible for both the selection of the desired contact points depending on the object, it's affordances and the selected primitive, as well as for the safe reaching of those contact points being safe for itself, its environment as well as humans in its workspace.

Main Functionalities

The module is instrumental in selecting the optimal contact points on the target based on task requirements and physical properties of the target. It ensures the reaching of those contact points while being compliant in unwanted external contact, while maintaining accurate trajectory tracking. It ensures the contact establishment with a desired force, to begin the selected manipulation primitive execution.

Internal Components



The Contact Reaching Module consists of the following internal components:

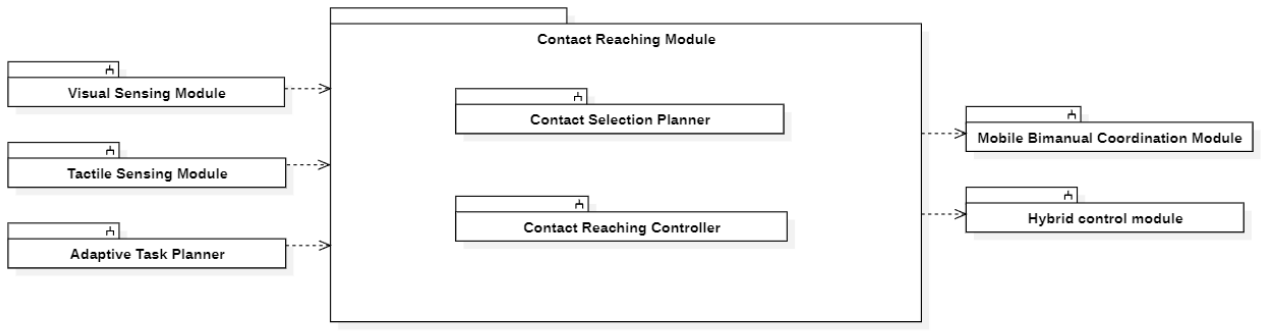
Contact Selection Planner

- The Contact Selection Planner is responsible for the selection of appropriate contact points for task execution. These can be multiple contact points on the same object for bimanual tasks or single contact points for unimanual tasks. Those contact points can be related to non-prehensile manipulation primitives, or grasping tasks. It is related to T4.3.

Contact Reaching Controller

- The Contact Reaching Controller is responsible for the real-time trajectory generation for the reaching of the contact points by the bimanual robot. It will leverage sensorial feedback to guarantee safe trajectory planning, while utilizing compliant control to ensure safety even in case of undesired environment contact. Nevertheless, the trajectory tracking accuracy will not be affected. It is related to T4.3.

Dependencies to other components



Visual Sensing Module (From)

- Object information is required to select the desired contact points.

Tactile Sensing Module (From)

- Tactile feedback can be used to generate compliant behaviors by position controlled robotic arms.

Adaptive Task Planner (From)

- The Adaptive Task Planner will provide information about the selected primitive, required for the contact point selection.

Mobile Bimanual Coordination Module (To)

- The generated control signals will be passed to the Mobile Bimanual Coordination Module, to execute those trajectories respecting the workspace constraints utilizing the redundant degrees of freedom.

Hybrid Control Module (To)

- The Hybrid Control Module receives the signal generated by this module to successfully guarantee the smooth transition between reaching the contact points and the selected primitive motions.

Addressed Requirements

Requirements ID	Description
SFR065	Detects the optimal contact points in boxes, bags and products to grasp the item

5.12 Hybrid Control Module

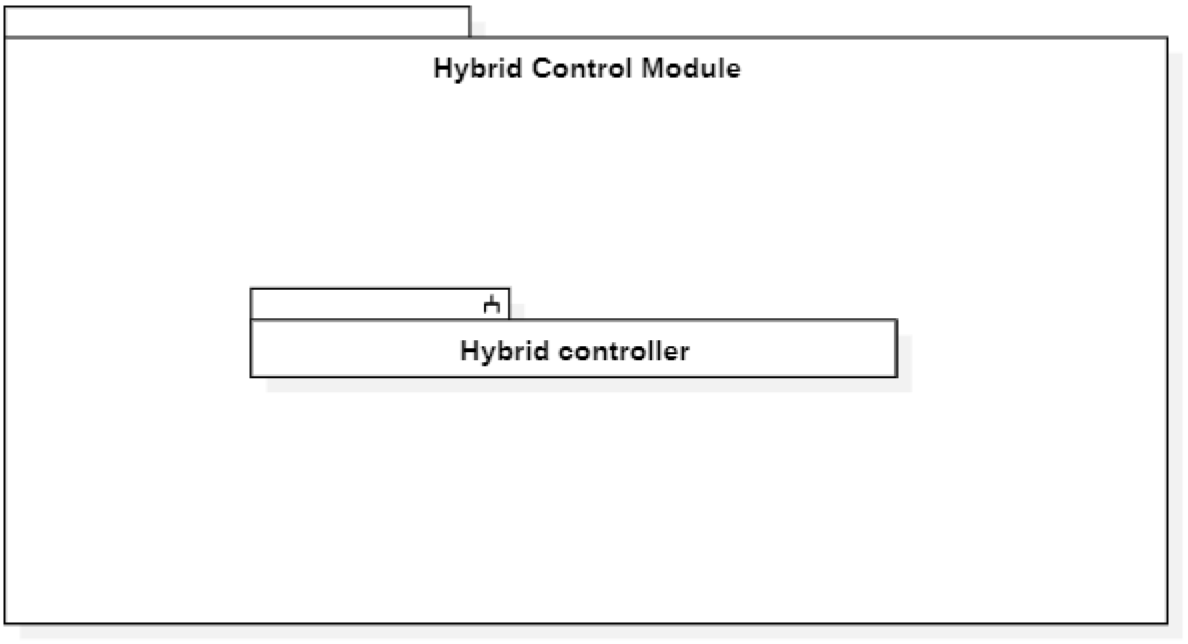
Description

The selection of manipulation primitives should be executed in timely and coordinated control manner, providing a stable, smooth and safe transition between two any sequential primitives. The learned manipulation primitive sequence should be executed with guaranteed stability and with additional attributes regarding the smoothness of the produced trajectories during the transition from different control modules responsible for different manipulation primitives. Furthermore, model uncertainties and external disturbances may seriously affect the accuracy of the performed manipulation leading to errors during operation. In this direction, the developed control module should supervise the performance during the whole operation to effectively handle cases with increased errors.

Main Functionalities

The main functionalities of this module are to provide stable and safe transition between different control modules incorporated to undertake the control of two sequential manipulation primitives. Additionally, this hybrid control module will supervise the operation of each primitive task to provide the whole operation with error-handling functionalities, if necessary.

Internal Components

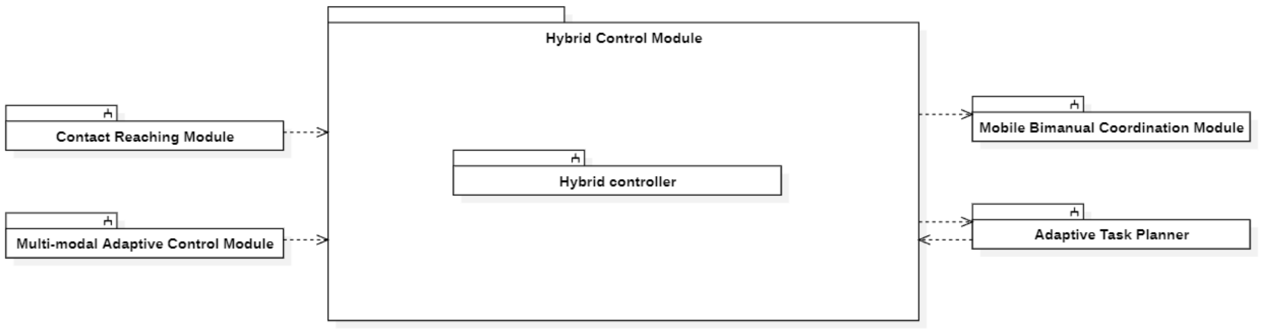


The Hybrid Control Module consists of one component:

Hybrid controller

- The Hybrid Controller is responsible for all the functionalities of this module.

Dependencies to other components



Multi-modal Adaptive Control Module (From)

- The multi-modal adaptive control modules should provide the control commands to the hybrid module in order to safely enforce a stable and smooth transition between the sequential primitives.

Mobile Bimanual Coordination Module (To)

- The generated control signals will be passed to the Mobile Bimanual Coordination Module, to execute those trajectories respecting the workspace constraints utilizing the redundant degrees of freedom.

Contact Reaching Module (From)

- The hybrid control module requires information regarding the successful contact establishment by the contact reaching module to enable proper controller initialization for the given task execution.

Adaptive Task Planner (From / To)

- This module requires information from the adaptive task planner to appropriately adjust its parameterization with respect to the specific needs of the given sequential primitives. Further, it should provide information to the task planner regarding the successful transition between the manipulation primitives.

Addressed Requirements

Requirements ID	Description
SFR38	Grasp products from boxes
SFR44	Place the products in a specified orientation, ensuring the front side of the product faces the customer
SFR46	Manipulate and replenish product boxes
SFR50	Grasp the luggage that made from soft material gently
SFR52	Place misplaced products in an interim storage position or to the cart or to the right shelf position
SFR53	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position
SFR62	Manipulate the baggage to detect the bagtag

5.13 Adaptive Modelling Module

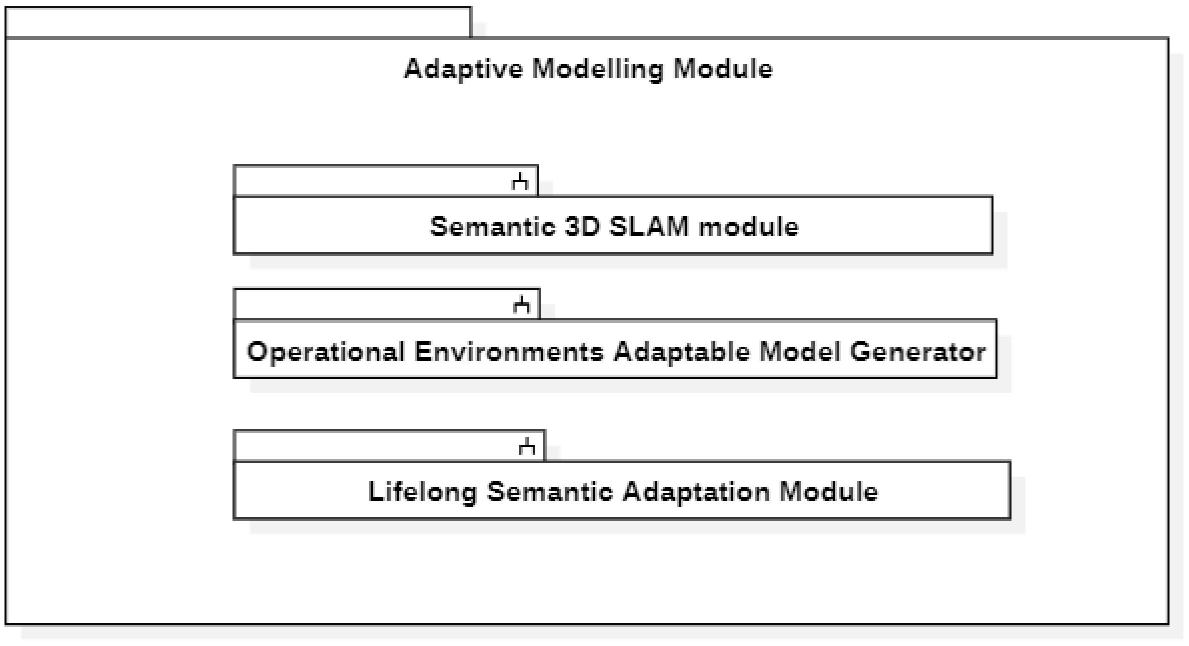
Description

The Adaptive Modelling Module utilizes the semantic information extracted from the Visual Scene Understanding Module, from the Visual Sensing Module, and from the optical sensors of the MANiBOT platform to create and update the operational environment models and semantic maps. Semantic information regarding the operational environments of the MANiBOT platform are stored adaptive models, ensuring lifelong semantic mapping.

Main Functionalities

The module facilitates the creation ontology-based models for the operational environments of the MANiBOT robot. The detected objects, affordances, actors, supporting surfaces and their relations are stored and updated, enabling the robot to act proactively in human-like context aware manner.

Internal Components


Adaptive Modelling Module
Semantic 3D SLAM module
Operational Environments Adaptable Model Generator
Lifelong Semantic Adaptation Module

The Adaptive Modelling Module consists of the following internal components:

Semantic 3D SLAM module

- Semantic 3D SLAM module utilizes the object instances and semantic information extracted from the Visual Scene Understanding to create 3D semantic maps of the operational environments of the MANiBOT platform.

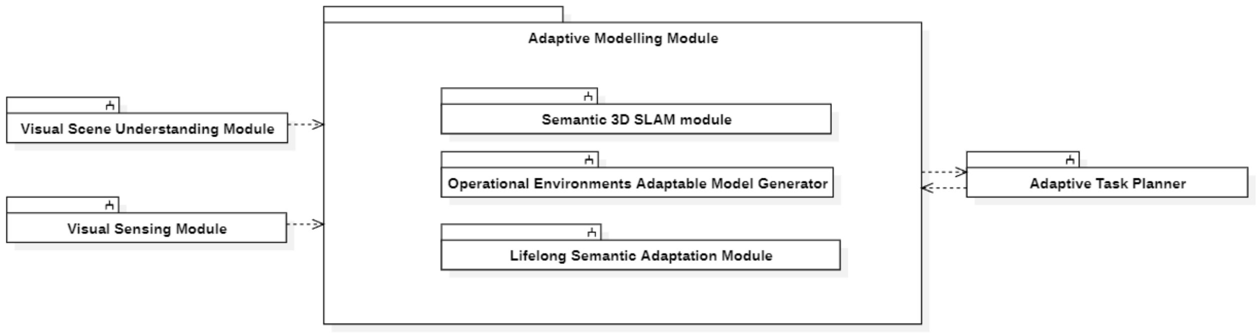
Operational Environments Adaptable Model Generator

- Operational Environments Adaptable Model Generator is responsible for creating adaptable semantic graphs/ontologies for modelling the operational environments.

Lifelong Semantic Adaptation Module

- Lifelong Semantic Adaptation Module will utilize RGBD information from the optical sensors of the platform and Input form the Visual Scene Understanding Module to update the adaptable operational environment models to correspond to the real-time environmental conditions during the robot operation.

Dependencies to other components


Adaptive Modelling Module
Semantic 3D SLAM module
Operational Environments Adaptable Model Generator
Lifelong Semantic Adaptation Module
Visual Scene Understanding Module
Visual Sensing Module
Adaptive Task Planner

Adaptive Modelling Module interacts with the following components:

Visual Scene Understanding Module (From)

- This component expects input from the Visual Scene Understanding Module in order to utilize the semantic information regarding the detected objects and actors and the structural information of the operational scene.

Visual Sensing Module (From)

- The Adaptive Modelling Module expects input from the Visual Sensing Module in order to utilize the detected instances of objects and actors, and their poses.

Adaptive Task Planner (From/To)

- The Adaptive Modelling interaction with the Adaptive Task Planner Module is bidirectional, as the Adaptive Task Planner will adapt the platforms behaviour based to the context information derived from the Adaptive Modelling Module, and the operational environment model will be adapted using information provided by the Adaptive Task Planner Module.

Addressed Requirements

Requirements ID	Description
SFR13	Recognize how many boxes of a specific product item can fit on the shelf
SFR18	Recognize misplaced products on the shelves
SFR22	Recognize the proximate area and shelves and compare the predefined assortment planogram with the actual placement of the ESLs
SFR24	Recognize product assortment on shelves according to ESLs and per product defined area (by assortment plan)
SFR25	Identify the correct destination on the shelf (through ESL) given the prior recognition of the mixed pallet and the respective product on the mixed pallet
SFR34	Ability to move on the pavement area alongside the conveyor belt
SFR36	Navigate avoiding restricted areas
SFR68	Ability to complete restocking tasks for shelves within a mobility radius (parameter with default value e.g., 5 meters) from the mixed pallet
SFR69	Decides from where it can leave the pavement track (either because it has been instructed to by the operator or if it needs to be charged)

5.14 Scene & Task Graph Generator

Description

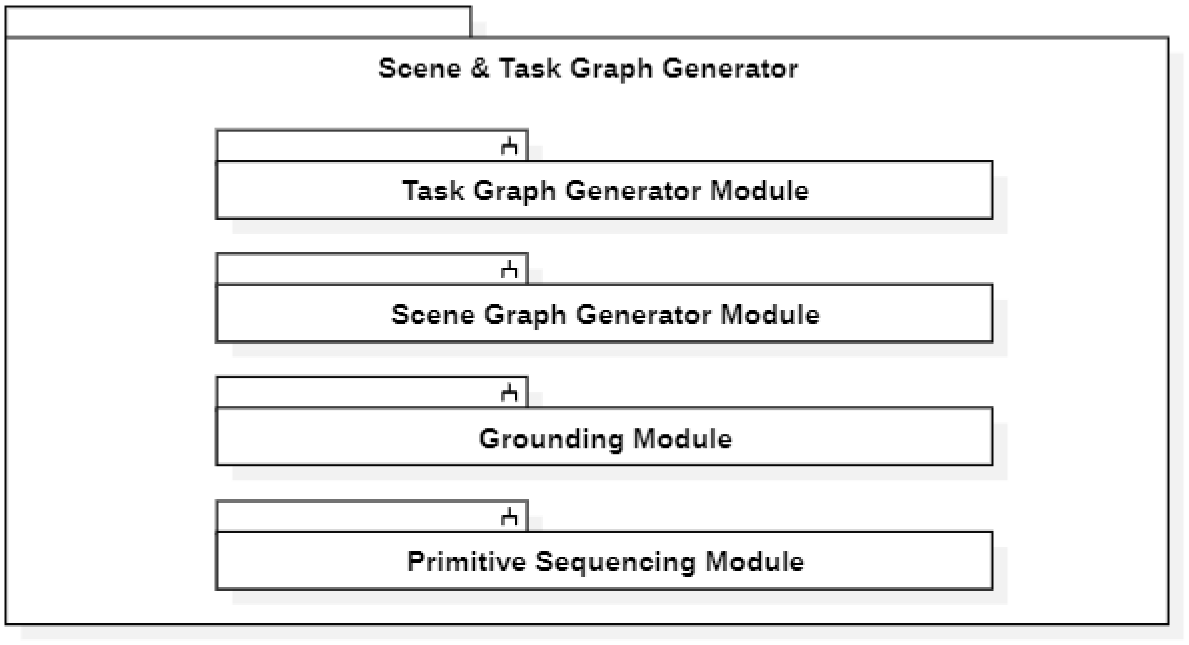
This task will provide a scene and task graph representation for context-aware reactive planning of mobile manipulation policies. The aim is to have a graph representation that enables grounding the generated task plans on the scene affordances as well as the robot's skills. Additionally, we will investigate the integration of learning and planning to enable the robot to condition on new contexts of domain or task description.

Main Functionalities

The key functionalities of this module are:

- Task Graph generation from a plan given by the Bi-manual Manipulations Orchestrator (T5.6) using the primitives learned in the Multi-modal adaptive control Module.
- Create a Scene Affordance Graph using information about the robot's position, semantic information, objects in the scene, perceived affordances, etc. from the Visual Sensing Module, Visual Scene Understanding Module, and Adaptive Modelling Module.
- Grounding the proposed plan using the Scene Affordance Graph and the Task Graph to ensure its feasibility based on the perceived scene and the reachability of the robot.
- Sequencing the learned primitives to adapt the learned skills based on the grounding.

Internal Components


Scene & Task Graph Generator
Task Graph Generator Module
Scene Graph Generator Module
Grounding Module
Primitive Sequencing Module

Task Graph Generator Module

- The goal of the Task Graph Module is to parse the generated Task Plan from the Bimanual Manipulations Orchestrator (T5.6) and create a coarse Task Graph using the learned skills available to the robot learned by the Multi-modal adaptive control Module.

Scene Graph Generator Module

- Based on the perceived objects in the scene, a hierarchical graph-based representation would be created that captures the semantic (category, affordance) and positional (location, shape) information of the different objects in the scene as well as the relations between the different objects (contact information, semantic positions like "in", "next to", "front of", "behind", etc.).

Grounding Module

- Once a coarse task graph has been generated and the scene has been perceived, the gap between abstract task descriptions and the physical world task plan needs to be bridged via grounding based on sensory perception and actionable tasks such as :
    - scene affordances, such as whether a given object can be grasped or not based on the geometry of the object and the scene, or if any new objects in the scene need to be considered.
    - task-related affordances, for example, where to grasp/place an object based on the task, such as lifting a box vs opening a box

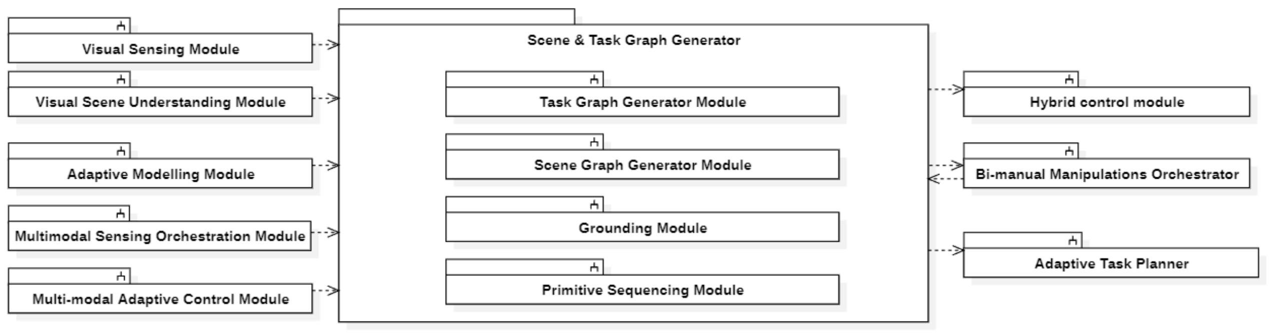
- reachability of the robot i.e. can the robot reach a position to perform a given task based on its kinematic and dynamic constraints, does it need bimanual manipulation or not, etc.

- This information would be used to refine the initial coarse task plan and provide constraints to the underlying skills for ensuring task success.

Primitive Sequencing Module

- Once the coarse Task Plan has been refined, the set of learned primitives that are used in the task needs to be effectively sequenced. For doing so, a Graph Neural Network would be employed to parse the refined Task Graph to learn heuristics that assist search-based task planning algorithms for generating an optimal sequence of actions which can then be provided to the Hybrid Control Module.

Dependencies to other components


Visual Sensing Module
Visual Scene Understanding Module
Adaptive Modelling Module
Multimodal Sensing Orchestration Module
Multi-modal Adaptive Control Module
Scene & Task Graph Generator
Task Graph Generator Module
Scene Graph Generator Module
Grounding Module
Primitive Sequencing Module
Hybrid control module
Bi-manual Manipulations Orchestrator
Adaptive Task Planner

Visual Sensing Module (From)

- The Visual Sensing Module would be queried to obtain information about the objects perceived in the scene, using which the positional aspects in Scene Graph representation would be generated.

Adaptive Modelling Module and Visual Scene Understanding Module (From)

- Object and Scene semantics provided by the Visual Scene Understanding Module would be needed to create the semantic relations and affordance information in the Scene Graph Representations.

Multimodal Sensing Orchestration Module (From)

- The Multimodal Sensing Orchestration Module would need the robot's current state (using SLAM, for example) to create the Scene Graph and Task Graph Representations, particularly its reachability with regard to the scene and the objects.

Multi-modal adaptive control Module (From)

- The Manipulation Motion Primitives Generator would provide the different skills that the robot is equipped (pick, place, push, grasp, etc.) with as well as the parametrizations of the constraints (object affordances for pick, location to place/push, grasp strength, etc.) which would then be used to solve for the Task and Motion Planning.

Hybrid Control Module (To)

- Once a task plan is created, the sequence of skills and the constraints will then be provided to the Hybrid Control Module, which will take care of successfully executing the sequence of primitives according to the task sequence.

Bi-manual Manipulations Orchestrator (To/From)

- The Bi-manual Manipulations Orchestrator would create a high-level task plan using the generated Scene Graph representation would be used to create a Task Graph that would then be used for Grounding and subsequent refinement.

Adaptive Task Planner (To)

- The generated task graph would be provided to the Adaptive Task Planner for reactive planning based on the assessment of the task progress.

Addressed Requirements

Requirements ID	Description
SFR038	Grasp products from boxes
SFR039	Ability to pick and place single items
SFR040	Ability to pick up boxes with max weight up to 10 kg
SFR041	Ability to pick up boxes without damaging them
SFR042	Place the products in the correct positions
SFR043	Place the products that expire earlier in the front of the shelf ahead of newly received items
SFR044	Place the products in a specified orientation, ensuring the front side of the product faces the customer
SFR045	Place momentarily the boxes on its equipped table or another interim storage position
SFR046	Manipulate and replenish product boxes
SFR050	Grasp the luggage that made from soft material gently
SFR051	Ability to prepare the products/SKUs (e.g., removing cardbox or plastic foil toppers)
SFR052	Place misplaced products in an interim storage position or to the cart or to the right shelf position
SFR053	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position
SFR054	Place the products in the right position without using a planogram or consult it when it is available
SFR055	Place the luggage made of hard material below the luggage made of soft material
SFR056	Place the luggage with a distance between them on the reclaim belt for the arrivals
SFR057	Place the luggage within the boundaries of the belt/cart
SFR058	Handle luggage only for the flight that has been allocated to
SFR059	Ability to restock shelves with product/SKUs from the pallet (top to bottom)
SFR060	Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism
SFR061	Place empty (cardboard) boxes from replenished items into a bin cart located next to the pallet
SFR062	Manipulate the baggage to detect the bagtag

5.15 Learning to plan from Demonstrations Module

Description

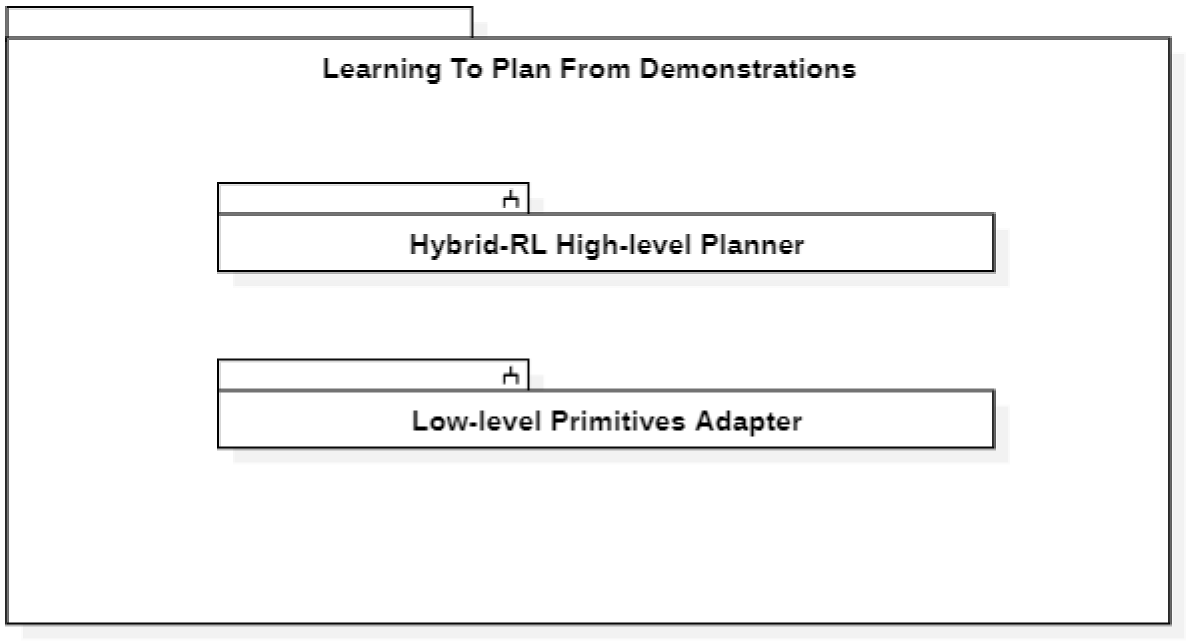
This module provides a task planning system that learns from demonstration data or robot experience. Its goal is to establish a hierarchical planning framework that adapts to the robot's various operational cycles. By analysing demonstration data, the system will infer and plan different time phases and options (these are higher-level actions derived from the timing and sequence of lower-level policies).

Main Functionalities

The key functionalities of this module are:

- Data-driven task plan and sequence generation. This will be done by learning from demonstration data and experience in a hybrid-RL setting (Reinforcement learning using offline and online data).
- Option generation and selection. This will involve choosing the appropriate lower-level motor policies and the options or parameters for these policies.

Internal Components


Learning To Plan From Demonstrations
Hybrid-RL High-level Planner
Low-level Primitives Adapter

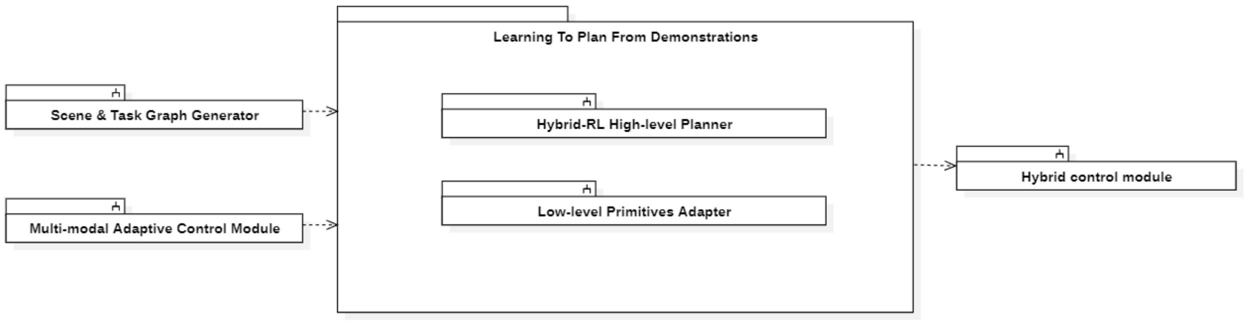
Hybrid-RL High-level Planner

- The goal of this component will be to use reinforcement learning to train an agent to output plans (or plan adaptations) that lead to a high-level task being fulfilled in an optimal way. We will collect a new set of task-specific human demonstrations and pre-train the RL agent using offline RL. Then, we will leverage a simulator to gather experience (online data) to further improve the planning performance of the RL agent.

Low-level Primitives Adapter

- This component will use learning to provide sub-goals and options to be used by lower-level policies and primitives (WP4). For example, we will learn to predict which time phase of the operational cycle the robot is in and, accordingly, what sub-goal the robot's end-effector should follow at the current and next timesteps.

Dependencies to other components


Learning To Plan From Demonstrations
Scene & Task Graph Generator
Multi-modal Adaptive Control Module
Hybrid-RL High-level Planner
Low-level Primitives Adapter
Hybrid control module

Scene & Task Graph Generator (From)

- The Scene & Task Graph obtained from the corresponding module would be needed to query and predict the task plan. The task graph will provide the current state of the world and the possibilities that will then be used for planning.

Multi-modal adaptive control Module (From)

- The Manipulation Motion Primitives Generator would provide the different skills that the robot is equipped (pick, place, push, grasp, etc.) with as well as the parametrizations of the constraints (object affordances for pick, location to place/push, grasp strength, etc.) which would then be used to solve Task and Motion Planning.

Hybrid Control Module (To)

- Once a Task Plan and options are generated, the sequence of skills and parameters will be provided to the Hybrid Control Module, which will then take care of successfully executing the sequence of primitives according to the task sequence & options.

Addressed Requirements

Requirements ID	Description
SFR38	Grasp products from boxes
SFR44	Place the products in a specified orientation, ensuring the front side of the product faces the customer
SFR46	Manipulate and replenish product boxes
SFR50	Grasp the luggage that made from soft material gently
SFR52	Place misplaced products in an interim storage position or to the cart or to the right shelf position
SFR53	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position
SFR62	Manipulate the baggage to detect the bagtag

5.16 Adaptive Task Planner

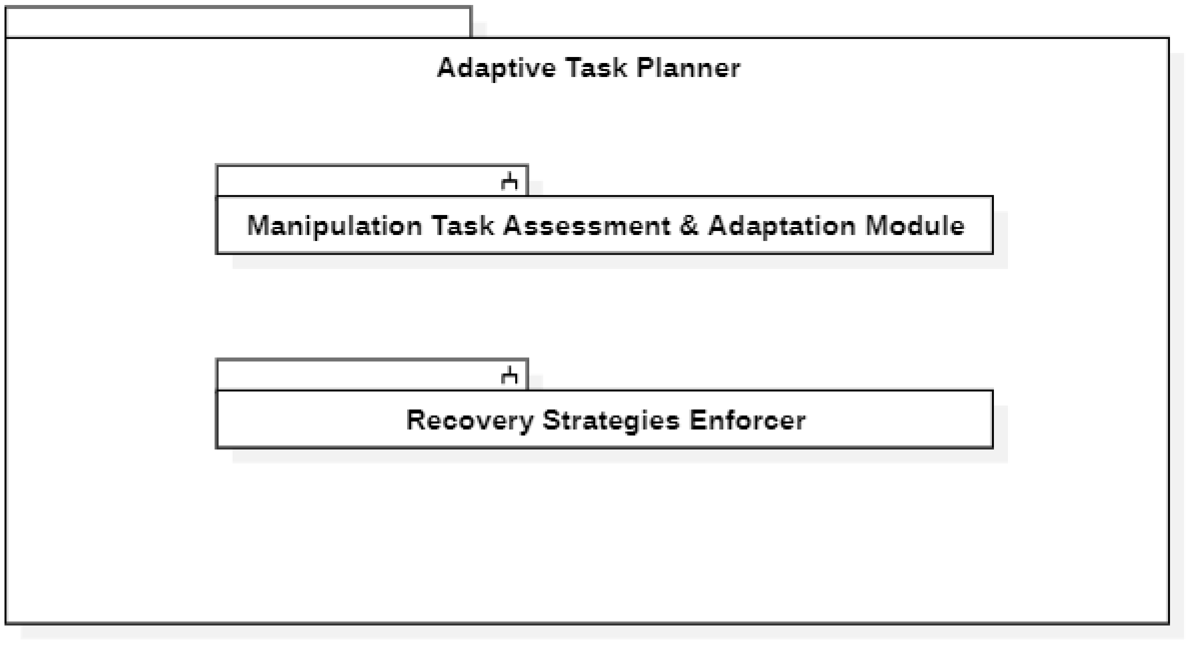
Description

This module will provide reactive task-level strategies for adapting the robot’s behavior both on the low-level and mid-level operation cycles. This will be done through reinforcement learning and monte-carlo tree search on the scene & task graph. Moreover, the module will provide a set of strategies to recover from failures and adapt to new conditions.

Main Functionalities

Given the encoded scene and affordances as a graph (Scene & Task Graph Generator), the set of feasible skills or behaviors (Multi-modal adaptive control Module), and the current high-level plan (Bi-manual Manipulations Orchestrator), this module will react & adapt the current task plan. Moreover, success and failure of sub-tasks or skills will be incorporated and reacted upon using a combination of learned and hand-designed adaptation or recovery modules.

Internal Components


Adaptive Task Planner
Manipulation Task Assessment & Adaptation Module
Recovery Strategies Enforcer

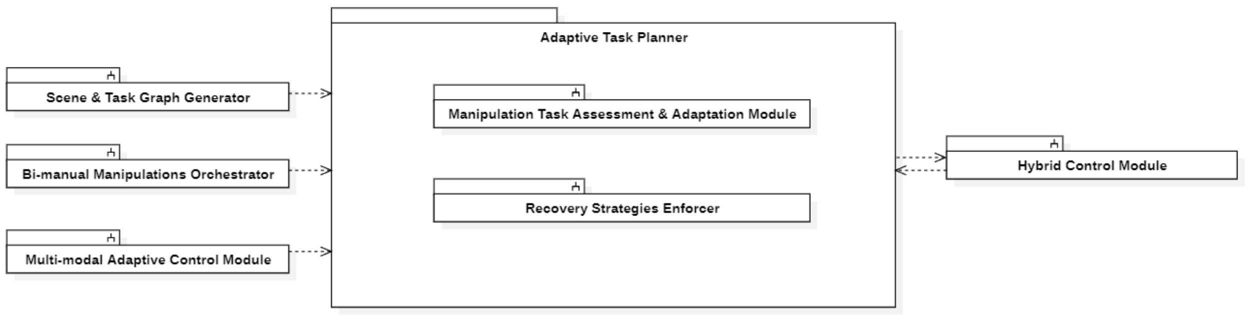
Manipulation Task Assessment & Adaptation Module

- This component will assess the feasibility or optimality of the current task plan and adapt it using reinforcement learning on simulated experience or using monte-carlo tree search to find the optimal plan. As the planning will be done on graphs at multiple levels, neural message passing will be used to inform about successes or failures in a specific task and adapt the plans accordingly.

Recovery Strategies Enforcer

- This component will provide a set of recovery strategies when failures are detected so that the robot can re-evaluate its current mid-level operational plan on the fly.

Dependencies to other components


Adaptive Task Planner
Manipulation Task Assessment & Adaptation Module
Recovery Strategies Enforcer
Scene & Task Graph Generator
Bi-manual Manipulations Orchestrator
Multi-modal Adaptive Control Module
Hybrid Control Module

Scene & Task Graph Generator (From)

- The Scene & Task Graph obtained from the corresponding module would be needed to query and adapt the task plan. The task graph will provide the current state of the world and the possibilities that will then be used for planning & adaptation.

Bi-manual Manipulations Orchestrator (From)

- The current plan (or plans) will be obtained from the Bi-manual Manipulations Orchestrator module and will be evaluated and modified on the go in a reactive fashion.

Multi-modal adaptive control Module (From)

- The Manipulation Motion Primitives Generator would provide the different skills that the robot is equipped (pick, place, push, grasp, etc.) with as well as the parametrizations of the constraints (object affordances for pick, location to place/push, grasp strength, etc.) which would then be used to solve Task and Motion Planning.

Hybrid Control Module (To)

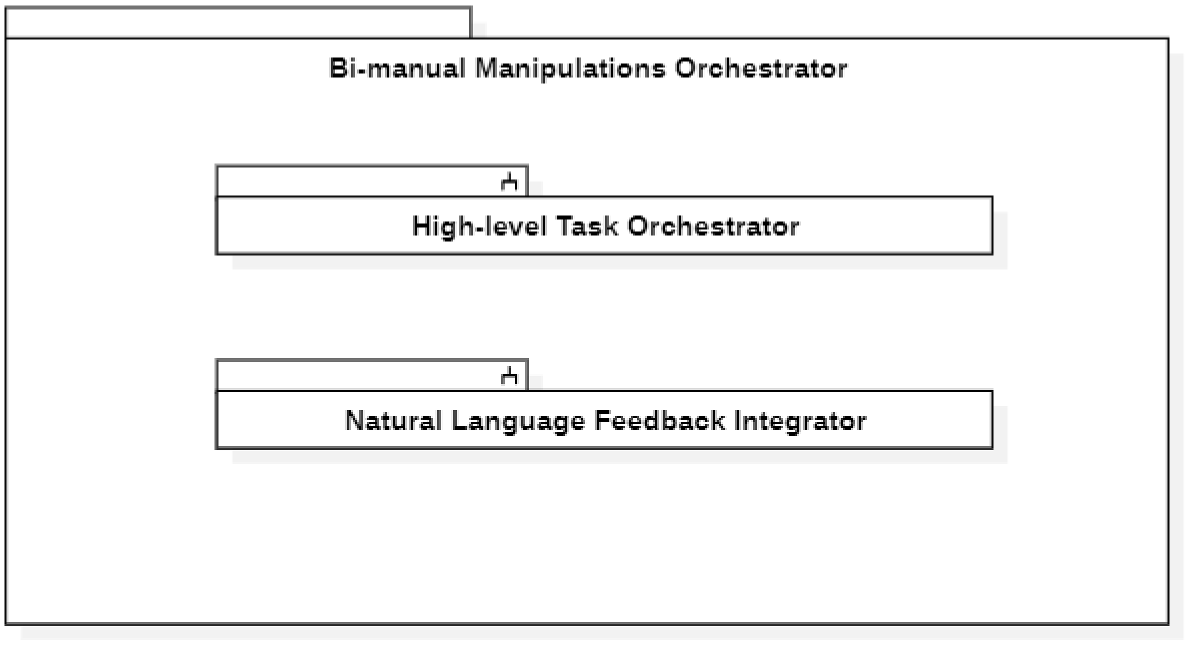
- Once the task plan has been adapted, the sequence of skills and the constraints will then be provided to the Hybrid Control Module, which will then take care of successfully executing the sequence of primitives according to the task sequence.

Addressed Requirements

Requirements ID	Description
SFR38	Grasp products from boxes
SFR44	Place the products in a specified orientation, ensuring the front side of the product faces the customer
SFR46	Manipulate and replenish product boxes
SFR50	Grasp the luggage that made from soft material gently
SFR52	Place misplaced products in an interim storage position or to the cart or to the right shelf position
SFR53	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position
SFR62	Manipulate the baggage to detect the bagtag
SFR066	Robot pauses its current task if employee or customer is too close to it
SFR067	Robots pauses and resumes tasks according to user's command
SFR068	Ability to complete restocking tasks for shelves within a mobility radius (parameter with default value e.g., 5 meters) from the mixed pallet

SFR069	Decides from where it can leave the pavement track (either because it has been instructed to by the operator or if it needs to be charged)

5.17 Bi-manual Manipulations Orchestrator


Bi-manual Manipulations Orchestrator
High-level Task Orchestrator
Natural Language Feedback Integrator

Description

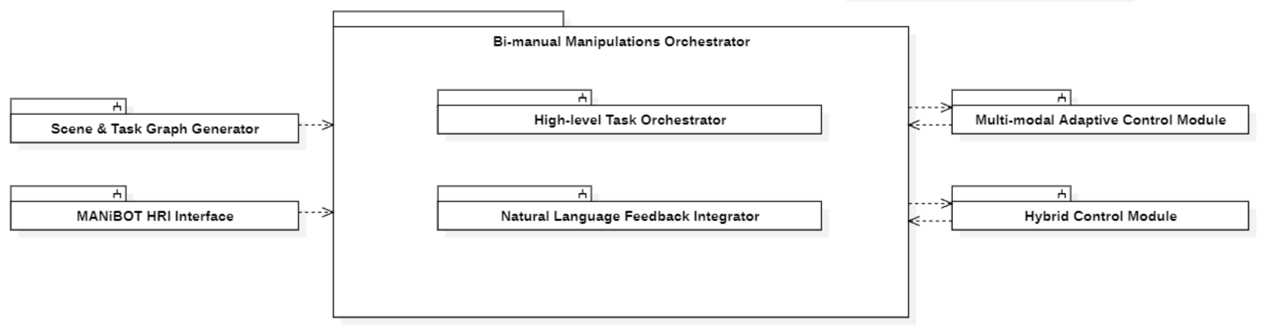
Using information from demonstrated data of tasks, which consists of the robot's actions, the perceived scene information, and the robot's skills, a pre-trained LLM would be fine-tuned using RL for effectively generating a successful task plan. This would be done using the low-level primitives such that the learning of the task plan takes into account the execution abilities of the robot. Moreover, since we would be fine-tuning an LLM, we can additionally incorporate Natural Language feedback based on a human user's preferences via the MANiBOT HRI Interface.

Main Functionalities

The key functionalities of this module are:

- Reinforcement Learning for generating successful task plans that take into account the execution capabilities of the robot.
- Incorporating Natural Language Feedback via the MANiBOT HRI Interface.

Internal Components


Bi-manual Manipulations Orchestrator
Scene & Task Graph Generator
MANiBOT HRI Interface
High-level Task Orchestrator
Natural Language Feedback Integrator
Multi-modal Adaptive Control Module
Hybrid Control Module

High-level Task Orchestrator

- The High-level Task Orchestrator will be responsible for generating the coarse task plan for the robot to perform a given task, which would be provided via the MANiBOT HRI Interface. The Natural Language Feedback Integrator would incorporate this information as global attributes in the perceived Scene Graph, which would then be given as inputs to an LLM model that is fine-tuned using RL that would use the learned robot skills to determine the success of a proposed task plan. This underlying success of execution, as compared to just generating a correct sequencing of skills, would be able to better inform the learning process and keep it grounded with what the robot can execute.

Natural Language Feedback Integrator

- Based on Natural Language inputs as well as high-level task scheduling from the MANiBOT HRI Interface, the Natural Language Feedback Integrator would consist of developing a representation that encodes the operator feedback from multiple modalities (task assignment, natural language feedback, alerts) would be fused to provide "advice" to the High-level Task Orchestrator.

Dependencies to other components

Scene & Task Graph Generator (From)

- The Scene and Task Graph from the **Scene & Task Graph Generator** would be used as input to the High-level Task Orchestrator.

Multi-modal adaptive control Module and Hybrid Control Module (From/To)

- The learned primitives from the multi-modal adaptive control module would be used to guide the learning by providing the predicted sequencing of the primitives to the Hybrid Control Module to see the success of the sequencing execution.

MANiBOT HRI Interface (From)

- User inputs from the MANiBOT HRI Interface would be used to provide global context to the High-level Task Orchestrator that helps specify the task the robot needs to do, as well as additional guidance via natural language.

Addressed Requirements

Requirements ID	Description
SFR38	Grasp products from boxes
SFR44	Place the products in a specified orientation, ensuring the front side of the product faces the customer
SFR46	Manipulate and replenish product boxes
SFR50	Grasp the luggage that made from soft material gently

SFR52	Place misplaced products in an interim storage position or to the cart or to the right shelf position
SFR53	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position
SFR62	Manipulate the baggage to detect the bagtag

5.18 MANiBOT HRI Interface

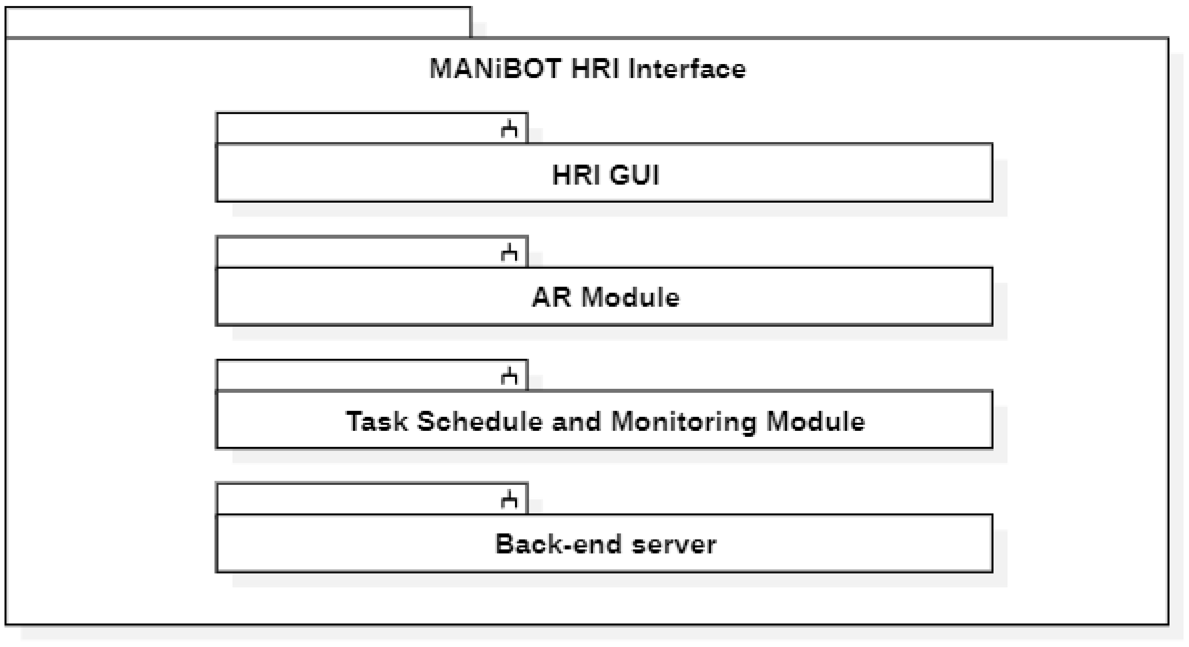
Description

The HRI Interface will provide a user-friendly graphical interface with multiple tools. With the HRI the human operator can communicate with the MANiBOT robot, setting it up and supervising the process of each use-case at all times. It is designed to facilitate intuitive, seamless, and safe communication between human operators and the robotic systems. This module serves as the primary point of interaction for users to control, monitor, and collaborate with robots, ensuring that human inputs are correctly interpreted and acted upon by the robotic systems.

Main Functionalities

The HRI interface facilitates collaborative task assignments between the human operator and the robot, allowing users to delegate specific tasks to the robot (e.g., navigation, loading/unloading tasks). It also allows for task progress monitoring, ensuring that the robots are correctly following instructions and adjusting as needed in response to real-time feedback from users or environmental changes. The HRI module offers controls to the robot, for the times when the robot is needed to be in manual mode instead of autonomous, task scheduling and monitoring properties, and AR features.

Internal Components



The HRI GUI consists of the following internal components:

HRI GUI

- The HRI GUI submodule provides a visually intuitive interface that allows users to interact with the robotic systems using graphical input devices. The submodule aims to provide an easy-to-use, responsive, and user-centered graphical interface that allows operators to control, monitor, and collaborate with robots in the MANiBOT environment. The HRI GUI is designed with simplicity and clarity in mind, allowing users to interact with robotic systems without requiring technical knowledge.

AR Module

- The AR (Augmented Reality) submodule of the MANiBOT HRI Interface improves human-robot interaction by incorporating augmented reality technologies into the user interface. This submodule enables the use of augmented reality devices to dynamically and interactively visualize robot tasks, paths, diagnostics, and the surrounding environment. AR allows users to intuitively control and monitor robots, improving decision-making and operational efficiency.

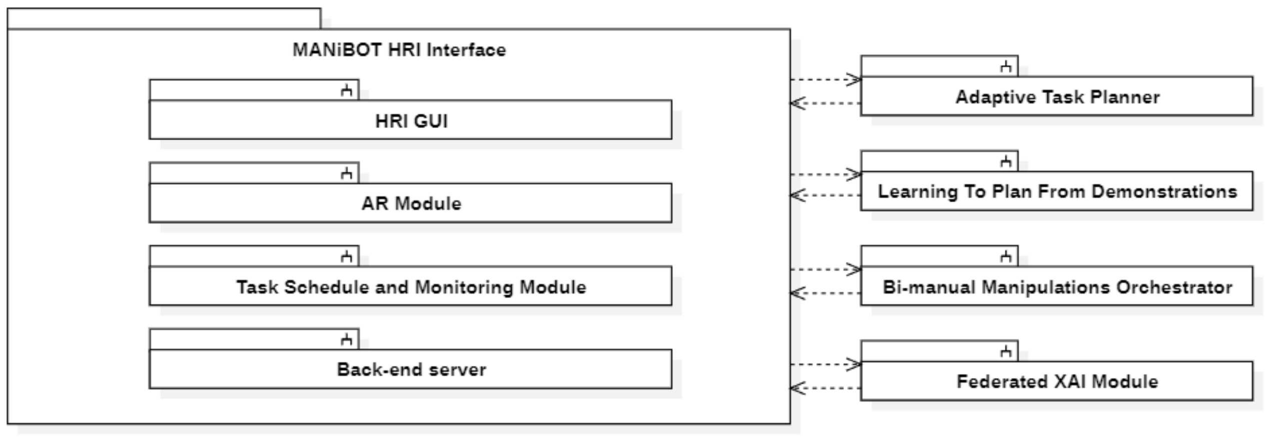
Task Schedule and Monitoring Module

- With the task schedule module, the operator can select new tasks to add to the robot's schedule tasks, or create custom tasks, depending on the situation of the user-case and robot's status. Concurrently, this submodule includes monitoring processes that will provide alerts and logs generated by the HRI and the robot. These will always keep the operator updated at all times for the robot's status, the process of the user-case, and will also provide errors that conflict with the flow of the scenario and need to be fixed in order to continue the process.

Back-end server

- The back-end server is the main point of the communication between the operator and the robot. All the information visualized on the HRI and all data management will take place in the back-end, including the connection with the robot. This submodule ensures that the HRI interface's various components can communicate with robotic systems seamlessly, process data efficiently, and respond to users in real time.

Dependencies to other components


MANiBOT HRI Interface
HRI GUI
AR Module
Task Schedule and Monitoring Module
Back-end server
Adaptive Task Planner
Learning To Plan From Demonstrations
Bi-manual Manipulations Orchestrator
Federated XAI Module

The HRI interacts with the following components:

Adaptive Task Planner (From/To)

- As users assign tasks or change commands via the HRI Interface, the Adaptive Task Planner dynamically adjusts the robot's behavior and execution strategies based on real-time environmental data and user inputs. This ensures a seamless transition from user intent to robotic action, allowing for efficient task execution and quick adjustments to changing conditions. In contrast, the Adaptive Task Planner's insights and updates feed back into the HRI Interface, providing users with real-time feedback on task progress and status updates.

Learning to plan from Demonstrations (From/To)

- When the human operator interacts with the HRI Interface to assign tasks or provide demonstrations, the Learning to Plan from Demonstrations module analyzes this input and uses demonstration data to develop a hierarchical planning framework. Because of this relationship, user demonstrations can be seamlessly transferred to efficient robotic task execution, allowing the robot to adaptively infer and carry out higher-level actions from lower-level policies that are based on prior knowledge.

Bi-Manual Manipulations Orchestrator (From/To)

- The Bi-manual Manipulations Orchestrator analyzes demonstrated data, such as the robot's actions, perceived scene information, and available skills, to optimize and refine the robot's task planning. The orchestrator uses reinforcement learning to fine-tune a pre-trained large language model (LLM), making sure that the task plans that are generated match the robot's execution capabilities. By integrating natural language feedback, the Bi-manual Manipulations Orchestrator improves the Human-Robot Interface (HRI) by enabling users to provide input in a more intuitive way and promoting a collaborative environment where human preferences are seamlessly integrated into robotic actions.

Federated XAI Module (From/To)

- The Federated XAI Module, which enhances user interactions by incorporating Explainable AI and advanced federated learning, is a critical component that the HRI Interface depends on. Users' feedback and interactions with the HRI Interface help the system learn through federated learning, which enhances its perception modules. Since the Federated XAI Module transparently explains the robot's decision-making processes while analyzing and learning from distributed data, this relationship guarantees a smooth transition from improved robotic performance to user experiences.

Addressed Requirements

Requirements ID	Description
SFR070	User defining replenishment task and destination
SFR071	User defining task and destination in the airport belts
SFR072	User commanding through interaction interface the robot to move to specified location (shelf or belt)
SFR073	Interaction interface notifies the user about the restocking task progress
SFR074	Interaction interface notifies the user if the robot needs help with the identification of a product, bag etc.
SFR075	Interaction interface notifies the user for help when a product falls on the floor
SFR076	Ability to give a signal (either audio or visual) when robot's workspace is invaded by a human

SFR077	Interaction interface notifies the user when the replenishment process is complete
SFR078	Interaction interface notifies the user if any of the processes failed
SFR079	Interaction interface notifies the user about the absence of planogram and difficulty in defining the right product position
SFR080	Interaction interface alerts when the bag it is manipulating is out of its capabilities (too heavy, too long, no handles, out of reach etc.).
SFR081	Interaction interface alerts when no bagtag found
SFR082	The robot alerts the operator if no other cart is found, goes in standby mode and awaits input from operator.
SFR083	Notifies the user through the UI for the table/interim storage state

5.19 Data Management Module

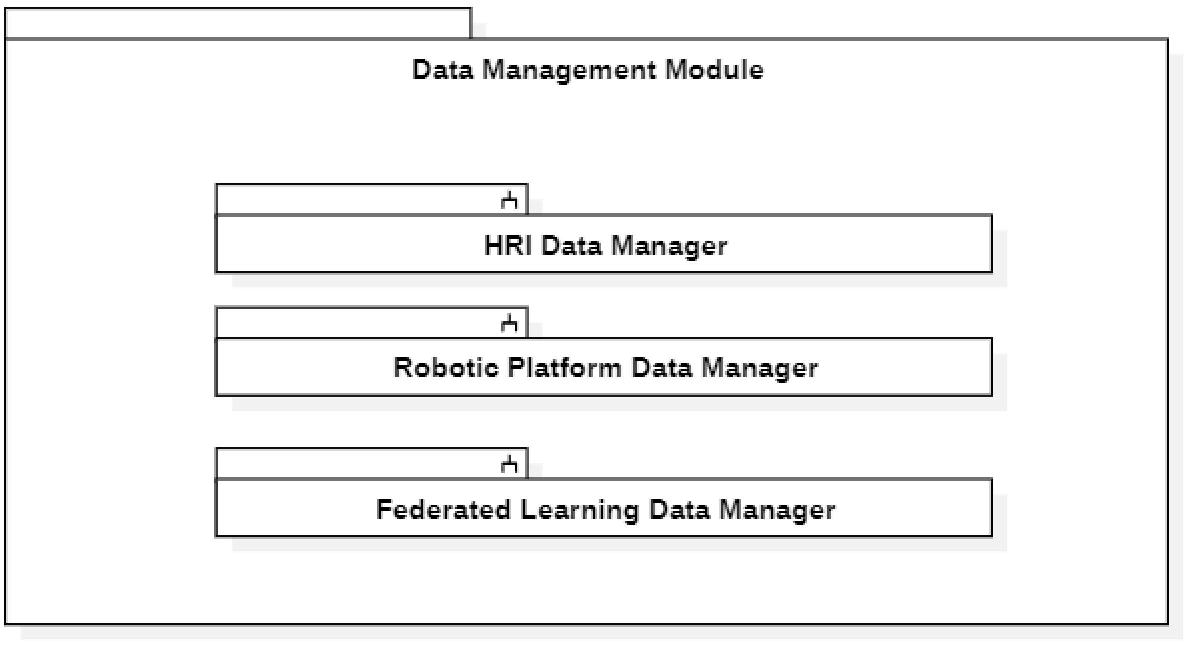
Description

This module focuses on creating the MANiBOT data management system's front-end and back-end to facilitate the safe sharing, offline processing, storage, and access of heterogeneous data. Specifically, the data categories listed below will be supported: i) Robot path and motion data; ii) diagnostic information about the functioning of the robotic system; iii) maps of environment and utilities; iv) information about baggage, including bagtags and the quantity of bags loaded; v) localization and real time path measurements; vi) items (surplus, expired, damaged) that are not suitable for restocking. The data will be accessed by various modules of the MANiBOT system.

Main Functionalities

The module streamlines the process of storing and retrieving data by offering interfaces for creating, deleting, altering, and accessing data generated by the system's modules. It makes it possible to handle and process the data related to the human operator and the robotic system while storing and managing the data for the federated learning module.

Internal Components


Data Management Module
HRI Data Manager
Robotic Platform Data Manager
Federated Learning Data Manager

The data management module consists of the following internal components:

HRI Data Manager

- The HRI Data Manager is in charge of collecting, processing, and managing data on human-robot interaction interface in real time within the MANiBOT system. This module will be responsible for processing the requests and retrieving information about the operator's robot assignments, the scheduling and the monitoring of them. It also handles data associated with notifications for the operator regarding any issues where the robot requires assistance or asks for clarification on the products.

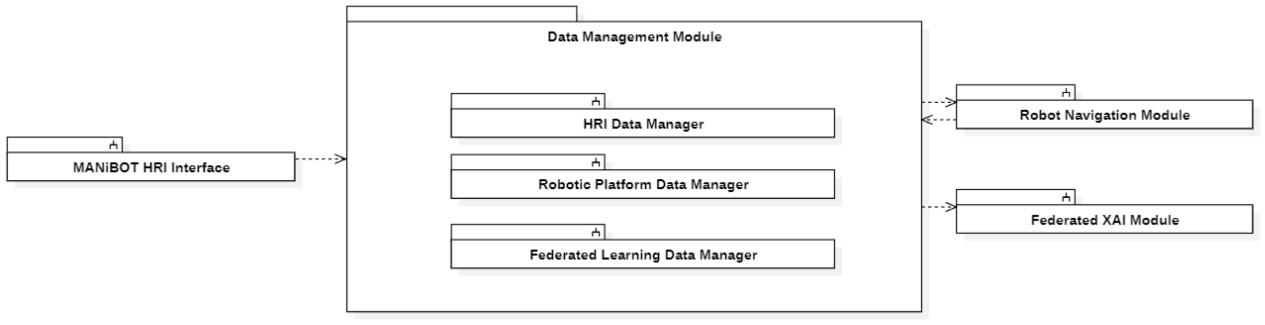
Robotic Platform Data Manager

- Within the MANiBOT system, all data related to the robotic platforms is managed by the Robotic Platform Data Manager. This covers data gathering, processing, storing, and retrieval related to the hardware, software, motion, diagnostics, and general operation of the robot. The module collects real-time diagnostic data on the robotic system's health, including battery levels, motor performance and sensor functionality. It manages the information acquired by the robot's sensors about the environment, including mapping and obstacle recognition in order to facilitate task completion and navigation.

Federated Learning Data Manager

- The Federated Learning Data Manager submodule will focus on managing the data required to enable federated learning across the MANiBOT system. This submodule ensures that the learning process is carried out efficiently, securely, and in a manner that protects data privacy, all while optimizing the overall model's performance across the system. This enables the training of machine learning models across multiple decentralized entities, such as devices/robots or servers, without the need to centralize data, ensuring privacy.

Dependencies to other components



The data management module interacts with the following components:

MANiBOT HRI Interface (To/From)

- The Data Management Module is critical to the MANiBOT HRI Interface because it allows for the organization and access of various data types, which is essential for handling human-robot interactions. The HRI Interface provides user-generated data back into the Data Management Module, storing data and facilitating continuous learning and adaptation on the MANiBOT platform.

Robot Navigation Module (To/From)

- The Data Management Module makes sure that the Robot Navigation Module has access to critical information like robot path and motion data, localization metrics, and environmental maps. This comprehensive data infrastructure enables the Robot Navigation Module to make informed decisions about route planning, obstacle avoidance, and real-time adjustments while performing navigation tasks. The Robot Navigation Module contributes to the Data Management Module by generating new data about navigation outcomes and environmental interactions.
- **Multi-Modal Adaptive Control Module (To)** The Data Management Module provides data to Multi-Modal Adaptive Control Module for performing manipulation tasks. By integrating this data, the Multi-modal Adaptive Control Module can optimize its grasping and manipulation actions, resulting in better performance in complex tasks.

Federated XAI Module (To)

This module provides data to the Federated XAI Module to effectively analyze and interpret information from various sources, in order to facilitate federated learning processes and improve the system's understanding of its surroundings. Based on the processed data, the Federated XAI Module produces insights and explanations. It then provides insightful feedback to the Data Management Module, enhancing the data repository with interpretive data that can be used for additional analysis and improvement

Addressed Requirements

Requirements ID	Description
SFR084	Robot stores/retrieves the bagtag information of the placed baggage in/from the database
SFR085	Scans barcode and retrieves information of the paper label and checks if it corresponds to the product to be replenished
SFR086	Retrieves information about the ESL tags
SFR087	If a mismatch identified, log the distance between the position of a product in the assortment plan and the position of the ESL
SFR088	Keep count of the number of baggage loaded

SFR089	Scans the bag tag (when placing the baggage) for BTRS (Baggage Tracking and Reconciliation System) purposes.
SFR090	Records and logs in the database the products that are not suitable for restocking (surplus, expired, damaged)

6 Technical Specifications

This chapter introduces the technical specifications of the MANiBOT system. Technical specifications can be divided into three categories: hardware, software, and performance requirements. Performance requirements, software, and hardware are the three categories into which technical specifications can be divided. The system's hardware specifications for carrying out the project's tasks are referred to as hardware requirements. The safety, security, and functionality requirements that the system's software components must satisfy are referred to as software requirements. Performance requirements are the system's quantitative capabilities.

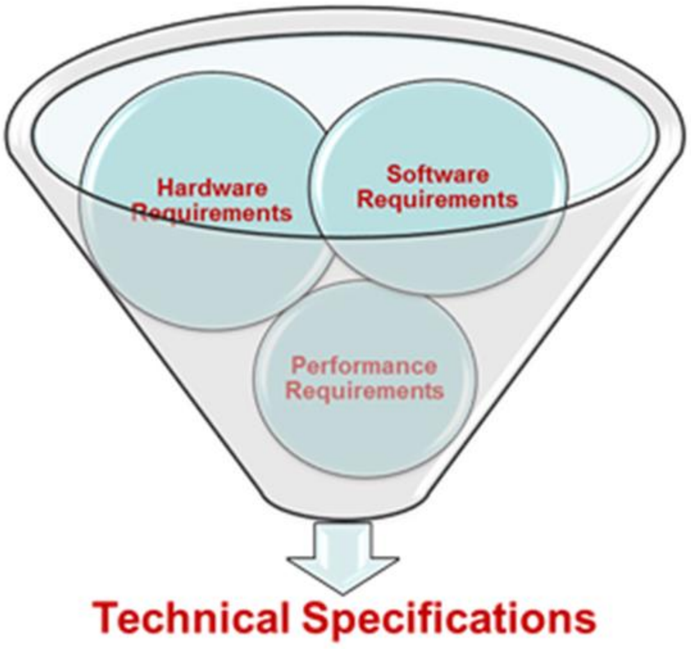


Figure 6 Illustrative representation on how the Technical Specifications are broken down

In summary, in section 6.1 we present the mapping procedure from User Requirements to technical specification. In section 6.1 there are three sections included, sections 6.1.1, 6.1.2 and 6.1.3 which outline the performance, hardware, and software requirements in detail. The main topic of section 6.2 is the prioritization of the functional requirements, which are derived from the user requirements' priority. Finally, section 6.3 describes the handling requirements in MANiBOT use cases that will lead to the corresponding Robotic Platforms' specifications. This section complements the two previous ones, since although handling requirements were extracted parallel to the analysis of user requirements, they were not reported in D2.2. Instead, they were combined with the pilot sites' description and the items considered for robotic manipulation in MANiBOT use cases in a holistic analysis that maps them directly to MANIBOT corresponding robotic platforms' specifications (see section 8.3.1.1).

6.1 Mapping of the User Requirements to Technical Specifications

There are two steps involved in mapping user requirements to technical specifications. The user requirements are first broken down into distinct use cases and their corresponding sub-use cases. Every sub-use case is examined in terms of a set of functional requirements, each of which is closely linked to particular performance requirements. The latter is the interface link that converts technical specifications into user requirements. The performance requirements act as a bridge that connects the technical specifications to the user needs. They ensure that the technical aspects, like hardware capabilities, software algorithms, and system architecture, are aligned with what the users expect in terms of performance. This breakdown helps in clearly identifying how the system is expected to behave in various situations, ensuring that all user needs are captured comprehensively.

Since the mapping process has been stabilized by combining the functional and performance requirements, all of the sub-use cases have been examined in terms of requirements in the second stage. Here, we've included a sample of the first mapping step, which is summed up in the table below. This illustration relates to the use-case and user requirements for moving from a charging station or resting position to the target replenishment cart.

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC1.1. Movement from resting position/charging station to the target replenishment cart	*UR01,UR03,UR05,UR06,UR07, UR20,UR24,UR27,UR28,UR29, UR30,UR32,UR33,UR34,UR35,UR36*	SFR031 - Platform 2D/3D localization	real-time, < 10 cm (RPE)	2D/3D Lidars, RTK-GPS	
		SFR032 - Platform Path planning ability	real-time		
		SFR033 - Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras	OpenCV, PCL, Camera API
		SFR066 – Robot pauses its current task if employee or customer is too close to it	real-time		
		SFR003 - Recognize the replenishment cart	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR028 - Recognize and track humans' trajectory	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR036 - Navigate avoiding restricted areas	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR067 – Robot pauses and resumes tasks according to user's commands	real-time		Web browser
		SFR070 - User defining replenishment task and destination	real-time		Web browser
		SFR072 - User commanding through interaction interface the robot to move to specified location (shelf or belt)	real-time, <10cm accuracy		Web browser
		SFR078 - Interaction interface notifies the user if any of the processes failed	real-time		Web browser

The process of mapping the modules and the intricacy of their connections are depicted in Figure 7. Our system is clearly complex because every user requirement is broken down into multiple sub-use cases, each of which is connected to a number of technical specifications and functional requirements, and vice versa.

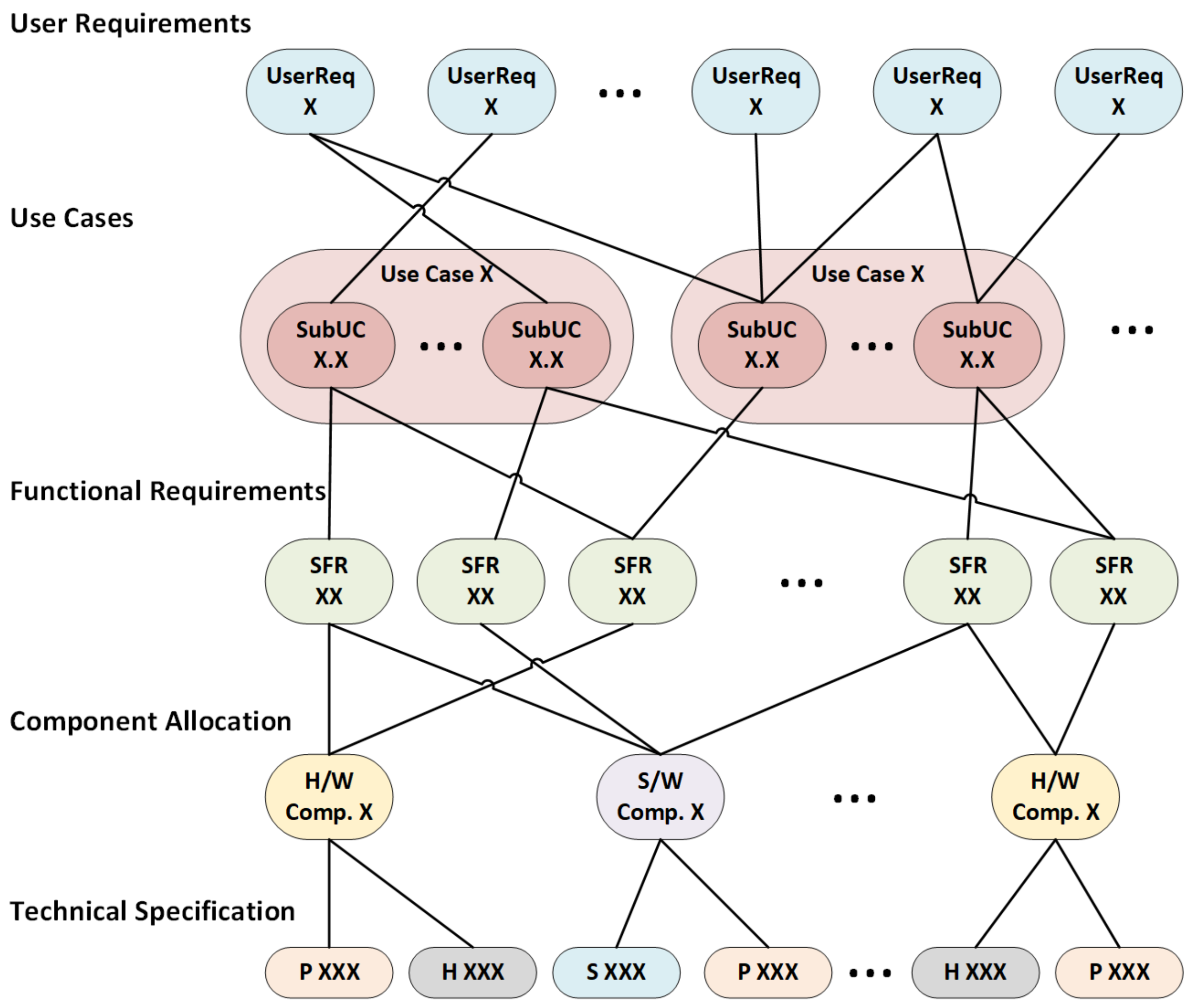


Figure 7 The mapping procedure and the complexity of the interconnections among the modules.

6.1.1 Performance Requirements

The performance requirements are essential for developing and designing the MANiBOT system, ensuring that the system meets its intended purpose. The corresponding functional requirements are defined both quantitatively and qualitatively. More specifically, the performance requirements serve as a bridge for determining the hardware and software requirements and specify the capabilities that the MANiBOT system should maintain. They include observations about the system's operational limitations, technical and physical capabilities, and time response. This also includes the system’s responsiveness, ensuring it performs efficiently under different conditions. A list of precise performance requirements focused on the fundamental hardware and software elements of the MANiBOT system has been established in conjunction with all functional requirements.

6.1.2 Hardware Requirements

Following the identification of the primary hardware components, the precise specifications for each will be developed. The physical architecture of the MANiBOT system is followed in the organization of these requirements, which relate to the hardware components that will be created or modified as part of the

MANiBOT project. The electrical, mechanical, sensing, actuation, control, safety, power supply, and usability requirements are gathered for every piece of hardware.

To retain information about the models of the hardware components and their specifications, the initially decided abstract list of hardware components should become more explicit in order to produce a thorough analysis. The final version of the hardware requirements will be included in the list of hardware requirements that will be contained in the future integration deliverable. By transitioning from an abstract outline to a detailed catalogue, hardware components can be evaluated not only for compatibility but also for performance, cost-effectiveness, and future scalability. This process will help to identify any operational constraints or potential areas for optimization, such as power consumption, processing power, or data throughput.

6.1.3 Software Requirements

The software requirements are essential for determining the needs that the MANiBOT system's software must address. They are made up of the identified essential functional elements that must be developed in order for the hardware to work together seamlessly. The software must act as the driving force that enables communication, coordination, and control between the various hardware modules. Along with serving particular routines that satisfy the demands of the sub-use cases, they are also accountable for partially meeting the performance requirements in terms of execution time and accuracy. Each sub-use case presents unique operational demands, so the software requirements must cater to these specific needs by defining how the software will handle and execute these tasks effectively. The final version of the software requirements will be included in the integration deliverable, which is currently being constructed.

6.2 Prioritization of the System Functional Requirements

Prioritizing the functional requirements to be implemented in accordance with the User-Requirements' established priorities was the aim of the analysis presented here. Consequently, entities with "High," "Medium," or "Low" priority according to particular rules make up the prioritization schema used in this process. In order to accomplish a thorough prioritization, the following standards were taken into account:

- According to the D2.2 User requirements and use cases, the priority ranking of the identified Use Cases, Sub-Use Cases, and User requirements
- The frequency with which each functional requirement appears in the particular Sub-Use Cases, as shown in the tables above. Given that it appears to be a functional requirement for handling numerous Sub-Use Cases, it should be given top priority because it shows great significance and relevance to the project.
- The dependency on critical (high priority) functional requirements required to complete a specific Sub-Use Case.

Following the aforementioned procedure, Annex II contains a summary table that shows the priority of each system functional requirement. By assessing the resulting table, it is realized that there are 90 identified system functional requirements, of which 34 have high priority, 45 have medium priority, and 11 remain low priority. Each priority's overall percentage coverage is displayed in the diagram below. It is ensured that software and hardware requirements necessary for the fulfillment of the high priority Sub-Use Cases will be developed early on by prioritizing the functional requirements, which are closely linked to the technical specifications.

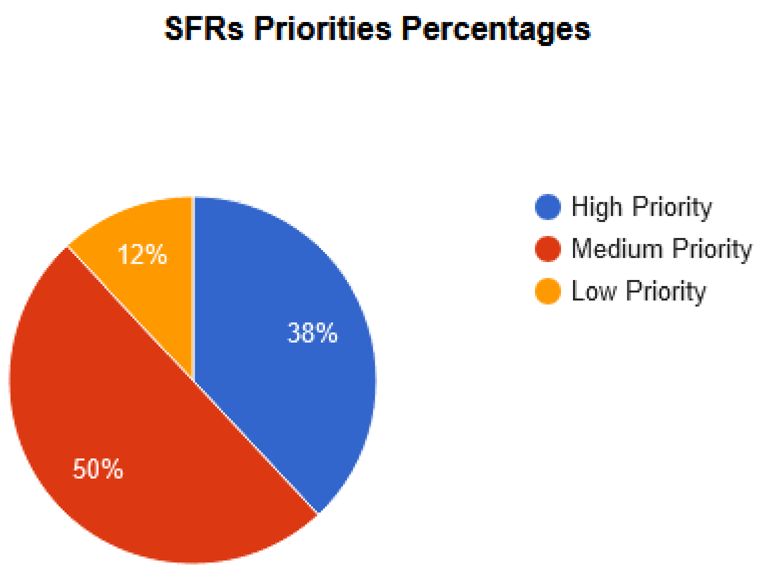


Figure 8 System Functional Requirements Priorities Percentages.

6.3 Handling requirements in MANiBOT use cases

Based on the use case scenarios, the items considered for robotic manipulation, and the pilot site descriptions, specific requirements have been extracted related to the robotic handling of the items involved in the four MANiBOT use cases. Considering the differences between the use cases, the systems will need to operate in different conditions and handle a range of objects. Therefore, the physical dimensions of each pilot, in each use case, need to be factored in. In general, there are two kinds of dimensional restrictions, environmental ones and those of the objects. The environmental restrictions are related to the spaces that the systems will need to move and operate, while the objects in each use case dictate the manipulation requirements. Table 3 to Table 5 report on the dimensions of the spaces and objects in UC1-UC4.

Table 3: ***UC1-Product replenishment in retail stores (MASOUTIS)***

Component Name	Type	Width (cm)	Length (cm)	Height (cm)	Weight (kg)
Corridor	Environment	180	-	-	-
Lower shelf	Supporting surface	40	-	40	-
Retail item	Object to handle	7-45	6-57	3.5-45	0.21-23

For the UC1 “product replenishment in retail stores” addressed by MASOUTIS pilot, the robotic platform will need to move within typical corridors that are found in a supermarket/grocery store. The width of the platform should be less than 180 cm, otherwise it will not be possible to navigate among the shelves. Once the platform reaches its destination, it will need to manipulate objects of various shapes and sizes to refill the shelves. The items can be as small as 6 cm in length and 7 cm in width, such as a box of cereal, or as big as 57 cm in length and 45 cm in width, for example a large can of olive oil. Similarly, depending on the product, the weight can range from 0.21 kg for a pack of paper towels to 23 kg for laundry detergent. It should be noted that larger items will not necessarily be the heaviest, and in fact, they might be fragile (*e.g.*, packs of paper rolls, food, etc.). The final set of retail products that will be considered for the pilot has not been selected yet, though the approach is to select objects with generic shape (such as boxes, bottles) and cover a decent range of different weights. However, it is unlikely that in UC1 large heavy items will be considered (this is better covered in UC2), so the platform's biggest challenge will be to handle small or medium sized, fragile/deformable objects, e.g. packages of pasta, rice etc. The lowest shelf that will need replenishment is at 40 cm height and its width is also at 40 cm. As such, the manipulator needs to be able to go that low and arrange the items inside this limited space. The reach of the robot should be sufficient because of the relatively tight spaces in UC1. Figure 9 shows the pilot site of UC1 (Masoutis supermarket).

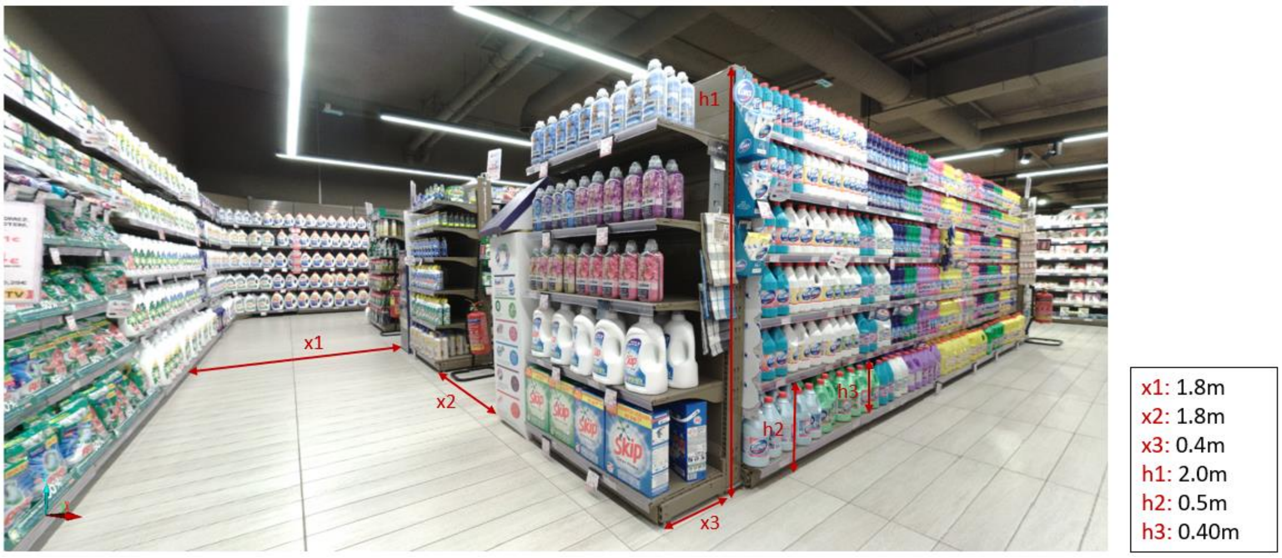


Figure 9 UC1 Pilot Site (Masoutis)

The tight corners that the platform will need to navigate have been taken into account. Even though it will be able to perform in-pace rotations, the robot is designed to be able to navigate the aisles while still having enough space to house all of its components.

Table 4: ***UC2. Product replenishment in retail stores (SDI)***

Component Name	**Type**	**Width (cm)**	**Length (cm)**	**Height (cm)**	**Weight (kg)**
Corridor	Environment	160-240	-	-	-
Lower shelf	Supporting surface	66.6-137.4	37-93.1	-	-
Single retail item	Object to handle	6-15	6-15	5-40	0.15-2.5
Retail item pack	Object to handle	10-60	10-80	5-40	1.3-23

Though UC2 share many similarities with the UC1, there are different specifications and constraints. For the UC2, the corridors have a width between 160cm-240cm, meaning that there will be fluctuations in the space available. The most appropriate design choice is to abide by the minimum specification (160 cm) and, when available, navigate in the extra space to approach the shelves. Similarly, the shelves can have a width that fluctuates between 66.6 cm to 137.4 cm and their length can be between 37 cm and 93.1 cm. The need to have the platform function in the environment without restrictions, drives the decision to design it to function within the smaller edge cases and use the extra space for more agile manipulation. In general, the range that a manipulator should be able to reach in UC2 is greater than in UC1 because of the larger spaces. The larger workspace that the robot will need to reach must be kept into perspective. It can be mitigated by having the navigation algorithm move the platform to an appropriate distance from each shelf, once there, the manipulator will not have an issue to reach the desired item.

Regarding single small items, their width and length will be between 6-15 cm, while their height will start from 5 cm, but it might reach 40 cm. Retail item packs can be much larger, their width will be between 10 cm to 60 cm and their length can start from 10 cm and reach 80 cm. Their height will be between 5 -40cm. The maximum weight of a manipulated object cannot exceed the weights presented in Table 3 and Table 4, however, the final set of item packs considered for the UC2 pilots has not been selected yet. Figure 10 shows the floorplan of UC2 pilot site (SDI).

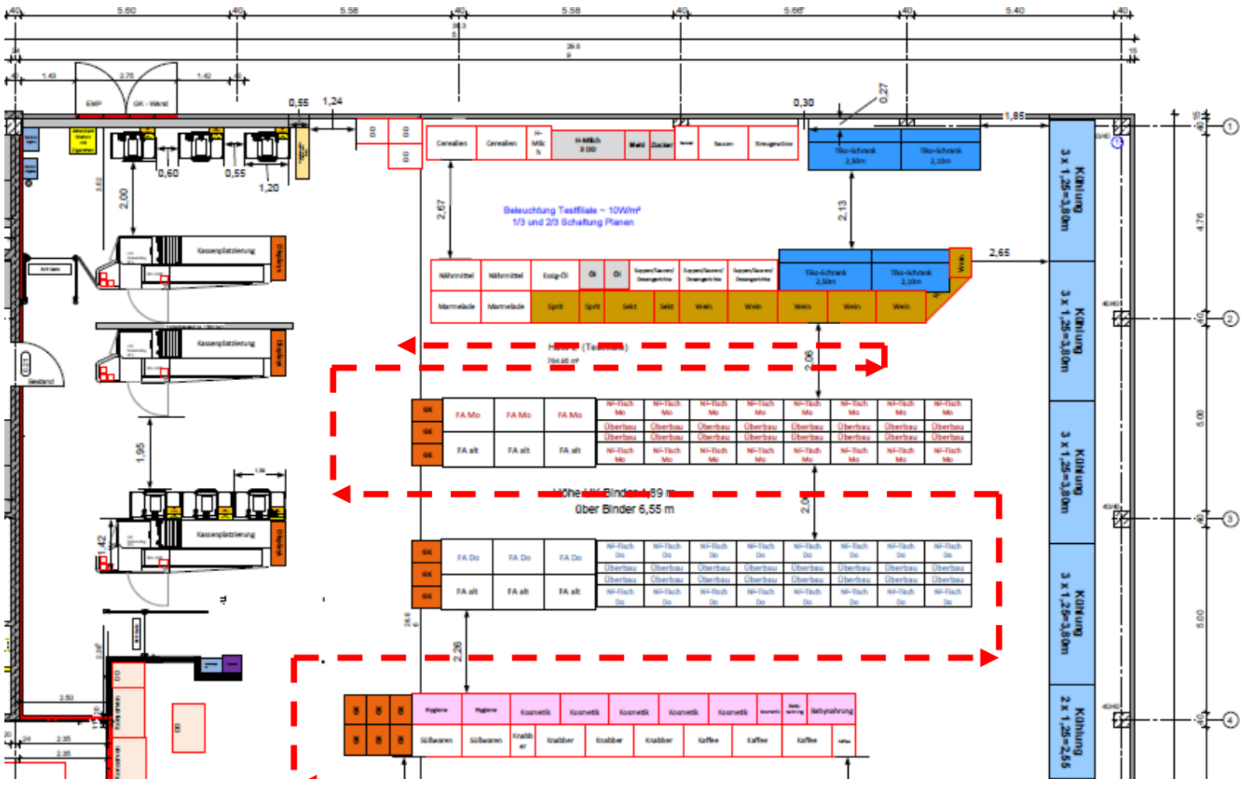


Figure 10 UC2 Pilot Site Floorplan (SDI)

In both UC1 and UC2, the platform needs to be able to navigate in the aisles with ease without obstructing the aisle to other users, such as employees, customers, or even other robots.

Table 5: ***UC3-UC4. Baggage loading and unloading from conveyor belts to carts (FG)***

Component Name	**Type**	**Width (cm)**	**Length (cm)**	**Height (cm)**	**Weight (kg)**
Conveyor belt	Environment	100	-	70	-
Side ramp in front of conveyor	Environment	90	-	16	-

Luggage cart	Supporting surface	150	230	54	-
Luggage item	Object to handle	7.5-45	25-100	5-75	0.5-23

The UC3 and UC4 use cases are considering an airport and the handling of luggage for departure and arrivals, respectively. In UC3 the different suitcases are moving along a conveyor belt that is 100 cm wide and it is 70 cm above the ground. In UC4 the robotic platform will need to move items from a luggage cart that is 150 cm wide, 230 cm long and 54 cm above the floor. The demands for longer reach in those use cases are much higher due to the larger spaces. Indeed, while the previous use cases present a challenge because of the smaller objects that they have, UC3 and UC4 have greater demands regarding the reachable workspace. Though the navigation algorithm will be designed to place the platform at an optimum distance, additional range is required for the manipulators, in order to perform the transfer of the luggage from the conveyor belt to the cart and vice-versa. The airport pilot site (FRAPORT) of UC3-4 is shown in Figure 11.

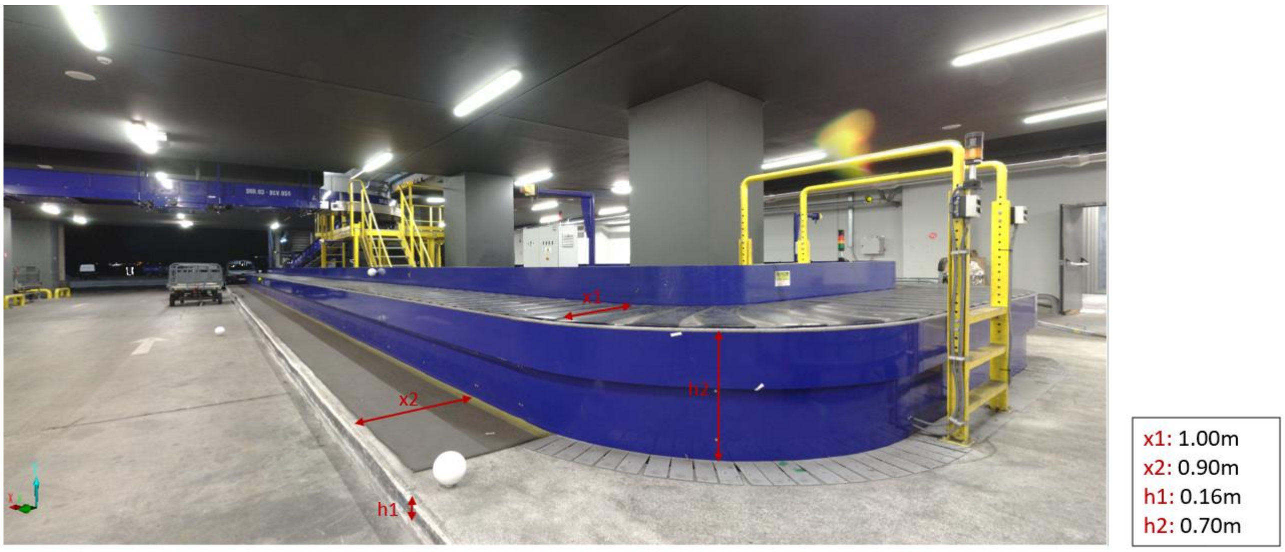


Figure 11 Airport Pilot Site (FRAPORT)

The robot is expected to navigate the dedicated side ramp in front of the conveyor belt that is 90 cm wide. Thus, the width of the MANiBOT mobile platform for the airport use cases should be less than 90 cm, without however compromising effectiveness.

The luggage can have a width from 7.5 cm to 45 cm and their length starts from 25 cm and it can reach 100 cm. Similarly, their height can vary from 5 cm to 75 cm. The weight ranges between 0.5 kg - 23 kg, though that's an estimation since the luggage weight will probably vary much more than the supermarket items. In case of large heavy suitcases, supporting surfaces will be utilised by the robot to avoid lifting the entire load during the transfer.

7 Dynamic Analysis of the System

7.1 Introduction

In this chapter, we present the system's dynamic structure analysis, which is based on the defined use cases which were described in D2.2. The dynamic analysis of the system sheds light on how it operates. In this version, SysML activity diagrams are used to describe the functionality of each use case. An activity diagram provides an overview of each task, depicting the robot's activities and collaboration with the system and operator. The goal is to provide an overall workflow expressed in terms of system components (software), mobile platforms, robotic manipulators, and human supervision.

In the following figure we give a brief overview of the semantics of the activity diagrams and the representation of the basic nodes (Figure 12). An "Initial Node" always starts the activity, and a "Final Node" always finishes it. A "Decision Node" is used when the system or the supervisor has to make the decision to act. For concurrent actions or flows, a "Fork/Join Node" is utilized. An "Action Node" is a representation of an action. Each diagram's primary section depicts specific actions taken by the system or the operator. The corresponding use-case descriptions are used to derive the actions in the diagrams.

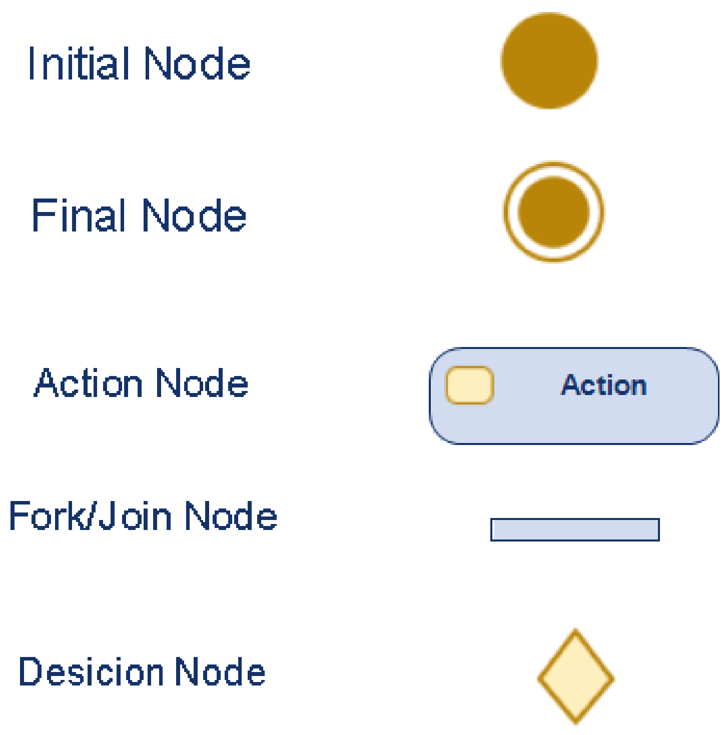


Figure 12 **Basic Nodes used in Activity Diagrams**

7.2 Use-case 1 - Product replenishment in retail stores (MASOUTIS)

7.2.1 Sub-use case 1.1 - Movement from resting position/charging station to the target replenishment cart

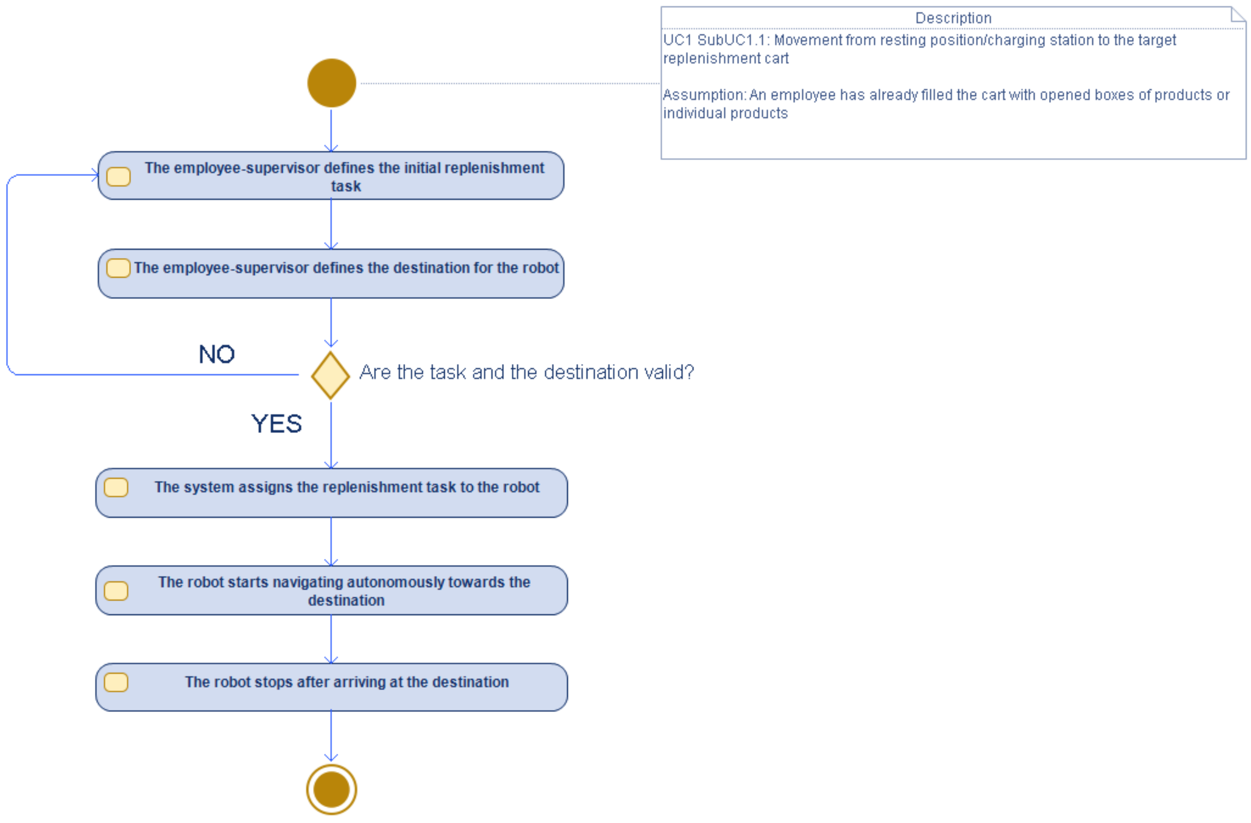

Description
UC1 SubUC1.1: Movement from resting position/charging station to the target replenishment cart
Assumption: An employee has already filled the cart with opened boxes of products or individual products
The employee-supervisor defines the initial replenishment task
The employee-supervisor defines the destination for the robot
NO
Are the task and the destination valid?
YES
The system assigns the replenishment task to the robot
The robot starts navigating autonomously towards the destination
The robot stops after arriving at the destination

Figure 13 Activity diagram of Sub-use case 1.1 Movement from resting position/charging station to the target replenishment cart

7.2.2 Sub-use case 1.2 - Identification of the products in the replenishment cart

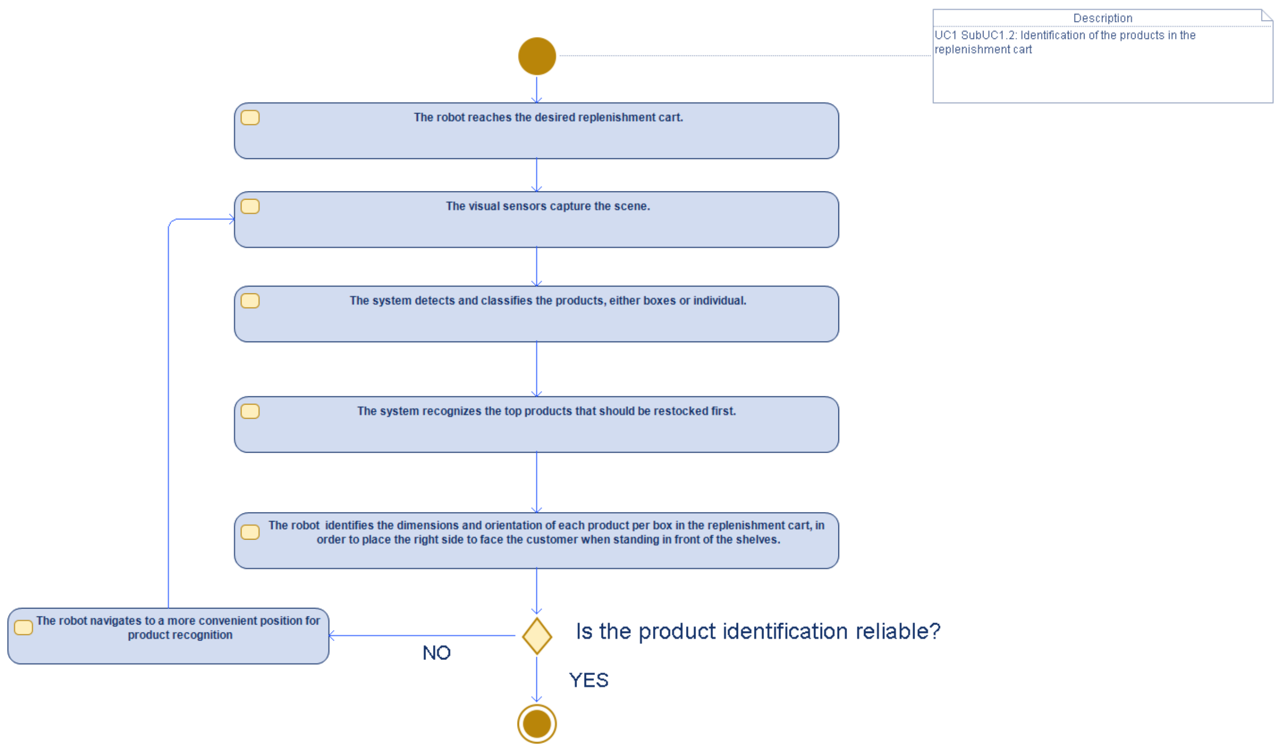

Description
UC1 SubUC1.2: Identification of the products in the replenishment cart
The robot reaches the desired replenishment cart.
The visual sensors capture the scene.
The system detects and classifies the products, either boxes or individual.
The system recognizes the top products that should be restocked first.
The robot identifies the dimensions and orientation of each product per box in the replenishment cart, in order to place the right side to face the customer when standing in front of the shelves.
Is the product identification reliable?
NO
The robot navigates to a more convenient position for product recognition
YES

Figure 14 Activity diagram of Sub-use case 1.2 Identification of the products in the replenishment cart

7.2.3 Sub-use case 1.3 - Identification of the right position for the product on the shelf

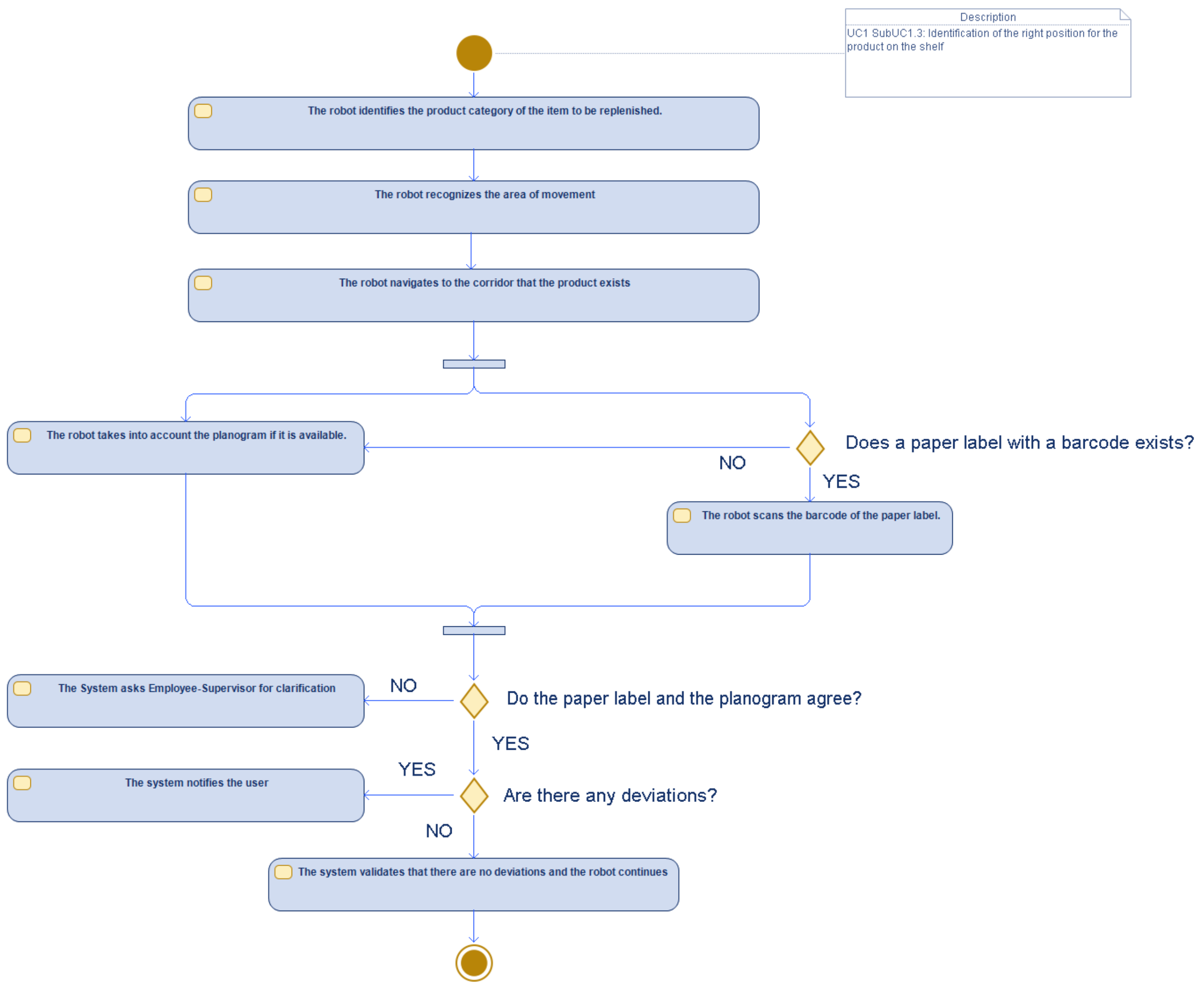

Description
UC1 SubUC1.3: Identification of the right position for the product on the shelf
The robot identifies the product category of the item to be replenished.
The robot recognizes the area of movement
The robot navigates to the corridor that the product exists
The robot takes into account the planogram if it is available.
Does a paper label with a barcode exists?
NO
YES
The robot scans the barcode of the paper label.
The System asks Employee-Supervisor for clarification
NO
Do the paper label and the planogram agree?
YES
The system notifies the user
YES
Are there any deviations?
NO
The system validates that there are no deviations and the robot continues

Figure 15 Activity diagram of Sub-use case 1.3 Identification of the right position for the product on the shelf

7.2.4 Sub-use case 1.4 - Replenishment of the shelves

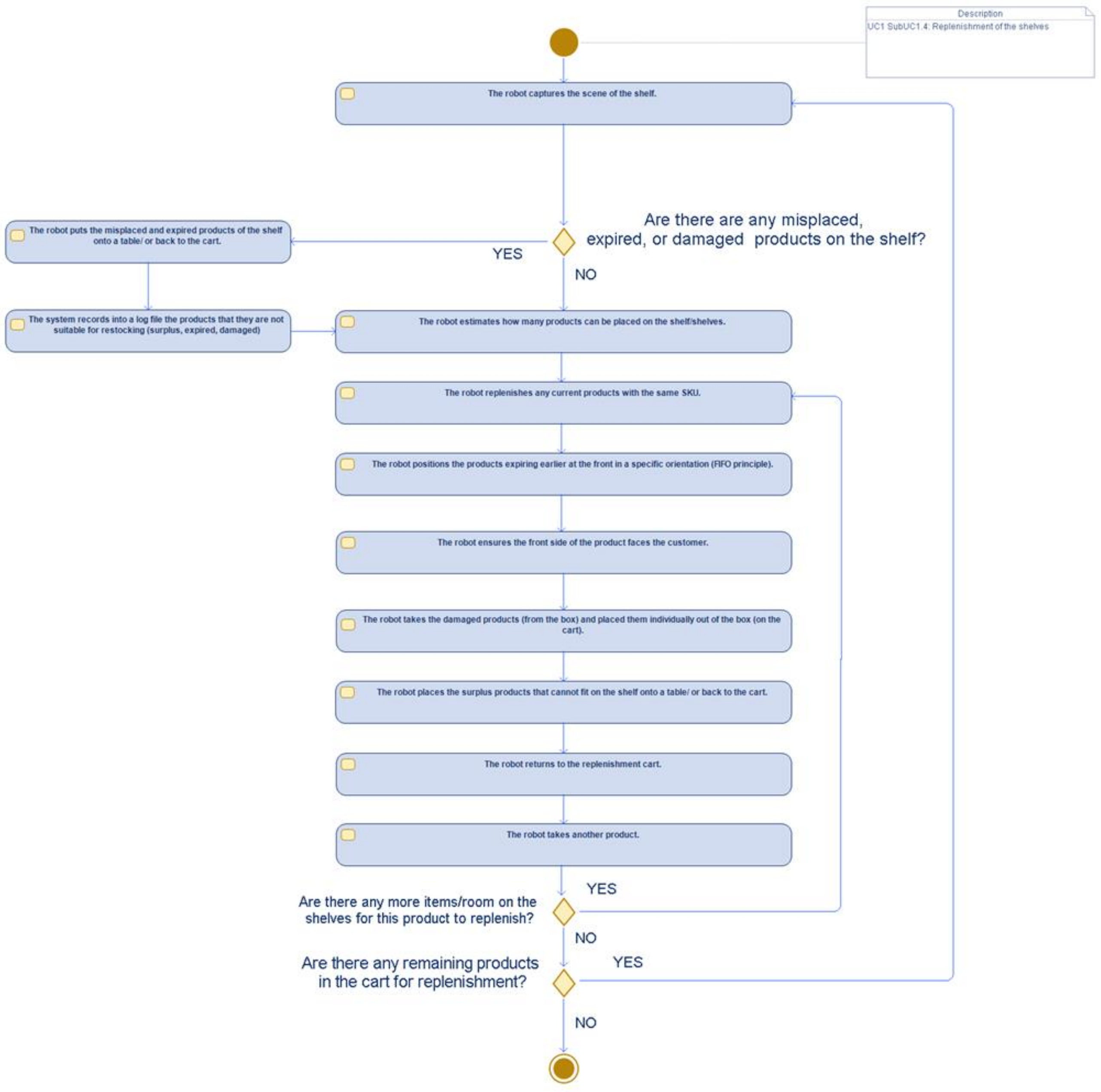

Description
UC1 SubUC1.4: Replenishment of the shelves
The robot captures the scene of the shelf.
Are there are any misplaced, expired, or damaged products on the shelf?
YES
NO
The robot puts the misplaced and expired products of the shelf onto a table/ or back to the cart.
The system records into a log file the products that they are not suitable for restocking (surplus, expired, damaged)
The robot estimates how many products can be placed on the shelf/shelves.
The robot replenishes any current products with the same SKU.
The robot positions the products expiring earlier at the front in a specific orientation (FIFO principle).
The robot ensures the front side of the product faces the customer.
The robot takes the damaged products (from the box) and placed them individually out of the box (on the cart).
The robot places the surplus products that cannot fit on the shelf onto a table/ or back to the cart.
The robot returns to the replenishment cart.
The robot takes another product.
YES
Are there any more items/room on the shelves for this product to replenish?
NO
Are there any remaining products in the cart for replenishment?
YES
NO

Figure 16 Activity diagram of Sub-use case 1.4 Replenishment of the shelves

7.2.5 Sub-use case 1.5 - Movement back to the resting position/charging station

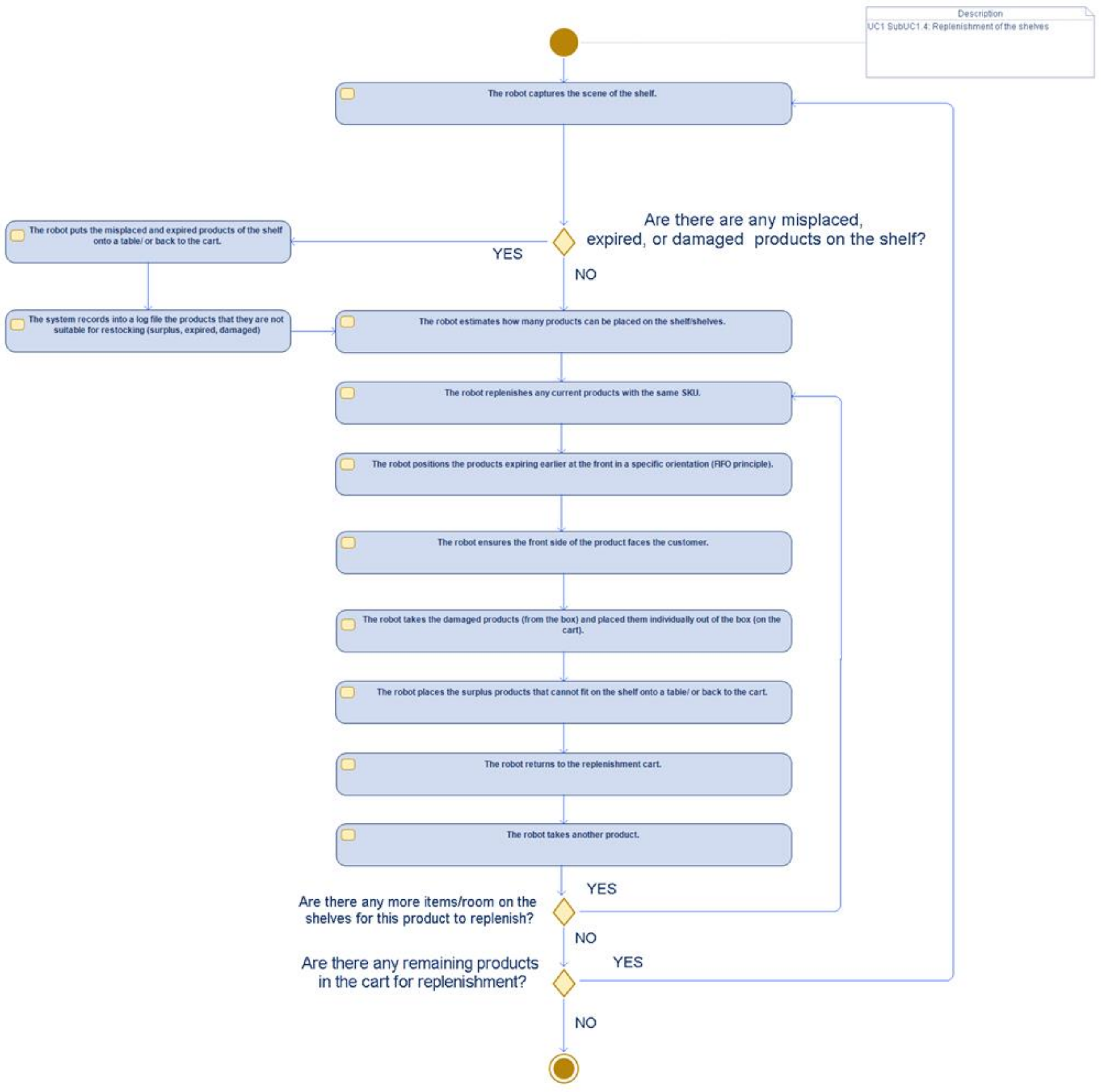

Description
UC1 SubUC1.4: Replenishment of the shelves
The robot captures the scene of the shelf.
Are there are any misplaced, expired, or damaged products on the shelf?
YES
NO
The robot puts the misplaced and expired products of the shelf onto a table/ or back to the cart.
The system records into a log file the products that they are not suitable for restocking (surplus, expired, damaged)
The robot estimates how many products can be placed on the shelf/shelves.
The robot replenishes any current products with the same SKU.
The robot positions the products expiring earlier at the front in a specific orientation (FIFO principle).
The robot ensures the front side of the product faces the customer.
The robot takes the damaged products (from the box) and placed them individually out of the box (on the cart).
The robot places the surplus products that cannot fit on the shelf onto a table/ or back to the cart.
The robot returns to the replenishment cart.
The robot takes another product.
YES
Are there any more items/room on the shelves for this product to replenish?
NO
Are there any remaining products in the cart for replenishment?
YES
NO

Figure 17 Activity diagram of Sub-use case 1.5 Movement back to the resting position/charging station

7.3 Use-case 2 - Product replenishment in retail stores (SDI)

7.3.1 Sub-use case 2.1 - Movement from resting position/charging station to the target destination for replenishment

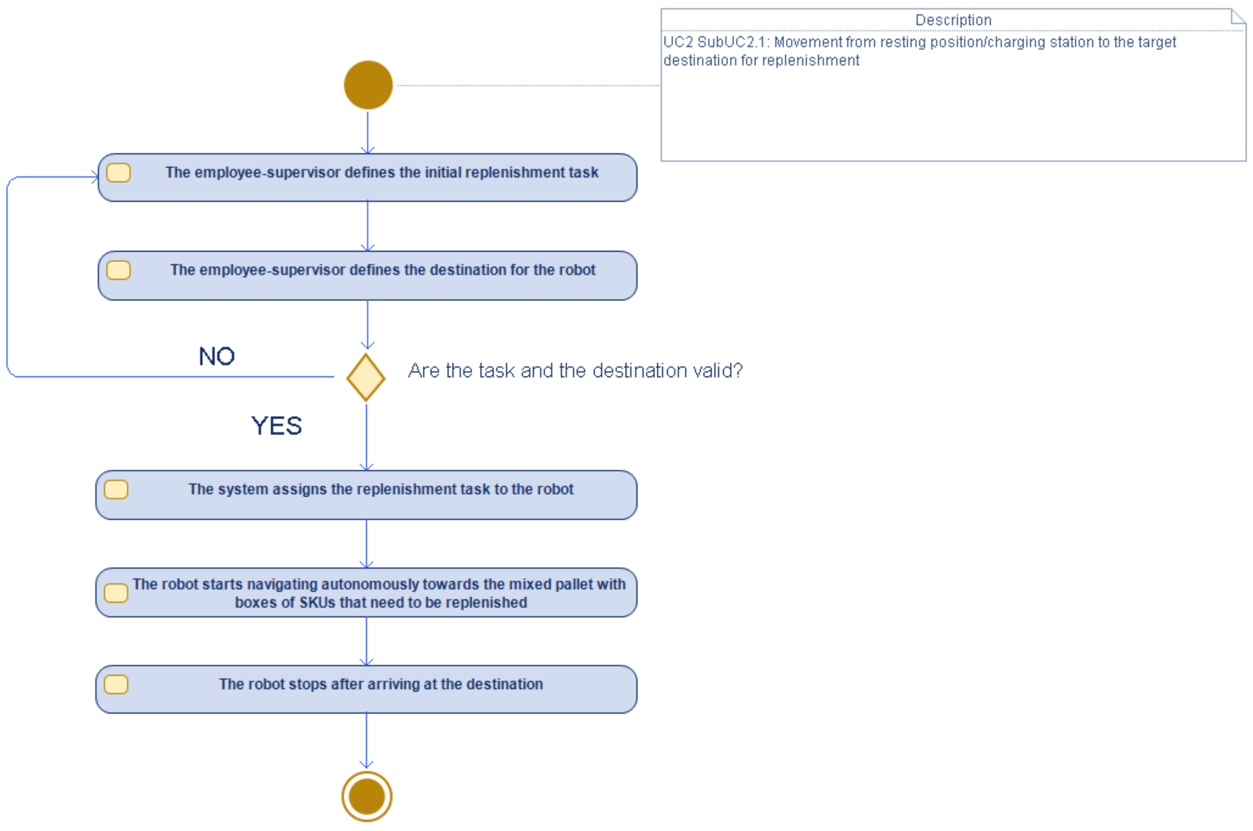

Description
UC2 SubUC2.1: Movement from resting position/charging station to the target destination for replenishment
The employee-supervisor defines the initial replenishment task
The employee-supervisor defines the destination for the robot
NO
Are the task and the destination valid?
YES
The system assigns the replenishment task to the robot
The robot starts navigating autonomously towards the mixed pallet with boxes of SKUs that need to be replenished
The robot stops after arriving at the destination

Figure 18 Activity diagram of Sub-use case 2.1 Movement from resting position/charging station to the target destination for replenishment

7.3.2 Sub-use case 2.2 - Identification of the products on mixed pallet

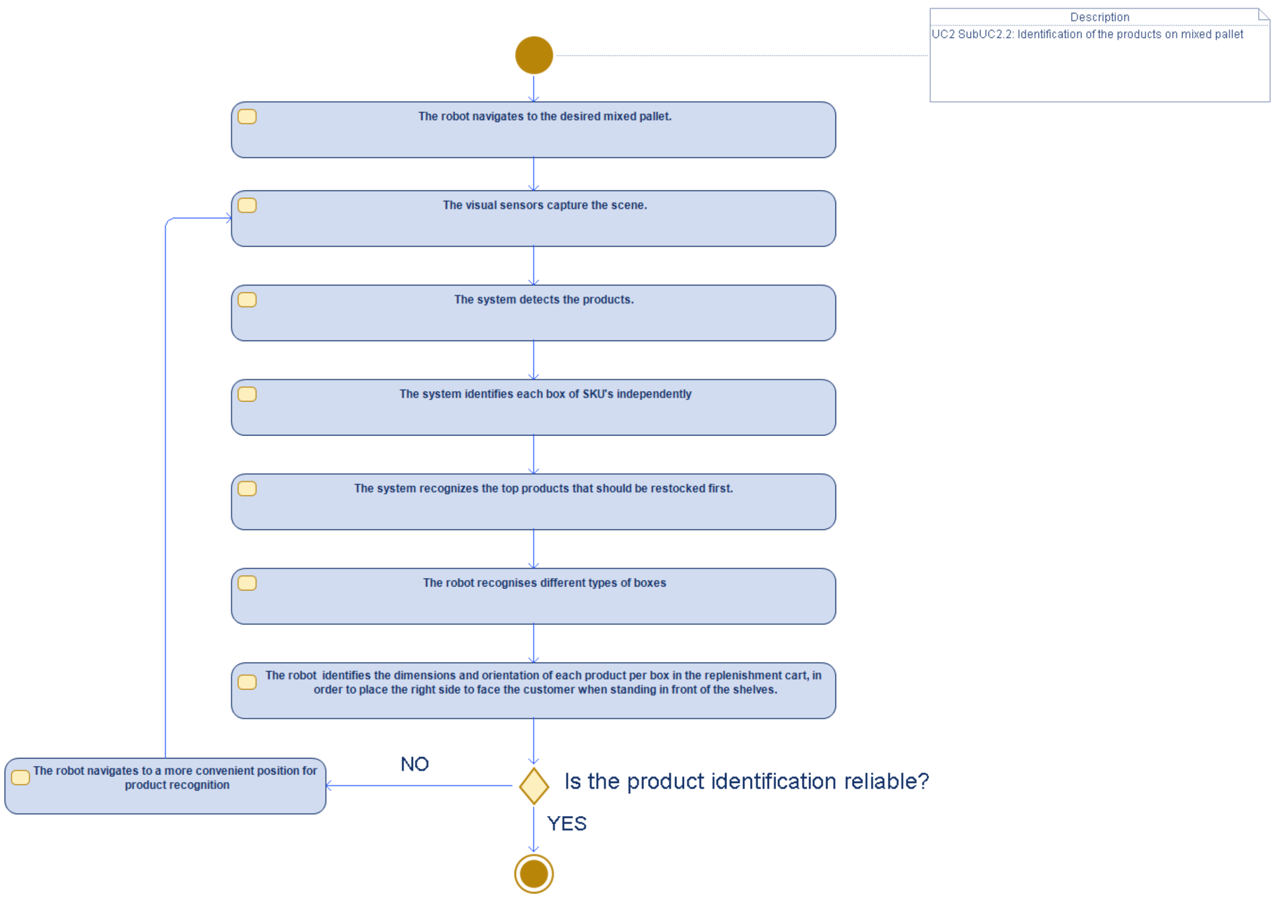

Description
UC2 SubUC2.2: Identification of the products on mixed pallet
The robot navigates to the desired mixed pallet.
The visual sensors capture the scene.
The system detects the products.
The system identifies each box of SKU's independently
The system recognizes the top products that should be restocked first.
The robot recognises different types of boxes
The robot identifies the dimensions and orientation of each product per box in the replenishment cart, in order to place the right side to face the customer when standing in front of the shelves.
Is the product identification reliable?
NO
The robot navigates to a more convenient position for product recognition
YES

Figure 19 Activity diagram of Sub-use case 2.2 Identification of the products on mixed pallet

7.3.3 Sub-use case 2.3 - Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)

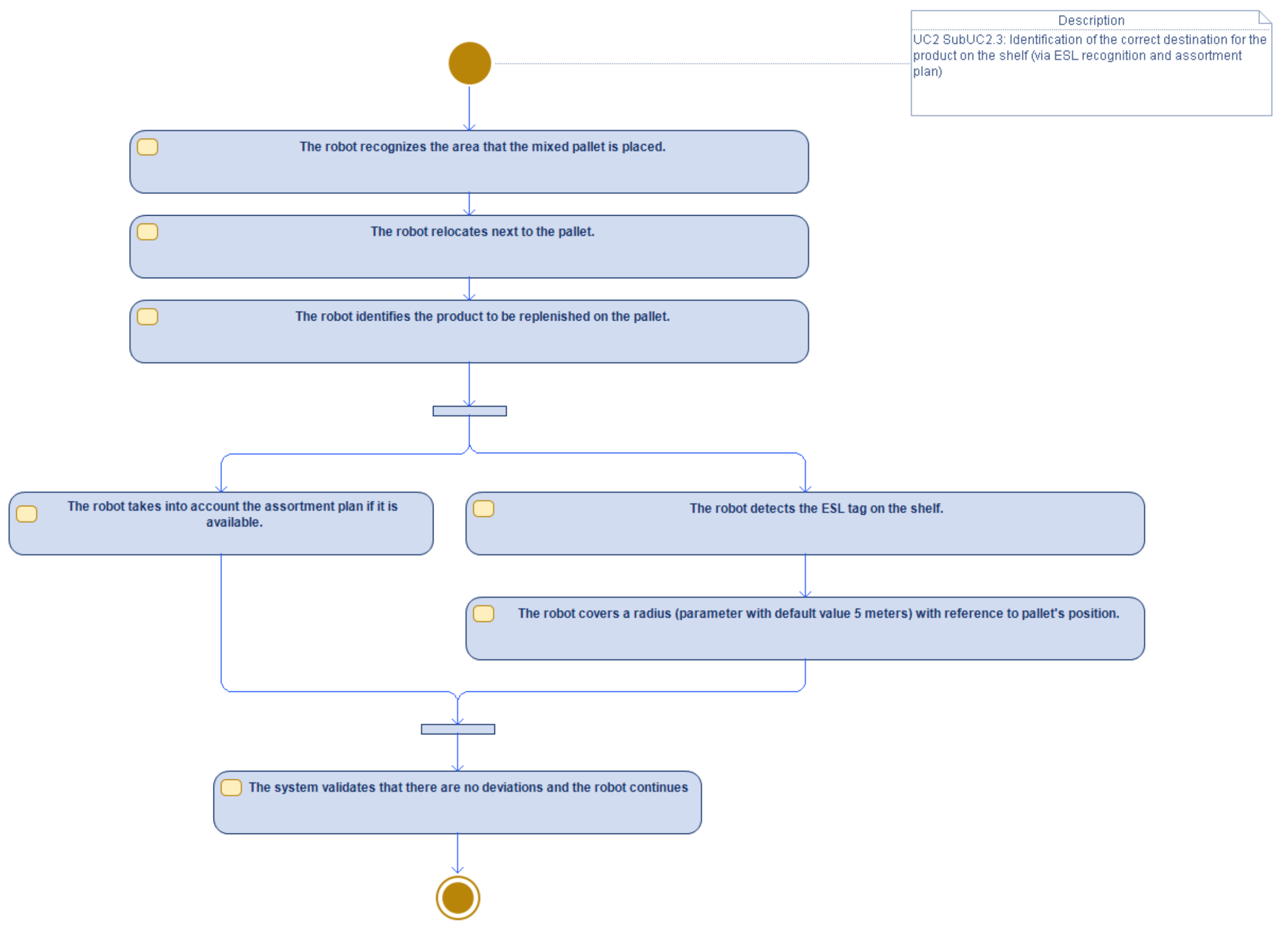

Description
UC2 SubUC2.3: Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)
The robot recognizes the area that the mixed pallet is placed.
The robot relocates next to the pallet.
The robot identifies the product to be replenished on the pallet.
The robot takes into account the assortment plan if it is available.
The robot detects the ESL tag on the shelf.
The robot covers a radius (parameter with default value 5 meters) with reference to pallet's position.
The system validates that there are no deviations and the robot continues

Figure 20 Activity diagram of Sub-use case 2.3 Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)

7.3.4 Sub-use case 2.4 - Preparation of the products before replenishment

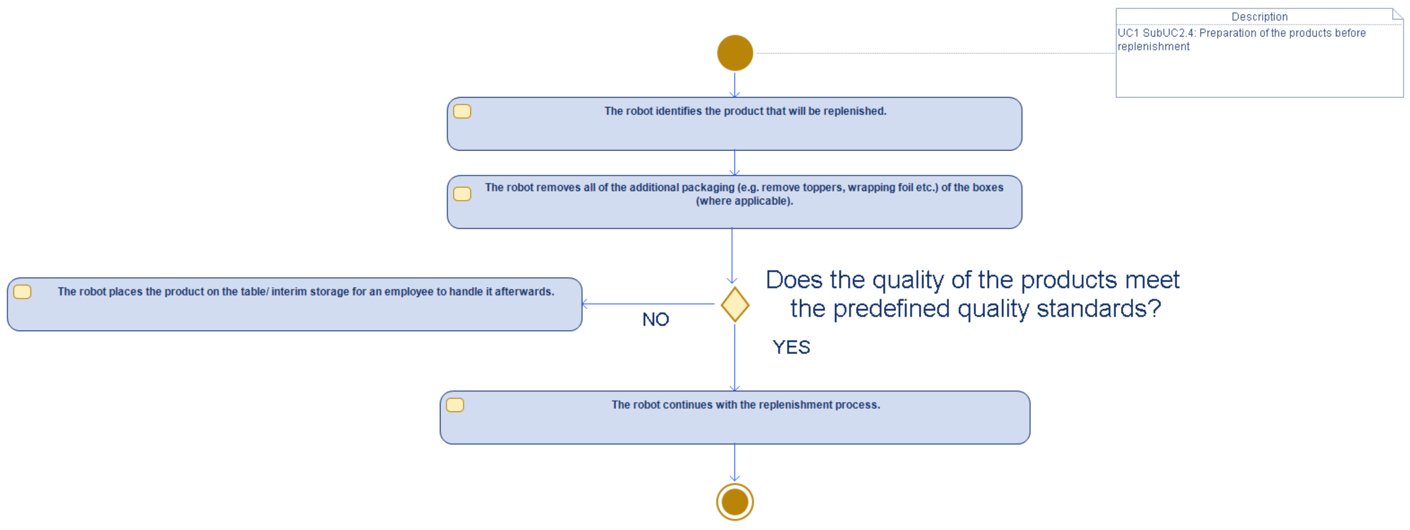

Description
UC1 SubUC2.4: Preparation of the products before replenishment
The robot identifies the product that will be replenished.
The robot removes all of the additional packaging (e.g. remove toppers, wrapping foil etc.) of the boxes (where applicable).
The robot places the product on the table/ interim storage for an employee to handle it afterwards.
NO
Does the quality of the products meet the predefined quality standards?
YES
The robot continues with the replenishment process.

Figure 21 Activity diagram of Sub-use case 2.4 Preparation of the products before replenishment

7.3.5 Sub-use case 2.5 - Product replenishment (applying FIFO principle and overstock management)

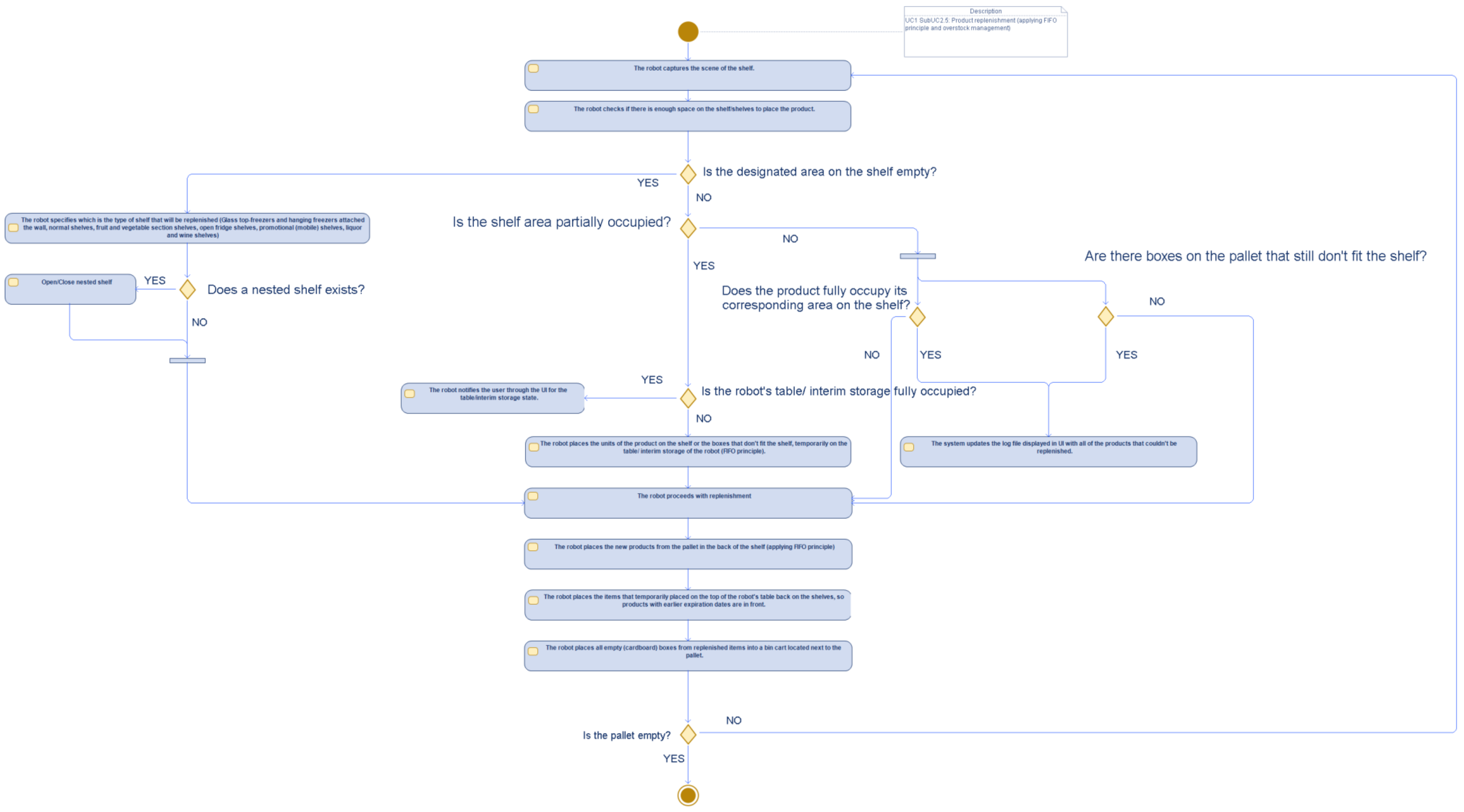

Description
UC1 SubUC2.5: Product replenishment (applying FIFO principle and overstock management)
The robot captures the scene of the shelf.
The robot checks if there is enough space on the shelf/shelves to place the product.
Is the designated area on the shelf empty?
YES
NO
The robot specifies which is the type of shelf that will be replenished (Glass top-freezers and hanging freezers attached the wall, normal shelves, fruit and vegetable section shelves, open fridge shelves, promotional (mobile) shelves, liquor and wine shelves)
Is the shelf area partially occupied?
NO
YES
Are there boxes on the pallet that still don't fit the shelf?
Open/Close nested shelf
YES
Does a nested shelf exists?
NO
Does the product fully occupy its corresponding area on the shelf?
NO
NO
YES
YES
YES
The robot notifies the user through the UI for the table/interim storage state.
Is the robot's table/ interim storage fully occupied?
NO
The robot places the units of the product on the shelf or the boxes that don't fit the shelf, temporarily on the table/ interim storage of the robot (FIFO principle).
The system updates the log file displayed in UI with all of the products that couldn't be replenished.
The robot proceeds with replenishment
The robot places the new products from the pallet in the back of the shelf (applying FIFO principle)
The robot places the items that temporarily placed on the top of the robot's table back on the shelves, so products with earlier expiration dates are in front.
The robot places all empty (cardboard) boxes from replenished items into a bin cart located next to the pallet.
NO
Is the pallet empty?
YES

Figure 22 Activity diagram of Sub-use case 2.5 Product replenishment (applying FIFO principle and overstock management)

7.3.6 Sub-use case 2.6 - Movement back to the resting position/charging

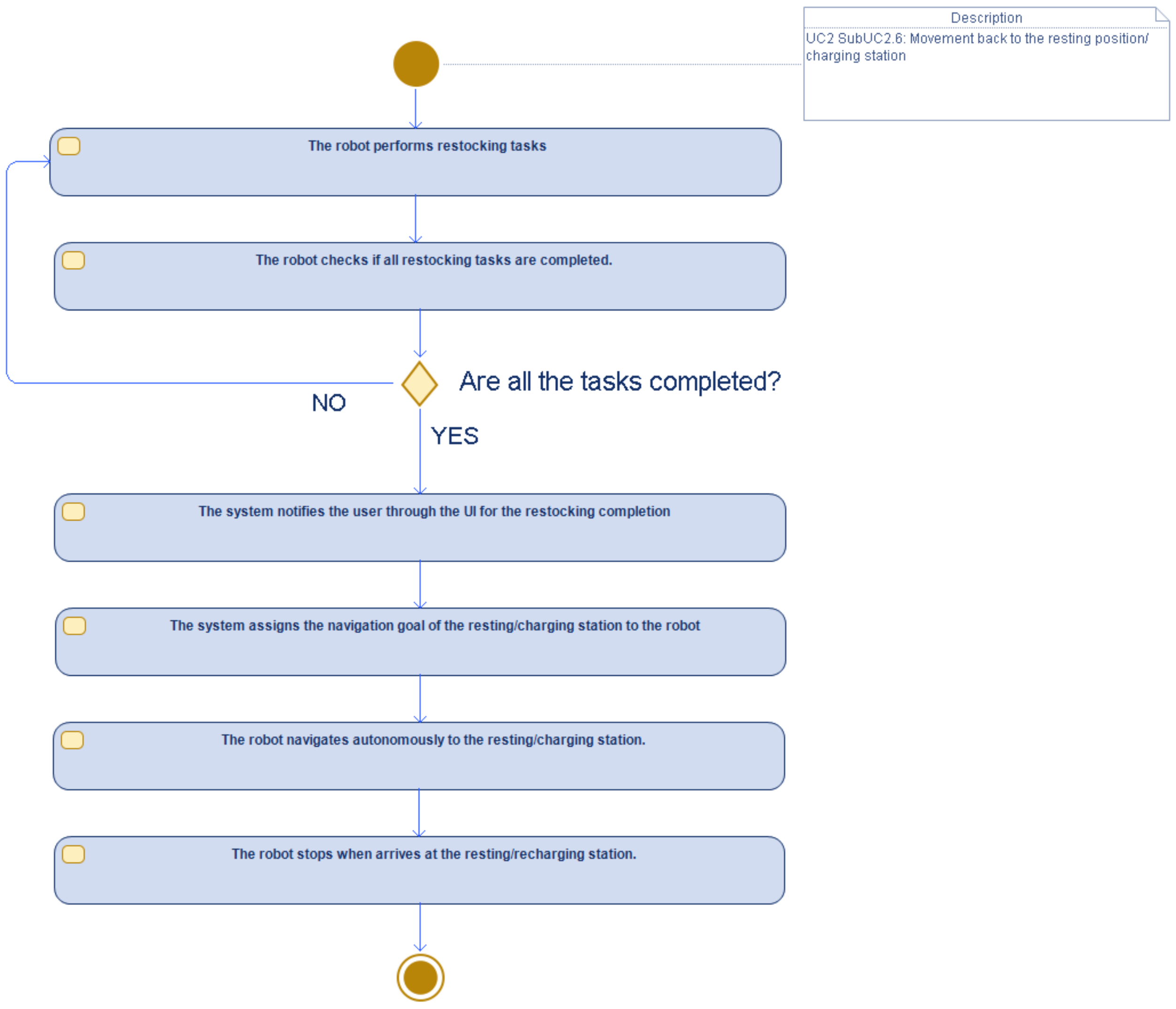

Description
UC2 SubUC2.6: Movement back to the resting position/ charging station
The robot performs restocking tasks
The robot checks if all restocking tasks are completed.
Are all the tasks completed?
NO
YES
The system notifies the user through the UI for the restocking completion
The system assigns the navigation goal of the resting/charging station to the robot
The robot navigates autonomously to the resting/charging station.
The robot stops when arrives at the resting/recharging station.

Figure 23 Activity diagram of Sub-use case 2.6 Movement back to the resting position/charging

7.4 Use-case 3 - Baggage loading and unloading from conveyor belts to carts (FG)

7.4.1 Sub-use case 3.1 - Preparing for loading

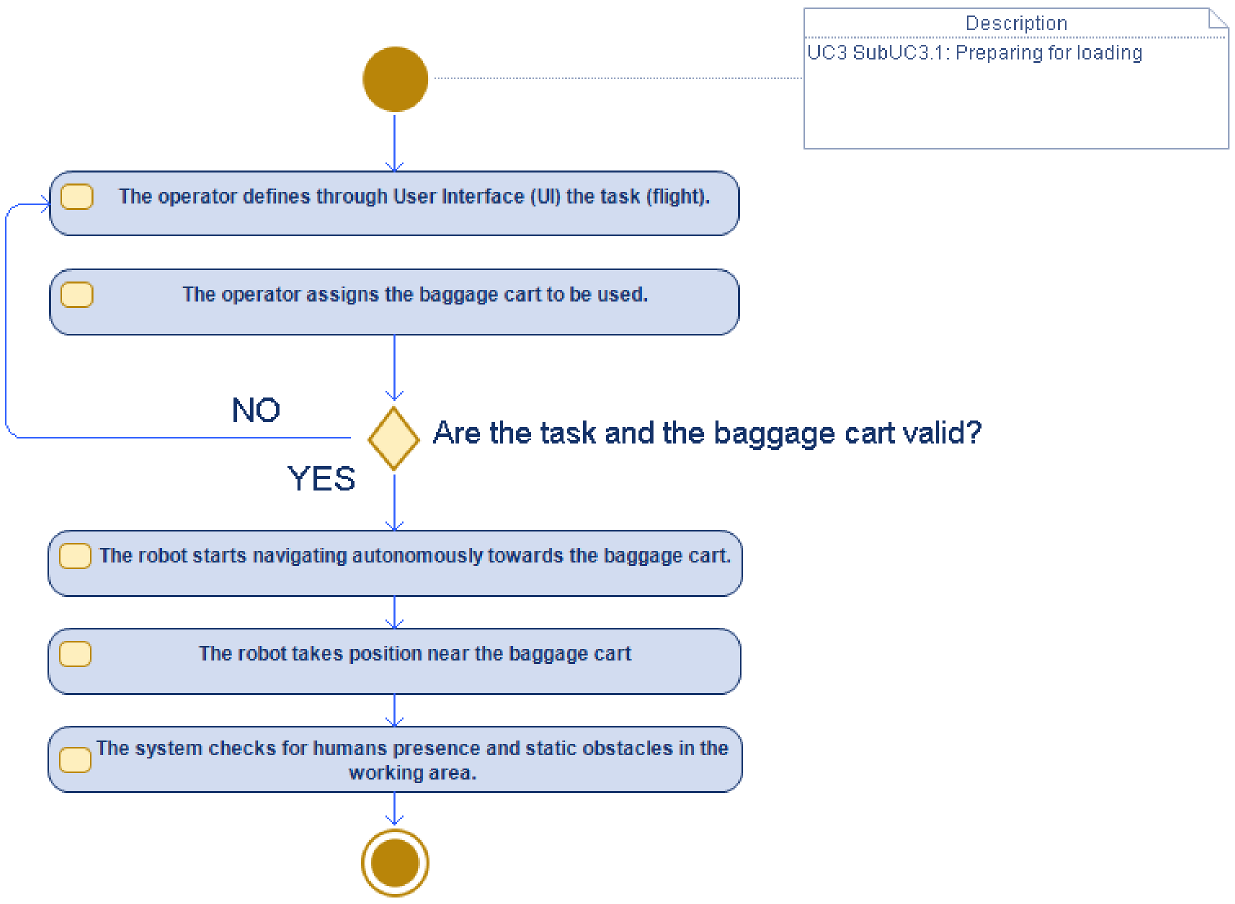

Description
UC3 SubUC3.1: Preparing for loading
The operator defines through User Interface (UI) the task (flight).
The operator assigns the baggage cart to be used.
NO
Are the task and the baggage cart valid?
YES
The robot starts navigating autonomously towards the baggage cart.
The robot takes position near the baggage cart
The system checks for humans presence and static obstacles in the working area.

Figure 24 Activity diagram of Sub-use case 3.1 Preparing for loading

7.4.2 Sub-use case 3.2 - Recognizing Baggage for assigned flight

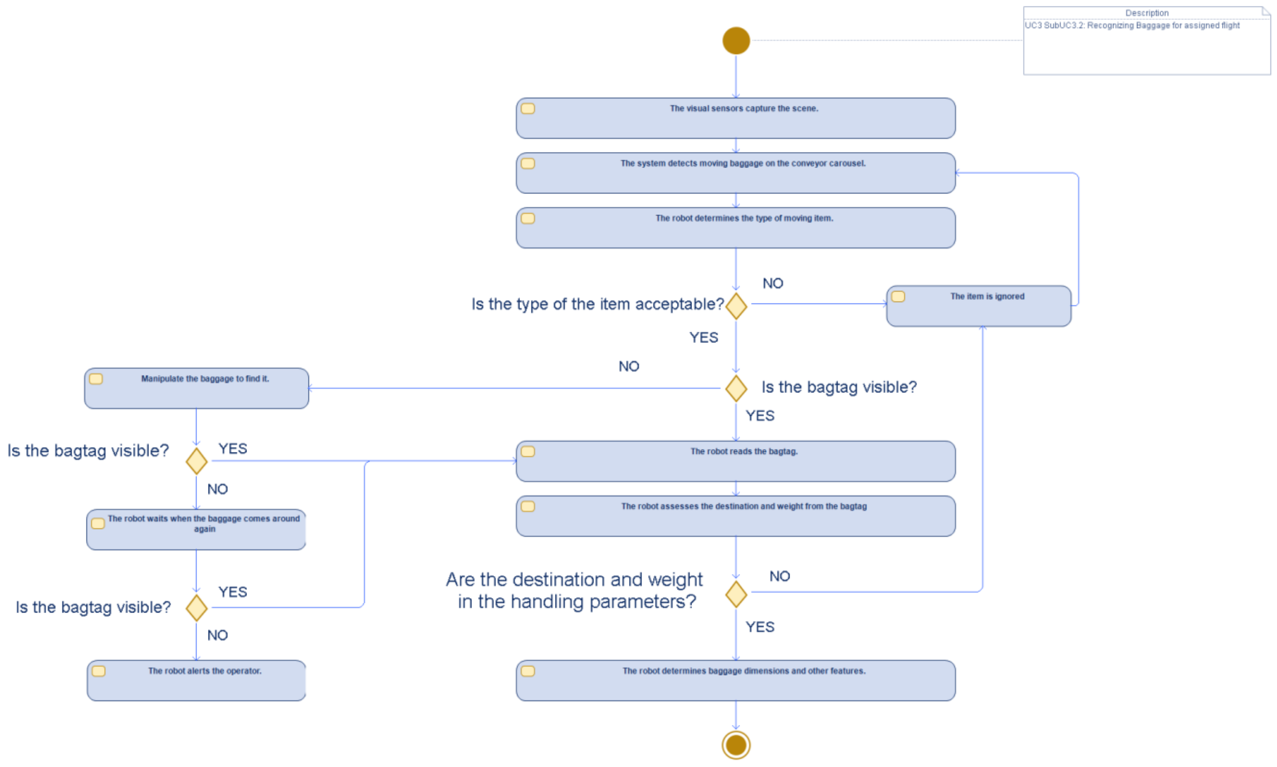

Description
UC3 SubUC3.2: Recognizing Baggage for assigned flight
The visual sensors capture the scene.
The system detects moving baggage on the conveyor carousel.
The robot determines the type of moving item.
Is the type of the item acceptable?
NO
The item is ignored
YES
NO
Is the bagtag visible?
YES
Manipulate the baggage to find it.
Is the bagtag visible?
YES
NO
The robot reads the bagtag.
The robot waits when the baggage comes around again
The robot assesses the destination and weight from the bagtag
YES
Is the bagtag visible?
NO
Are the destination and weight in the handling parameters?
NO
YES
The robot alerts the operator.
The robot determines baggage dimensions and other features.

Figure 25 Activity diagram of Sub-use case 3.2 Recognizing Baggage for assigned flight

7.4.3 Sub-use case 3.3 - Loading of allocated baggage on the cart

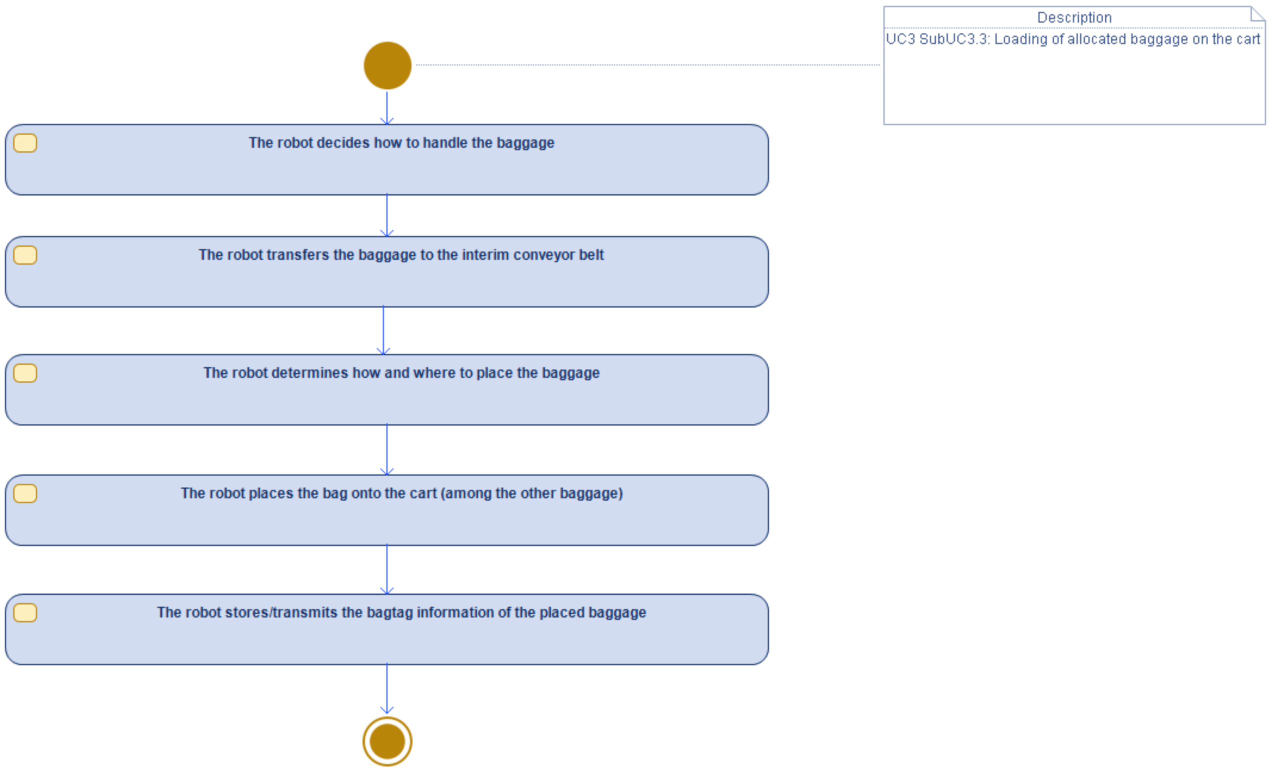

Description
UC3 SubUC3.3: Loading of allocated baggage on the cart
The robot decides how to handle the baggage
The robot transfers the baggage to the interim conveyor belt
The robot determines how and where to place the baggage
The robot places the bag onto the cart (among the other baggage)
The robot stores/transmits the bagtag information of the placed baggage

Figure 26 Activity diagram of Sub-use case 3.3 Loading of allocated baggage on the cart

7.4.4 Sub-use cases 3.2 – 3.3 - Concatenation

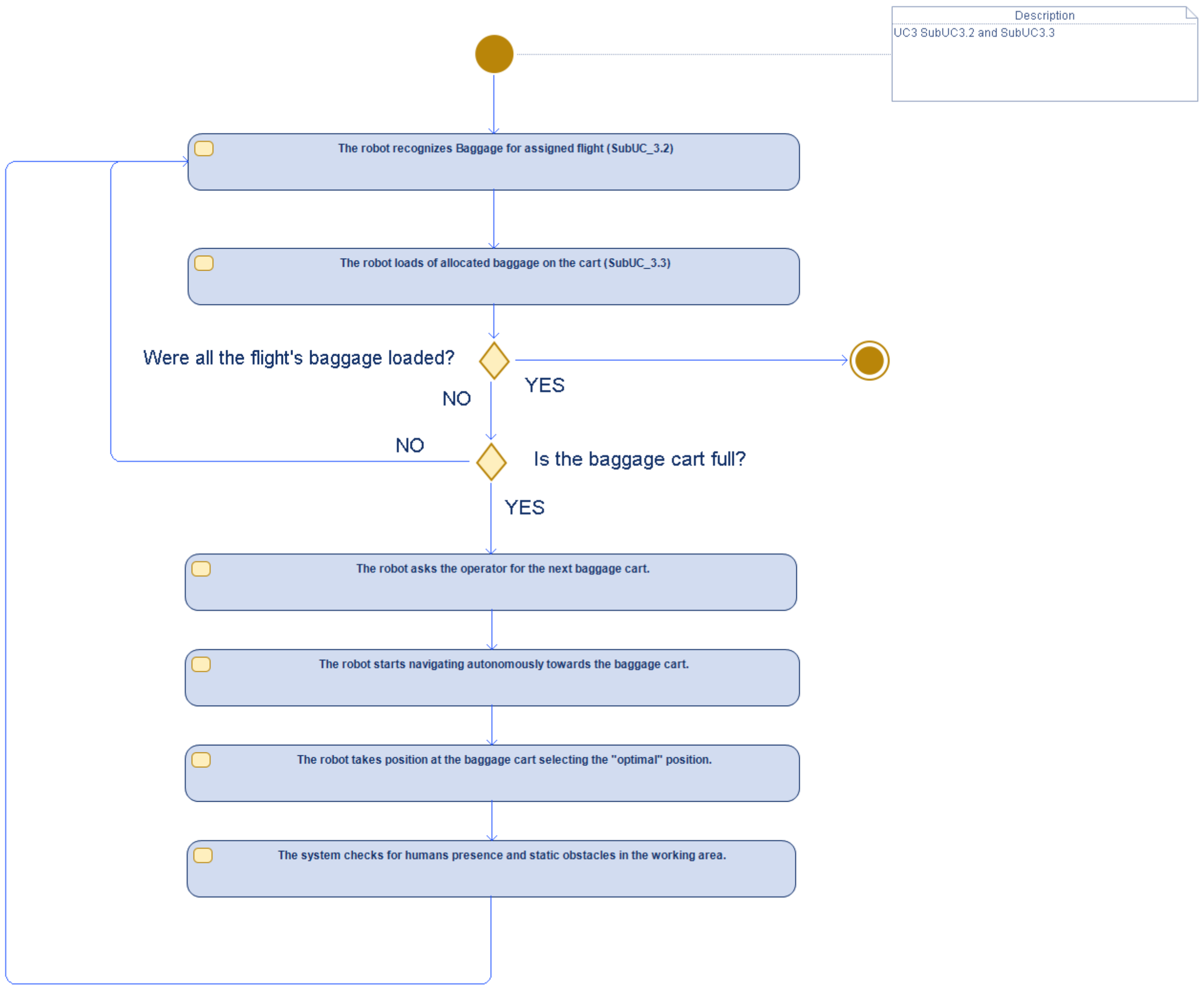

Description
UC3 SubUC3.2 and SubUC3.3
The robot recognizes Baggage for assigned flight (SubUC_3.2)
The robot loads of allocated baggage on the cart (SubUC_3.3)
Were all the flight's baggage loaded?
YES
NO
NO
Is the baggage cart full?
YES
The robot asks the operator for the next baggage cart.
The robot starts navigating autonomously towards the baggage cart.
The robot takes position at the baggage cart selecting the "optimal" position.
The system checks for humans presence and static obstacles in the working area.

Figure 27 Activity diagram of the concatenation of Sub-use cases 3.2 and 3.3

7.4.5 Sub-use case 3.4 - Movement back to resting position/charging point

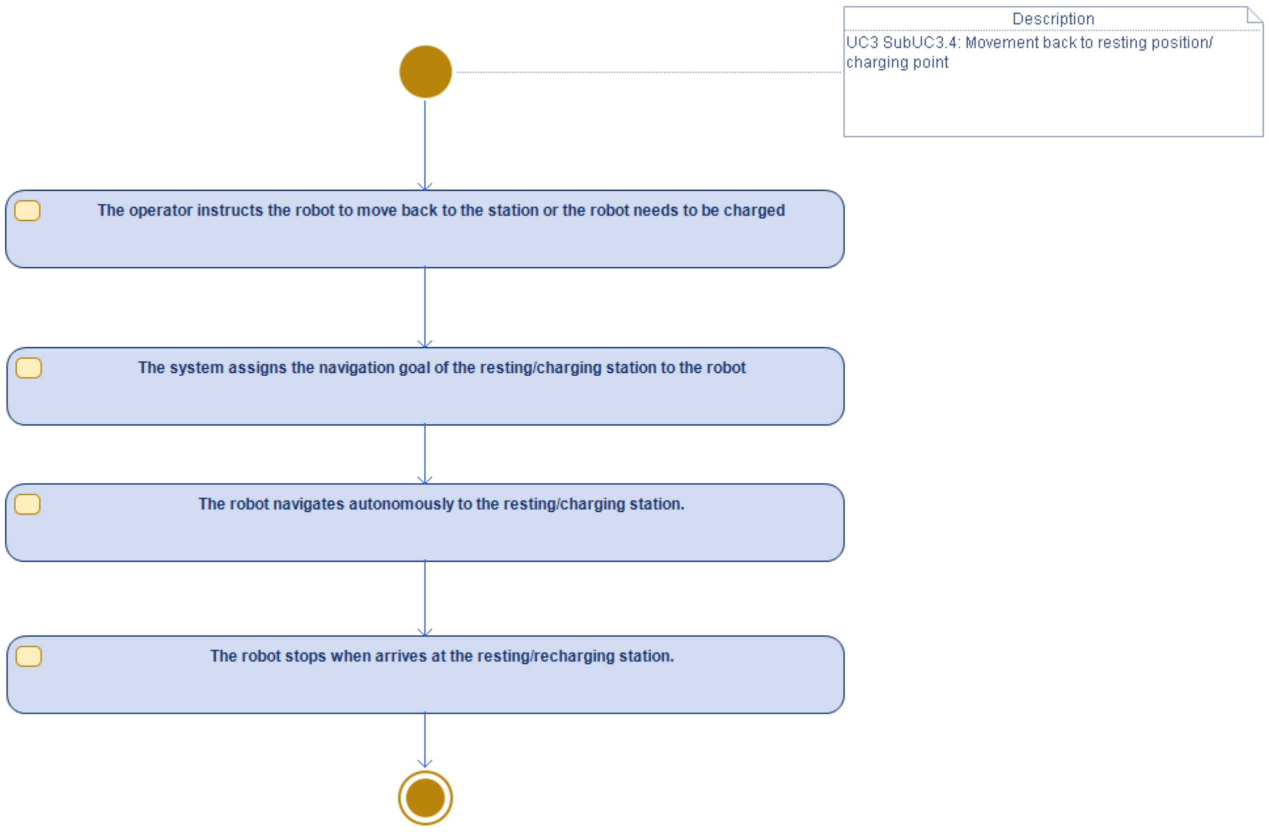

Description
UC3 SubUC3.4: Movement back to resting position/ charging point
The operator instructs the robot to move back to the station or the robot needs to be charged
The system assigns the navigation goal of the resting/charging station to the robot
The robot navigates autonomously to the resting/charging station.
The robot stops when arrives at the resting/recharging station.

Figure 28 Activity diagram of Sub-use case 3.4 Movement back to resting position/charging point

7.5 Use-case 4 - Baggage loading and unloading from carts to conveyor belts (FG)

7.5.1 Sub-use case 4.1 - Movement from resting position/charging station to the target position

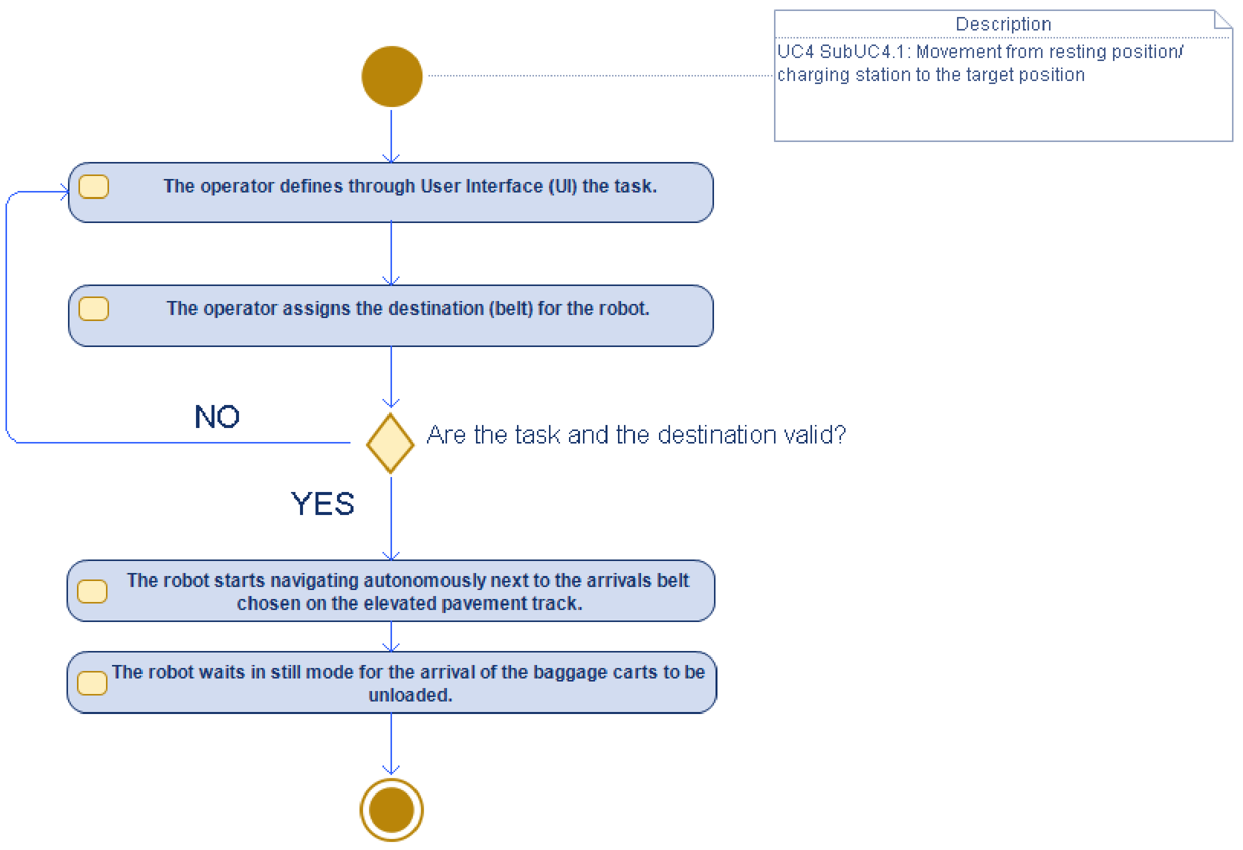

Description
UC4 SubUC4.1: Movement from resting position/ charging station to the target position
The operator defines through User Interface (UI) the task.
The operator assigns the destination (belt) for the robot.
NO
Are the task and the destination valid?
YES
The robot starts navigating autonomously next to the arrivals belt chosen on the elevated pavement track.
The robot waits in still mode for the arrival of the baggage carts to be unloaded.

Figure 29 Activity diagram of Sub-use case 4.1 Movement from resting position/charging station to the target position

7.5.2 Sub-use case 4.2 - 4.3 - 4.4 – Concatenation

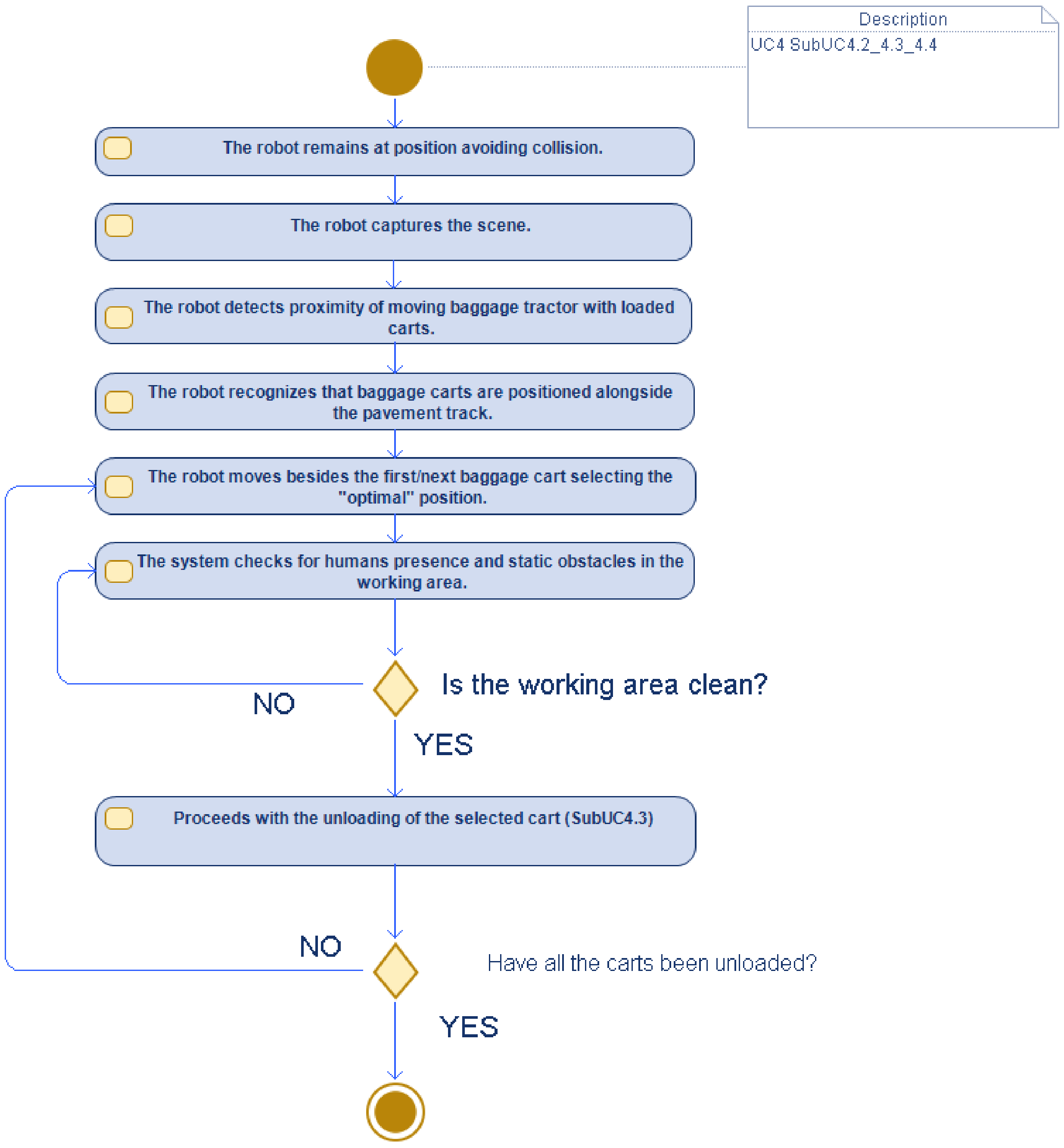

Description
UC4 SubUC4.2_4.3_4.4
The robot remains at position avoiding collision.
The robot captures the scene.
The robot detects proximity of moving baggage tractor with loaded carts.
The robot recognizes that baggage carts are positioned alongside the pavement track.
The robot moves besides the first/next baggage cart selecting the "optimal" position.
The system checks for humans presence and static obstacles in the working area.
Is the working area clean?
NO
YES
Proceeds with the unloading of the selected cart (SubUC4.3)
NO
Have all the carts been unloaded?
YES

Figure 30 Activity diagram of the concatenation of Sub-use cases 4.2 - 4.3 - 4.4

7.5.3 Sub-use case 4.3 - Unloading from Cart to Conveyor belt

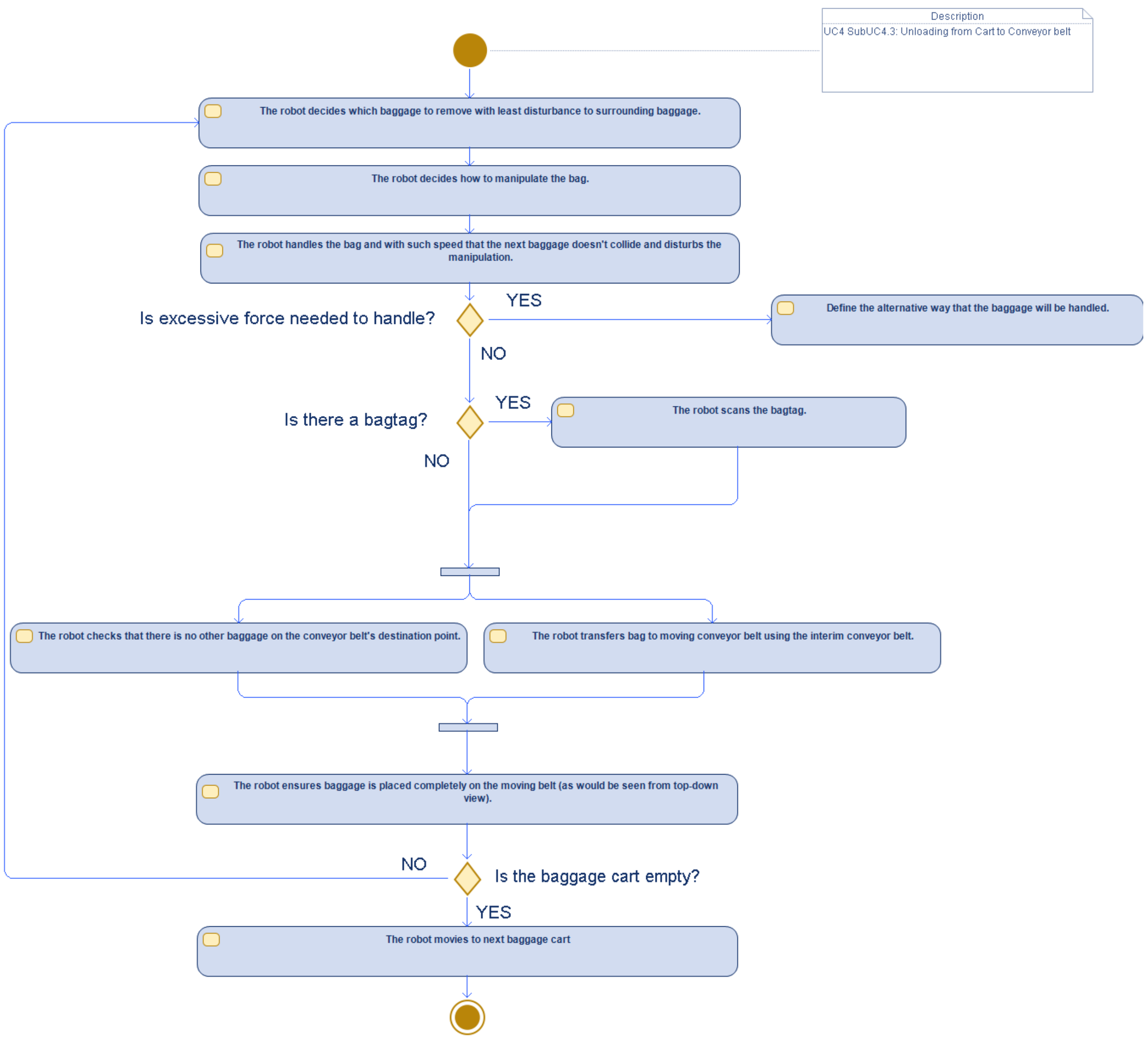

Description
UC4 SubUC4.3: Unloading from Cart to Conveyor belt
The robot decides which baggage to remove with least disturbance to surrounding baggage.
The robot decides how to manipulate the bag.
The robot handles the bag and with such speed that the next baggage doesn't collide and disturbs the manipulation.
Is excessive force needed to handle?
YES
Define the alternative way that the baggage will be handled.
NO
Is there a bagtag?
YES
The robot scans the bagtag.
NO
The robot checks that there is no other baggage on the conveyor belt's destination point.
The robot transfers bag to moving conveyor belt using the interim conveyor belt.
The robot ensures baggage is placed completely on the moving belt (as would be seen from top-down view).
NO
Is the baggage cart empty?
YES
The robot movies to next baggage cart

Figure 31 Activity diagram of Sub-use case 4.3 Unloading from Cart to Conveyor belt

7.5.4 Sub-use case 4.5 - Movement back to resting position/charging point

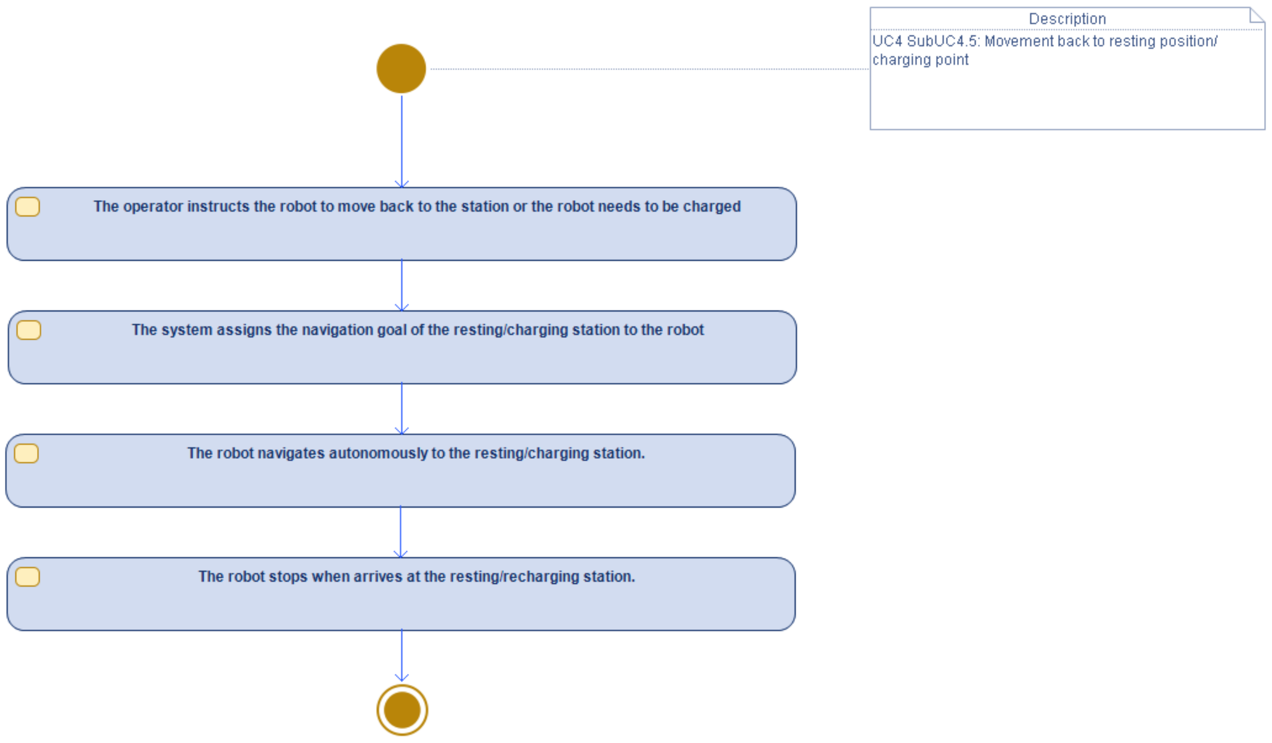

Description
UC4 SubUC4.5: Movement back to resting position/ charging point
The operator instructs the robot to move back to the station or the robot needs to be charged
The system assigns the navigation goal of the resting/charging station to the robot
The robot navigates autonomously to the resting/charging station.
The robot stops when arrives at the resting/recharging station.

Figure 32 Activity diagram of Sub-use case 4.5 Movement back to resting position/charging point

8 Implementation

8.1 Introduction

This chapter comprises an overview of the MANiBOT system and its software modules' requirements. We begin with a brief description of the main framework that most modules employ and introduce the source code management platform. The chapter continues with a description of the hardware requirements for the employed software tools, and continues with a detailed description of the main hardware components of MANiBOT system including a preliminary analysis for the computational resources needed.

8.2 Development View

8.2.1 Robot Operating System (ROS)

The MANiBOT system comprises of one mobile platform with two robotic manipulators, various sensors, and a data management/computation computer, that need to be able to communicate with each other. Therefore, an operating system is needed for the integration of asynchronous sensors with the computation PC and the mobile and manipulation robotic platforms. The role of which is fulfilled by the widely used open-source software platform ROS (Robot Operating System). Since ROS will be the main integration medium, we will briefly describe its main components such as the nodes, messages, topics, and services.

The main processes are being executed as ROS nodes. A ROS node is a process that performs some computation and makes its result available for other processes through the ROS infrastructure. A ROS node can either be a multithread process or a single thread process and it can perform computations by request or at a given frequency. The use of nodes in ROS helps to create an easily maintainable distributed system. Crashes are isolated to individual nodes, which makes debugging easier and more straightforward. All running nodes have a uniquely defined name that makes them visible and identifiable to the rest of the system.

ROS offers a well-supported communication system; nodes can exchange data through topics by posting and receiving ROS messages, which are predefined data structures supporting all common data types (integer, floating point, boolean, etc.) but also can support custom data types. Nodes' communication via messages is done by using the same topic, one node publishes data and another node subscribes to the same topic to receive them. ROS supports remote procedure calls, which can either be the type of services or actions. A ROS service is a blocking call in contrast to a ROS action which is non-blocking. Both services and actions require one or more clients and exactly one server. A client can make a request to a service server and it will receive the corresponding response after the service process ends. An action client can send a goal to an action server, which can send feedback during the whole time of processing and return the final result at the completion of the task.

8.2.2 Module Development

Name	**Owner**	**Programming Language / Technology**
Visual Sensing Module	CERTH/TUW	C++, Python, ROS, OpenCV, PCL, Tensorflow, PyTorch
Tactile Sensing Module	UoB	C++, Python, ROS
Predictive Proximity Sensing Module	SSSA/CERTH	C++, Python, ROS, OpenCV, PCL, Tensorflow, PyTorch

Multimodal Sensing Orchestration Module	CERTH	C++, Python, ROS, OpenCV, PCL, Tensorflow, PyTorch
Visual Scene Understanding Module	CERTH	C++, Python, ROS, OpenCV, PCL, Tensorflow, PyTorch
Federated XAI Module	THL	C++, Python, ROS, Tensorflow, PyTorch
Mobile bimanual low-level controller	UBU	C++, Python, ROS
Robot Navigation Module	CERTH	C++, Python, ROS
Multi-modal adaptive control Module	AUTH	C++, Python, ROS
Mobile Bimanual Motion Coordination Module	AUTH	C++, Python, ROS
Contact Reaching Module	AUTH	C++, Python, ROS
Hybrid Control Module	AUTH	C++, Python, ROS
Adaptive Modelling Module	CERTH	C++, Python, ROS, OpenCV, PCL, Tensorflow, PyTorch
Scene & Task Graph Generator	TUDa	C++, Python, ROS, OpenCV, PCL, Tensorflow, PyTorch
Learning to Plan from Demonstrations Module	TUDa	C++, Python, ROS, Tensorflow, PyTorch
Adaptive Task Planner	TUDa	C++, Python, ROS, Tensorflow, PyTorch
Bi-manual Manipulations Orchestrator	TUDa	C++, Python, ROS, Tensorflow, PyTorch
MANiBOT HRI Interface	CERTH	Javascript, ROS, Angular, NodeJS, roslibJS
Data Management Module	CERTH/THL	Javascript, NodeJS, MySQL, MongoDB

8.2.3 Source Code and Configuration Management

In software development the core of development management depends to a version control system which is essential in large scale projects, where multiple partners must contribute to the developing system. Such system is able to maintain current and older versions of files and folder structure, allowing multiple programmers to easily cooperate. Configuration management contains version control, and extends it by providing additional methods for project management such as:

- identification of Software Configuration Items (SCIs) and their versions through unique identifiers
- control over performed changes through management or instance control
- accounting of the state of individual components
- auditing of selected versions (which are scheduled for a release) by a quality control team

For the development and management of the source code for all MANiBOT modules, Git will be used, which is a free and open-source distributed version control program.

8.3 Deployment View

8.3.1 System H/W Architecture

The system's hardware architecture is designed based on the analysis of the expected robot functionalities, *i.e.* the system functional requirements described in D2.2, the available technology mobile base, and the consortium members' unique ideas. Before constructing the physical prototype, multiple simulated iterations were done to assess its requirements and to confirm hypotheses. Transitioning to the physical system validated many critical structural assumptions, examined manufacturing feasibility and identified discrepancies between simulated and physical environments. A direct result of this process was the selection of steel as the material that would be used for the frame. Similarly, the dimensions were re-iterated and fine-tuned to match the requirements of the use cases and distribute the load in a way that will ensure the system will be able to handle heavy objects without issues. It should be noted that each iteration of the platform maintained all structural advancements of its previous one, and the twelfth and final design is a robust and modular subsystem that has additional compartments to house additional devices and subsystems. The following sections cover the hardware architecture from both the functional and physical perspectives. The functional architecture is a top-down view of how the system's functionality will be realised through specific H/W components. Similarly, the primary system functions will be covered in the following sections. The recognized hardware components of the system are organised in the following structure, which represents the system's physical components:

- MANiBOT Mobile Robotic Platforms;
- Robot Cognition and Data Management Workstation.

Furthermore, we present the hardware requirements per software module. The analysis of hardware requirements per module allows us to estimate the necessary computational resources for the entire system accurately. In this document, we present the initial estimation for each software module, which is likely to be updated later on and reported in the WP6 deliverables.

8.3.1.1 MANiBOT Robotic Platforms

The MANiBOT Mobile Robotic Platforms integrated in the MANiBOT project are two similar mobile robotic platforms based on two different ABB Flexley AMR models (AMRP204 and AMRP603/604) each carrying two GoFa-12 manipulators by ABB. They are modular general-purpose platforms in which various types of sensors will be integrated. The need for two different mobile platforms arises from the different requirements of the projects use cases and the need for different autonomous navigation approaches in the airport and supermarket use cases. The mechanical specifications of the MANiBOT platforms (Figure 33) are based on the user requirements analysis of D2.2 and the description of the pilot sites' environment analysed in section 6.3. The specifications are the following:

- **Width**: ≤ 67 cm
- **Length**: = 180 cm
- **Height:** 100 cm
- 2 x GoFa 12 arm
- 2 x linear actuators
- Mobile platform conveyor belt

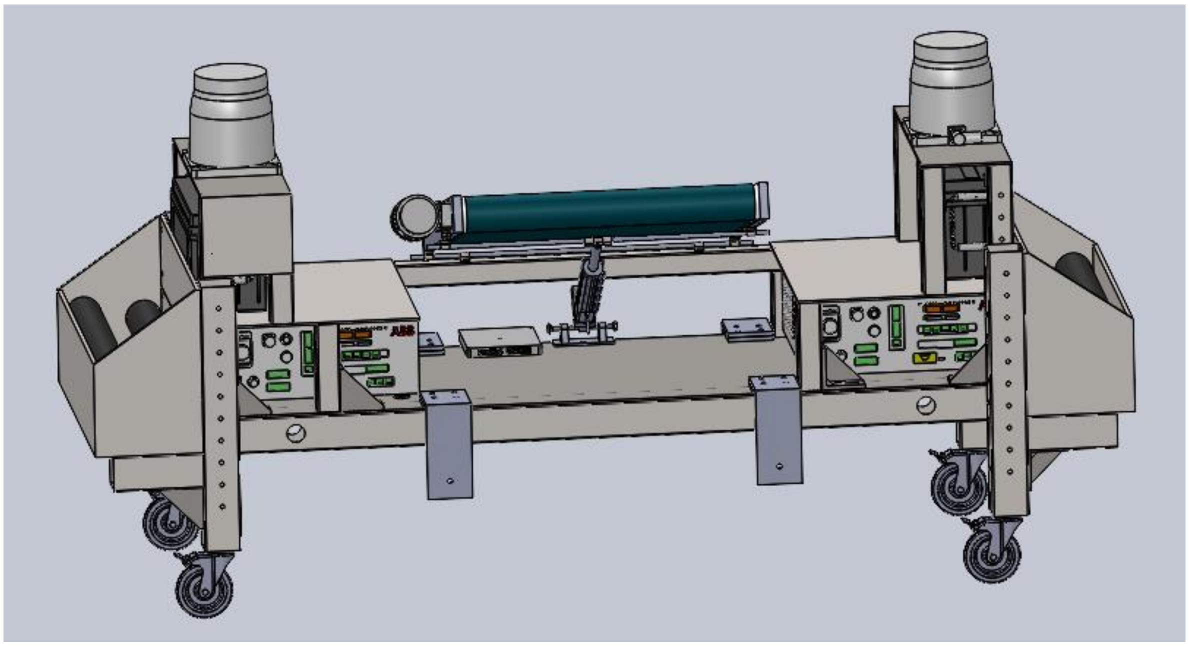


Figure 33 MANiBOT basic platform design. The AMR model at the base will be differentiated between the two platforms.

For UC3 and UC4 the mobile platform's max width will be at 67 cm. The AMR that has been selected for these use cases (ABB AMRP204) has a width of 57 cm, the extra space is occupied by the side structural supports. This width is perfectly adequate for all UCs and affords enough extra space to navigate. With future redesigns, it might be possible to decrease the width even less than 67 cm, but it cannot exceed that value. The length is going to be 1.8 m. This is a compromise to be able to reduce the width. Indeed, the platform has most of its components arranged along its length. This shouldn't pose a problem because all of the UCs are restricted in the lateral dimension. Simulation has validated these dimensions as appropriate for all environments.

Both MANiBOT Mobile Robotic Platforms will integrate two GoFa 12 arms that will perform the manipulation tasks with the help of the different sensors (*e.g.*, vision, tactile and proximity sensors) that will provide enough information of the surroundings, in addition to the required grippers which will be connected to each arm in order to complete the required tasks. It should be noted that the max payload a GoFa 12 can handle is 12 kg, which means that when the system encounters heavier objects, both arms will be used and/or the object will be dragged on the conveyor belt. Since the max weight from the specifications will be 23 kg, the control algorithm will face the challenge to distribute the load between the two arms and complete the tasks efficiently. The next figure provides a scheme of the components integrated in the robot and their interaction.

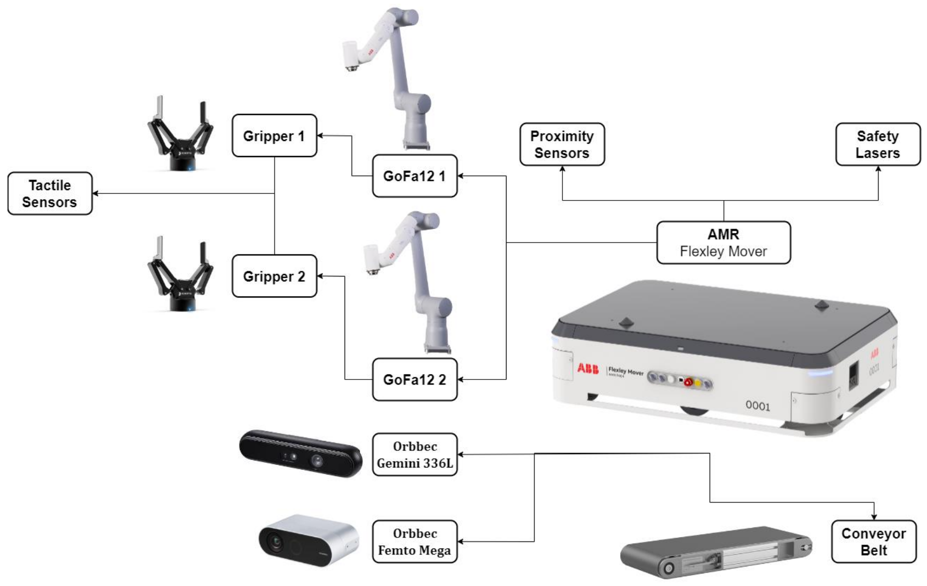


Figure 34 Overview of the MANiBOT hardware architecture

The components shown in Figure 34 will be integrated to the chassis shown in Figure 33, creating the MANiBOT mobile platform. Additional systems and components will be housed on board to collect the signals and coordinate all MANiBOT's functions. These systems are shown in Table 6.

Table 6: *Control and Auxiliary devices*

Device	Quantity	Dimensions (mm) (L × W × H)	Weight (kg)
Robot Controller	2	450 × 600 × 200	25
Inverter	1	450 × 250 × 150	7
Battery	2	400 × 200 × 250	15
Conveyor Belt	1	600 × 700 × 200	50
NUC (Embedded PC)	1	250 × 150 × 40	0.5
PC	2	200 × 200 × 50	—
Jetson (NVIDIA Module)	2	100 × 100 × 50	—
NUC (Embedded PC)	1	200 × 200 × 50	0.5
Computing Device	1	200 × 200 × 150	—

To test the range and the theoretical reachable envelope and ensure that the designed platform addresses the reachability requirements of the use cases, an envelope study was conducted using the ABB RobotStudio software suite. The GoFa 12 arms were placed on the mobile platform in their respective place and their reach was simulated. Figure 35 and Figure 36 show that the reachable envelope in both the supermarket and the airport use cases is large enough to cover all relevant points within a safe operational margin.

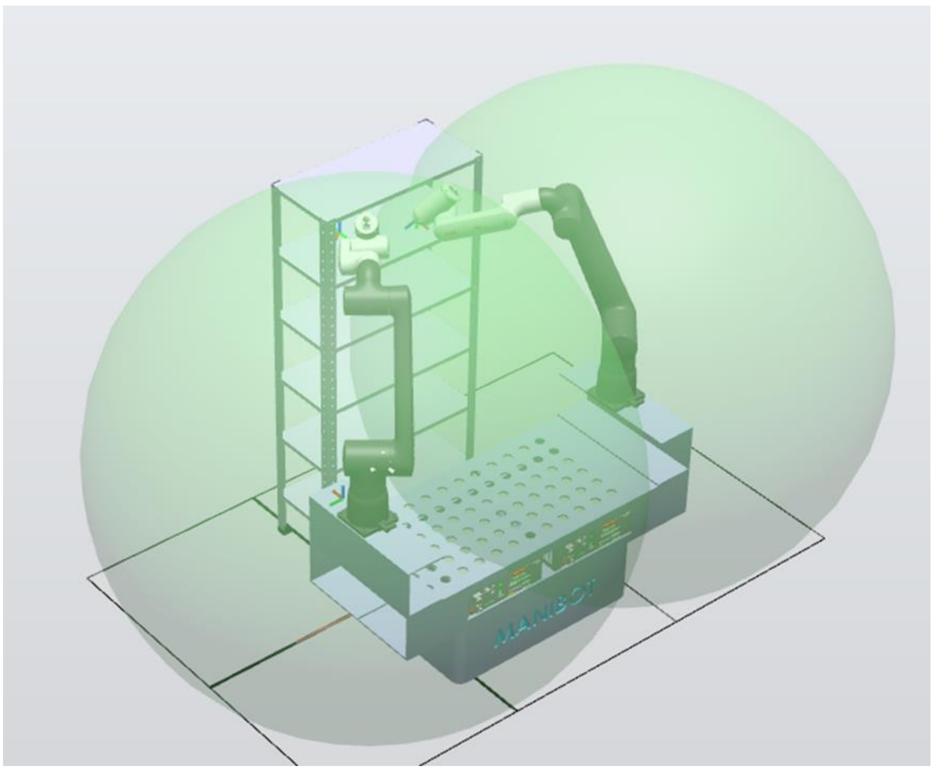


Figure 35 Reach envelop in the supermarket environment.

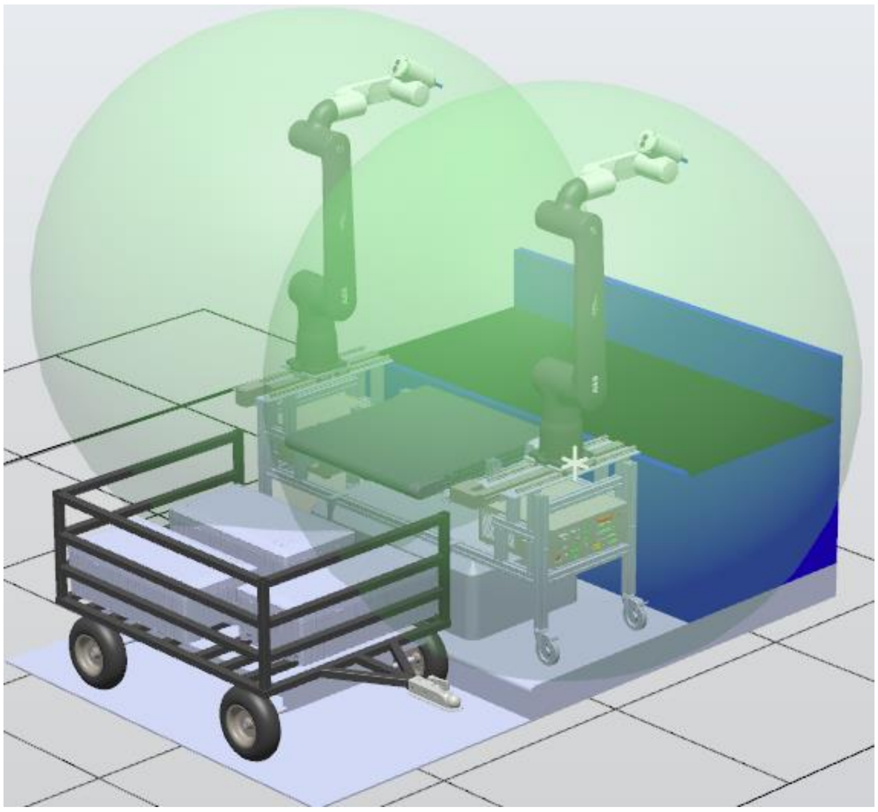


Figure 36 Reach envelop in the airport environment.

The space each arm can reach not only spans all relevant areas, but it has enough overlap with each other to allow for dual manipulation of larger and heavier objects.

Complementary to the reachability study, a trajectory study was also conducted. The purpose of this study was to examine the manipulability and dexterity of the expected motions. This analysis was conducted exclusively for the airport use cases, as it represents the most demanding scenarios within the MANiBOT project, involving the manipulation of large and heavy objects directly on top of the robotic platform.

- Multiple trajectories were tested involving object pickup and placement actions.
- Complex motion sequences were simulated, including reaching across the platform, simultaneous dual-arm coordination, and maneuvers around structural elements.

The following representative tasks were analyzed:

- Picking up luggage from a conveyor and placing it inside a cart.
- Handing over objects between the two arms.

Figures 37-39 show the simulated trajectories when transferring a suitcase from the conveyor belt and placing it inside the transport cart.

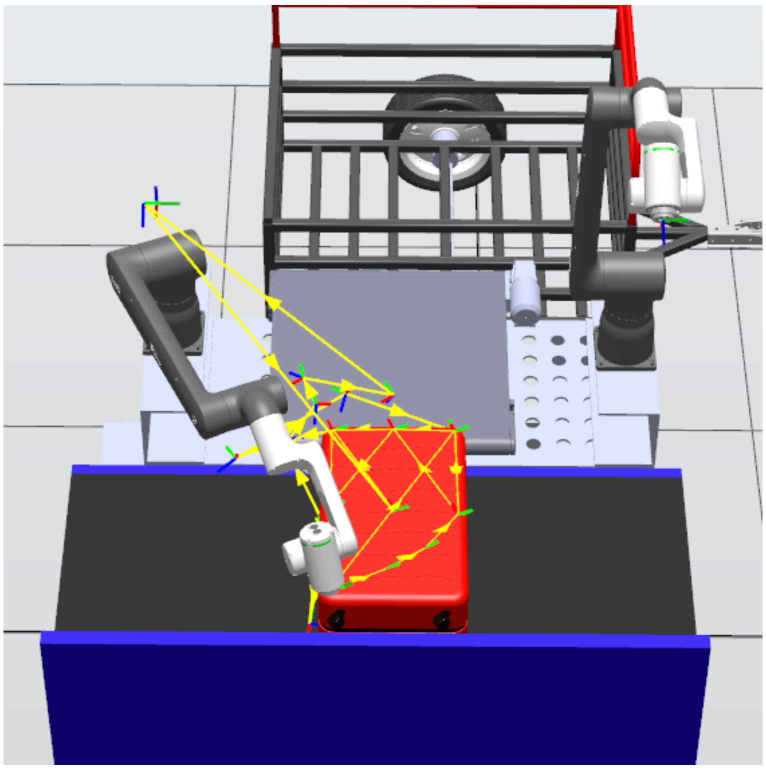


Figure 37 Transferring a suitcase from the conveyor belt

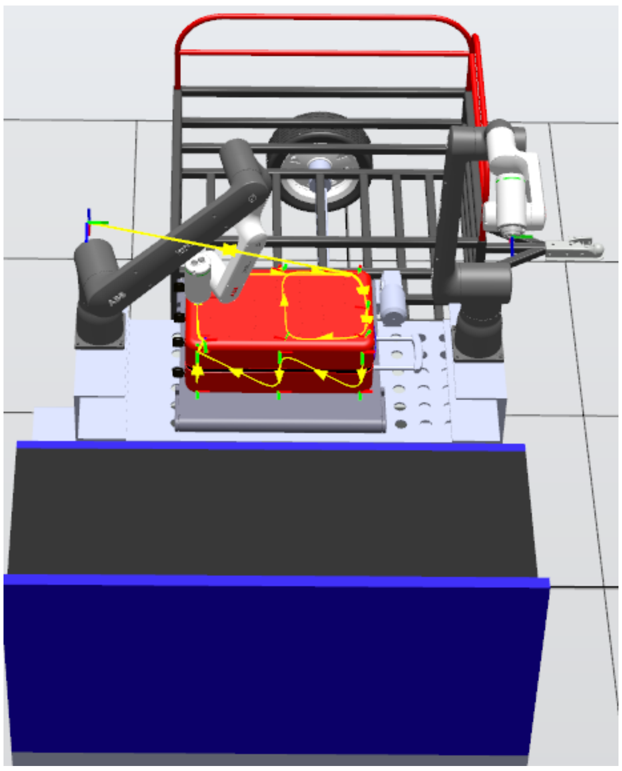


Figure 38 Placing a suitcase on the MANiBOT platform

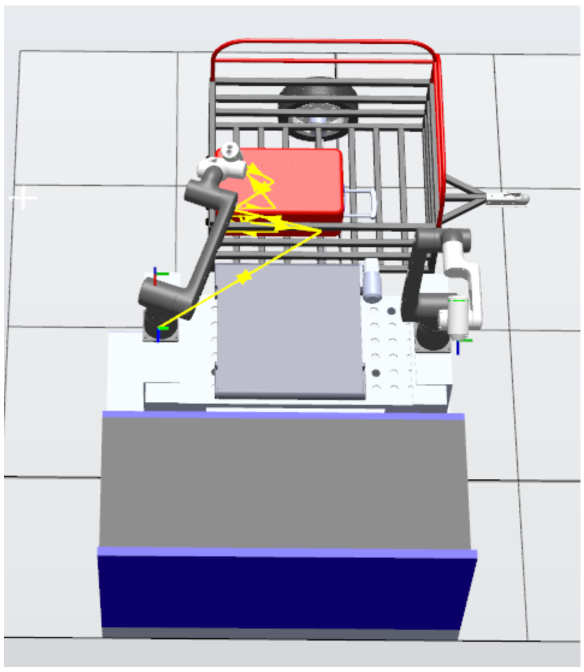


Figure 39 The suitcase is placed inside the cart

During the simulation, no singularities or critical limitations were observed while each task was successfully completed. Additionally, there was no excessive joint effort, confirming the suitability of the robotic arms for the selected use cases.

In summary, the GoFa 12 arms combined with the linear actuators that allow them to move closer to their targets (initially the conveyor belt side and then the cart side) satisfy the requirements for the MANiBOT platform. Their compact design, high dexterity, and mechanical range allow for the execution of the tasks that each use case incorporates, making them a sound choice for this project.

8.3.1.2 Robot Cognition and Data Management Workstation

The robot cognition and data management workstation will provide the necessary resources for the cognition tasks of the robot in real time. Furthermore, it will facilitate the human-robot interaction interfaces with map overlays and task planning by the supervisor. The data management module can be also implemented in the same machine due to the close association it has with the cognition and interaction modules. The minimum hardware requirements for this workstation are the following:

CPU: Multicore processor with 8 cores at 3.5GHz or higher

GPU: Nvidia with min memory 12GB at 15GHz, minimum of 3000 CUDA cores

MEMORY: 32GB DDR5

STORAGE: 2TB RAID1 SSD

NETWORK: 2 x 1Gbps Ethernet ports minimum

8.3.1.3 Hardware requirements per software module

Software module	On-line/Off-line	Hardware requirements
Visual Sensing Module	On-line	CPU: Multicore with 6 cores at 4GHz RAM: 32GB HD: Min 500GB GPU: Nvidia with min memory 12GB at 15GHz, minimum of 3000 CUDA cores Extra Hardware: Orbbec Gemini 336L, Orbbec Femto Mega (RGBD Cameras)
Tactile Sensing Module	On-line	CPU: Multicore with 6 cores at 4GHz RAM: 32GB HD: Min 200GB GPU: Nvidia with min memory 12GB at 15GHz, minimum of 3000 CUDA cores Extra Hardware: Additional USB2 bus in PC
Predictive Proximity Sensing Module	On-line	CPU: Multicore with 6 cores at 4GHz RAM: 32GB HD: Min 200GB GPU: Nvidia with min memory 12GB at 15GHz, minimum of 3000 CUDA cores Extra Hardware: Data Acquisition System (DAQ), Waveform generator
Multimodal Sensing Orchestration Module	On-line	Min 32GB RAM Multithreaded CPU (i7 10th gen or similar) Robot & platform with sensors
Visual Scene Understanding Module	On-line	CPU: Multicore with 6 cores at 4GHz RAM: 32GB HD: Min 500GB GPU: Nvidia with min memory 12GB at 15GHz, minimum of 3000 CUDA cores
Federated XAI Module	On-line / Off-line	CPU: Multicore with 8 cores at 4GHz RAM: 64GB HD: Min 500GB

		GPU: Nvidia with min memory 24GB at 15GHz, minimum of 3000 CUDA cores
Mobile bimanual low-level controller	On-line	CPU: Intel Core i7 RAM: 32GB HD: Min 500GB GPU: Intel Iris X graphics
Robot Navigation Module	On-line	TBD
Multi-modal adaptive control Module	On-line	CPU: Multicore processor with 8 cores at 3.5GHz or higher RAM: 32GB DDR5 HD: 500GB SSD M.2 Network: 2x 1Gbps Ethernet ports
Mobile Bimanual Motion Coordination Module	On-line	Multicore processor with 8 cores at 3.5GHz or higher 32GB DDR5 500GB SSD M.2 Network: 2x 1Gbps Ethernet ports
Contact Reaching Module	On-line	Multicore processor with 8 cores at 3.5GHz or higher 32GB DDR5 500GB SSD M.2 Network: 2x 1Gbps Ethernet ports
Hybrid Control Module	On-line	Multicore processor with 8 cores at 3.5GHz or higher 32GB DDR5 500GB SSD M.2 Network: 2x 1Gbps Ethernet ports
Adaptive Modelling Module	On-line / Off-line	CPU: Multicore with 6 cores at 4GHz RAM: 32GB HD: Min 500GB GPU: Nvidia with min memory 12GB at 15GHz, minimum of 3000 CUDA cores
Scene & Task Graph Generator	On-line / Off-line	CPU: Multicore with 6 cores at 4GHz RAM: 64GB HD: Min 500GB GPU: Nvidia with min memory 24GB at 15GHz, minimum of 3000 CUDA cores
Learning to Plan from Demonstrations Module	On-line / Off-line	CPU: Multicore with 6 cores at 4GHz RAM: 64GB HD: Min 500GB GPU: Nvidia with min memory 24GB at 15000MHz, minimum of 3000 CUDA cores
Adaptive Task Planner	On-line	CPU: Multicore with 6 cores at 4GHz RAM: 64GB HD: Min 500GB GPU: Nvidia with min memory 24GB at 15GHz, minimum of 3000 CUDA cores
Bi-manual Manipulations Orchestrator	On-line	CPU: Multicore with 6 cores at 4GHz RAM: 64GB

		HD: Min 500GB GPU: Nvidia with min memory 24GB at 15GHz, minimum of 3000 CUDA cores
MANiBOT HRI Interface	On-line	CPU: Multicore with 6 cores at 4GHz RAM: 32GB HD: Min 500GB GPU: Nvidia with min memory 12GB at 15GHz, minimum of 3000 CUDA cores
Data Management Module	On-line / Off-line	HD: 2TB SSD CPU: Multicore with 6 cores at 4GHz RAM: 32GB

8.3.2 Preliminary Analysis of Computational Resources

As briefly described in section 8.3.1, the mobile robotic platforms of MANiBOT will integrate multiple sensors and a large amount of data will be collected during their operation. Most of the collected data should be processed on-line for the navigation and localization system and for the perception modules. Thus, significant computational resources will be required (see Table 6 for a preliminary estimation of these resources). An average commercial computer cannot handle the platform's multiple processes, which will require data to be collected, processed and interpreted on-line. Our preliminary analysis supports that the NVIDIA Jetson boards will suffice for processing data collected by RGB-D sensors, and CUDA enabled GPUs will suffice for processing the navigation/localization sensors and additional sensor data.

Furthermore, the system requires significant computational resources for training models for all software modules that use AI and the off-line procedures that are time-consuming, such as adaptive modelling. Thus, extra workstation computers with CUDA enabled GPUs, multi-threaded CPU and minimum 32GB RAM are required. Given that this is the preliminary analysis, we plan to present the computational analysis in detail in WP6 deliverables.

9 Conclusion

This report presented the MANiBOT conceptual architecture, describing the system's main building blocks, providing a comprehensive overview of all components and illustrating their high-level functionality and interdependencies. The analysis conducted here follows a hierarchical pattern in detail. Inspired from the Rozanski & Woods paradigm [1], we established a respective methodology to define the MANiBOT architecture on a hierarchical basis.

This starts from the MANiBOT concept and proceeds to the high-level description of the abstract modules, describing the system's functional components, their responsibilities and primary interactions with each other. Moving deeper in the hierarchy, the dynamic analysis of the system provides an overview of each sub-use case by using an activity diagram, depicting the flow of the robot's activities.

A functional analysis of the system was performed by defining the system functional requirements depending on the identified sub-use cases of the system. Also, technical specifications, comprising performance requirements, software and hardware requirements, were defined and mapped to user requirements and sub-use cases.

A Development view presents how the architecture supports the development process, while the Deployment View describes the hardware requirements of the software components for the MANiBOT use cases.

To sum up, by applying the presented methodology to our architectural design, we have identified the main building blocks of the system and broke them down into manageable components, with specific responsibilities and functionalities. Since we are applying an iterative approach to system design and development the presented architecture and specifications will further be revised and updated utilising the development and deployment feedback. The updated analysis will be tracked via internal reports and presented in future MANiBOT deliverables, such as D6.4 "Integrated bimanual mobile manipulator robot".

References

[1] Rozanski, Nick, and Eóin Woods. Software systems architecture: working with stakeholders using viewpoints and perspectives. Addison-Wesley, 2011.

[2] D2.2 User requirements and use cases

Annex I: Priority Ranking of the System Functional Requirements

No. / ID	Ranked Robot Functional Requirements	Priority
SFR01	Recognize the product to be replenished	**High**
SFR02	Recognize expired and damaged products	**Medium**
SFR03	Recognize the replenishment cart	**Medium**
SFR04	Recognize dimensions and orientation of each product per box	**High**
SFR05	Recognize dimensions and orientation of boxes per pallet	**High**
SFR06	Recognize the type of box	**Medium**
SFR07	Recognize electronic shelf labels and extract content	**Low**
SFR08	Recognize the remaining space on shelves	**High**
SFR09	Detect the appropriate baggage cart	**Medium**
SFR10	Detects moving baggage on the conveyor belt	**High**
SFR11	Recognize the front side (i.e., customer facing side) for each box	**Medium**
SFR12	Detects each baggage separately identifying its shape and dimensions	**High**
SFR13	Recognize how many boxes of a specific product item can fit on the shelf	**Medium**
SFR14	Recognize when a baggage cart is full or empty	**High**
SFR15	Recognize damaged luggage for the arrivals	**Medium**
SFR16	Recognize the luggage material (soft or hard)	**Medium**
SFR17	Ability to determine the optimal quantity of products for placement at the forefront of the shelf	**Medium**
SFR18	Recognize misplaced products on the shelves	**Medium**
SFR19	Recognize the free space at the target destination (baggage cart or belt) to place the luggage	**High**
SFR20	Performs quality check on products (flowers are not withered, fruits and vegetables are not deviated)	**Low**
SFR21	Recognizes the location for the baggage placement	**High**
SFR22	Recognize the proximate area and shelves and compare the predefined assortment planogram with the actual placement of the ESLs	**Low**
SFR23	Recognize boxes' arrangement/placement on pallet (top to bottom)	**Medium**
SFR24	Recognize product assortment on shelves according to ESLs and per product defined area (by assortment plan)	**Medium**
SFR25	Identify the correct destination on the shelf (through ESL) given the prior recognition of the mixed pallet and the respective product on the mixed pallet	**High**
SFR26	Inspect the state and replenishment feasibility of each package (per product)	**Medium**
SFR27	Recognize the mixed pallet	**High**

SFR28	Recognize and track humans' trajectory	**Medium**
SFR29	Detects proximity of moving baggage tractor with loaded carts	**Medium**
SFR30	The robot decides which baggage to remove with least disturbance to surrounding baggage	**High**
SFR31	Platform 2D/3D localization	**High**
SFR32	Platform Path planning ability	**High**
SFR33	Platform human-aware navigation	**High**
SFR34	Ability to move on the pavement area alongside the conveyor belt	**High**
SFR35	Ability to step up to the pavement area (height 15cm) around the conveyor belt (with or without the use of a ramp)	**Low**
SFR36	Navigate avoiding restricted areas	**Medium**
SFR37	The robot selects a desired position regarding the baggage cart	**Medium**
SFR38	Grasp products from boxes	**High**
SFR39	Ability to pick and place single items	**High**
SFR40	Ability to pick up boxes with max weight up to 10 kg	**High**
SFR41	Ability to pick up boxes without damaging them	**High**
SFR42	Place the products in the correct positions	**High**
SFR43	Place the products that expire earlier in the front of the shelf ahead of newly received items	**Medium**
SFR44	Place the products in a specified orientation, ensuring the front side of the product faces the customer	**Medium**
SFR45	Place momentarily the boxes on its equipped table or another interim storage position	**High**
SFR46	Manipulate and replenish product boxes	**Medium**
SFR47	Ability to close and open nested shelves, fridges, or freezing units	**Low**
SFR48	Ability to grab newspapers and magazines enclosed in brown boxes and place them in storage positions	**Low**
SFR49	Ability to open closed boxes and stock their content in the corresponding shelves	**Low**
SFR50	Grasp the luggage that made from soft material gently	**Medium**
SFR51	Ability to prepare the products/SKUs (e.g., removing card box or plastic foil toppers)	**Low**
SFR52	Place misplaced products in an interim storage position or to the cart or to the right shelf position	**High**
SFR53	Place the surplus products that cannot fit on the shelf, in a predefined interim storage position	**Medium**
SFR54	Place the products in the right position without using a planogram or consult it when it is available	**Medium**
SFR55	Place the luggage made of hard material below the luggage made of soft material	**Medium**
SFR56	Place the luggage with a distance between them on the reclaim belt for the arrivals	**Medium**
SFR57	Place the luggage within the boundaries of the belt/cart	**High**

SFR58	Handle luggage only for the flight that has been allocated to	**Medium**
SFR59	Ability to restock shelves with product/SKUs from the pallet (top to bottom)	**High**
SFR60	Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism	**Medium**
SFR61	Place empty (cardboard) boxes from replenished items into a bin cart located next to the pallet.	**Low**
SFR62	Manipulate the baggage to detect the bagtag	**High**
SFR63	Ability to avoid collision with moving or static objects	**High**
SFR64	Ability to avoid collisions with itself in bimanual tasks	**High**
SFR65	Detects the optimal contact points in boxes, bags and products to grasp the item	**High**
SFR66	Robot pauses its current task if employee or customer is too close to it	**High**
SFR67	Robots pauses and resumes tasks according to user's command	**Medium**
SFR68	Ability to complete restocking tasks for shelves within a mobility radius (parameter with default value e.g., 5 meters) from the mixed pallet	**Medium**
SFR69	Decides from where it can leave the pavement track (either because it has been instructed to by the operator or if it needs to be charged)	**Medium**
SFR70	User defining replenishment task and destination	**Medium**
SFR71	User defining task and destination in the airport belts	**Medium**
SFR72	User commanding through interaction interface the robot to move to specified location (shelf or belt)	**Medium**
SFR73	Interaction interface notifies the user about the restocking task progress	**Medium**
SFR74	Interaction interface notifies the user if the robot needs help with the identification of a product, bag etc.	**Medium**
SFR75	Interaction interface notifies the user for help when a product falls on the floor	**Medium**
SFR76	Ability to give a signal (either audio or visual) when robot's workspace is invaded by a human	**High**
SFR77	Interaction interface notifies the user when the replenishment process is complete	**Medium**
SFR78	Interaction interface notifies the user if any of the processes failed	**Medium**
SFR79	Interaction interface notifies the user about the absence of planogram and difficulty in defining the right product position	**Medium**
SFR80	Interaction interface alerts when the bag it is manipulating is out of its capabilities (too heavy, too long, no handles, out of reach etc.).	**Medium**
SFR81	Interaction interface alerts when no bagtag found	**Medium**
SFR82	The robot alerts the operator if no other cart is found, goes in standby mode and awaits input from operator.	**Medium**
SFR83	Notifies the user through the UI for the table/interim storage state	**Medium**
SFR84	Robot stores/retrieves the bagtag information of the placed baggage in/from the database	**High**
SFR85	Scans barcode and retrieves information of the paper label and checks if it corresponds to the product to be replenished	**Medium**
SFR86	Retrieves information about the ESL tags	**Low**

SFR87	If a mismatch identified, log the distance between the position of a product in the assortment plan and the position of the ESL	**Low**
SFR88	Keep count of the number of baggage loaded	**High**
SFR89	Scans the bag tag (when placing the baggage) for BTRS (Baggage Tracking and Reconciliation System) purposes.	**High**
SFR90	Records and logs in the database the products that are not suitable for restocking (surplus, expired, damaged)	**Medium**
SFR91	Specific AI modules of the robot are updated while maintaining organisation and persons privacy	**High**
SFR92	The outputs of specific AI modules of the robot are explained through the UI (User Interface)	**High**

Annex II Mapping of the User Requirements to Technical Specifications

		High Priority **Medium Priority** **Low Priority**	Technical Specifications		
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC1.1 Movement from resting position/charging station to the target replenishment cart	*UR01,UR03,UR05,UR06,UR07, UR20,UR24,UR27,UR28,UR29, UR30,UR32,UR33,UR34,UR35,UR36*	SFR031 - Platform 2D/3D localization	real-time, < 10 cm (RPE)	2D/3D Lidars, RTK-GPS	
		SFR032 - Platform Path planning ability	real-time		C++, Python
		SFR033 - Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR066 – Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR003 - Recognize the replenishment cart	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR028 - Recognize and track humans' trajectory	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR036 - Navigate avoiding restricted areas	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR067 – Robot pauses and resumes tasks according to user's commands	real-time		Angular, NodeJS, Javascript, Web browser
		SFR070 - User defining replenishment task and destination	real-time		Angular, NodeJS, Javascript, Web browser
		SFR072 - User commanding through interaction interface the robot to move to specified location (shelf or belt)	real-time, <10cm accuracy		Angular, NodeJS, Javascript, Web browser
		SFR078 - Interaction interface notifies the user if any of the processes failed	real-time		Angular, NodeJS, Javascript, Web browser

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC1.2. Identification of the products in the replenishment cart	*UR2,UR8,UR12,UR13,UR20,UR21,UR23,UR24,UR25,UR27,UR28,UR29,UR30,UR32,UR33,UR34,UR35,UR36*	SFR001 - Recognize the product to be replenished	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR004 - Recognize dimensions and orientation of each product per box	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR002 – Recognize expired and damaged products	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR074 - Interaction interface notifies the user if the robot needs help with the identification of a product, bag etc.	real-time		Angular, NodeJS, Javascript, Web browser
		SFR075 - Interaction interface notifies the user for help when a product falls on the floor	real-time		Angular, NodeJS, Javascript, Web browser
		SFR078 - Interaction interface notifies the user if any of the processes failed	real-time		Angular, NodeJS, Javascript, Web browser

		🟥 **High Priority**	Technical Specifications		
		🟦 **Medium Priority**			
		🟩 **Low Priority**			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC1.3. Identification of the right position for the product on the shelf	*UR3,UR4,UR5,UR6, UR7,UR8,UR12,UR13,UR20,UR22,UR23, UR24,UR26,UR27,UR28,UR29,UR30,UR2,UR32,UR33,UR34, UR35,UR36*	🟥 SFR001 - Recognize the product to be replenished	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟥 SFR076 - Ability to give a signal (either audio or visual) when robot's workspace is invaded by a human	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores, Proximity Sensors	OpenCV, PCL, Camera API, Web browser, Javascript, Angular, NodeJS
		🟦 SFR002 – Recognize expired and damaged products	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟦 SFR026 - Inspect the state and replenishment feasibility of each package (per product).	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, Camera API
		🟦 SFR074 - Interaction interface notifies the user if the robot needs help with the identification of a product, bag etc.	real-time		Angular, NodeJS, Javascript, Web browser
		🟦 SFR075 - Interaction interface notifies the user for help when a product falls on the floor	real-time		Angular, NodeJS, Javascript, Web browser
		🟦 SFR078- Interaction interface notifies the user if any of the processes failed	real-time		Angular, NodeJS, Javascript, Web browser
		🟦 SFR085 - Scans barcode and retrieves information of the paper label and checks if it corresponds to the product to be replenished	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras, Scanner	Angular, NodeJS, Javascript, Web browser, MySQL, Camera API

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC1.4 Replenishment of the shelves	*UR3,UR5,UR6,UR7, UR8,UR9,UR10,UR11,UR12,UR13,UR14, UR15,UR16,UR17,UR20,UR21,UR24,UR25,UR26,UR27,UR28,UR29,UR30,UR31,UR32,UR33,UR34,UR35,UR36*	SFR038 - Grasp products from boxes	real-time	RGBD cameras	OpenCV, PCL, Camera API, PyTorch, Python
		SFR042 - Place the products in the correct positions	real-time	RGBD cameras	OpenCV, PCL, Camera API, PyTorch, Python
		SFR052 - Place misplaced products in an interim storage position or to the cart or to the right shelf position	real-time	RGBD cameras	OpenCV, PCL, Camera API, PyTorch, Python
		SFR059 - Ability to restock shelves with product/SKUs from the pallet (top to bottom)	real-time	2D/3D Lidars, RGBD cameras	OpenCV, PCL, Camera API, PyTorch, Python
		SFR063 - Ability to avoid collision with moving or static objects	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR064 - Ability to avoid collisions with itself in bimanual tasks	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR065 - Detects the optimal contact points in boxes, bags and products to grasp the item	real-time	RGBD cameras	OpenCV, PCL, Camera API, PyTorch, Python
		SFR076 - Ability to give a signal (either audio or visual) when robot's workspace is invaded by a human	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores, Proximity Sensors	OpenCV, PCL, Camera API, Web browser, Javascript, Angular, NodeJS
		SFR045 - Place momentarily the boxes on its equipped table or another interim storage position	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR002 – Recognize expired and damaged products	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR011 - Recognize the front side (i.e., customer facing side) for each box	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, PyTorch, Python

		SFR013 - Recognize how many boxes of a specific product item can fit on the shelf	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, PyTorch, Python
		SFR017 - Ability to determine the optimal quantity of products for placement at the forefront of the shelf	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, PyTorch, Python
		SFR018- Recognize misplaced products on the shelves	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, PyTorch, Python
		SFR043 - Place the products that expire earlier in the front of the shelf ahead of newly received items	real-time	RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR044 - Place the products in a specified orientation, ensuring the front side of the product faces the customer	real-time	RGBD cameras, 2D/3D LiDAR	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR053 - Place the surplus products that cannot fit on the shelf, in a predefined interim storage position	real-time	RGBD cameras, 2D/3D LiDAR	OpenCV, PCL, Camera API
		SFR060 - Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR078 - Interaction interface notifies the user if any of the processes failed	real-time		Angular, NodeJS, Javascript, Web browser
		SFR090 - Records and logs in the database the products that are not suitable for restocking (surplus, expired, damaged)	online	HDD >= 500GB, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	MySQL or MongoDB

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC1.5 Movement back to the resting position / charging station	*UR3,UR5,UR6,UR7, UR18,UR19,UR20,UR27,UR28,UR29,UR30,UR32,UR33,UR34,UR35,UR36*	SFR031 - Platform 2D/3D localization	real-time, < 10 cm (RPE)	2D/3D Lidars, RTK-GPS	
		SFR032 - Platform Path planning ability	real-time		C++, Python
		SFR033 - Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras, GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores,	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR066 - Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR028 – Recognize and track humans' trajectory	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	OpenCV, PCL, Camera API, Pytorch, Python
		SFR036 - Navigate avoiding restricted areas	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR073 - Interaction interface notifies the user about the restocking task progress	real-time		Angular, NodeJS, Javascript, Web browser
		SFR077 - Interaction interface notifies the user when the replenishment process is complete	real-time		Angular, NodeJS, Javascript, Web browser
		SFR078- Interaction interface notifies the user if any of the processes failed	real-time		Angular, NodeJS, Javascript, Web browser

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC2.1 Movement from resting position/charging station to target destination for replenishment	UR03,UR06,UR07, UR24,UR29,UR30, UR32,UR35,UR36, UR54,UR55,UR56, UR57,UR59,UR61, UR62,UR64,UR65, UR68,UR83,UR84, UR85	SFR028 - User defining replenishment task and destination	online	RGBD cameras	Web browser
		SFR031 - Platform 2D/3D localization	real-time	2D/3D Lidars, RTK-GPS	OpenCV, PCL, Camera API, Pytorch, Python, C++
		SFR032 - Platform Path planning ability	real-time		C++, Python
		SFR033 - Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras, GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, Proximity sensors	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR036 - Navigate avoiding restricted areas	real-time	2D/3D Lidars, RGBD cameras	OpenCV, PCL, Camera API
		SFR066 – Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR067 - Robots pauses and resumes tasks according to user's command	real-time		Web browser, Javascript, Angular, NodeJS
		SFR070 - User defining replenishment task and destination	real-time		Web browser, Javascript, Angular, NodeJS
		SFR072 - User commanding through interaction interface the robot to move to specified location (shelf or belt)	real-time		Web browser, Javascript, Angular, NodeJS
		SFR078 - Interaction interface notifies the user if any of the processes failed	real-time		Web browser, Javascript, Angular, NodeJS

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC2.2 Identification of the products on mixed pallet	UR03,UR06,UR07, UR09,UR21,UR24, UR29,UR30,UR32, UR35,UR36,UR41, UR42,UR45,UR46, UR54,UR55,UR56, UR57,UR 59,UR60,UR61,UR 62,UR63,UR64,UR 65,UR68,UR83,UR 84,UR85	SFR001 - Recognize the product to be replenished	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, Camera API
		SFR005 - Recognize dimensions and orientation of boxes per pallet	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, Camera API
		SFR006 - Recognize the type of box	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, Camera API
		SFR023- Recognize boxes' arrangement/placement on pallet (top to bottom)	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, Camera API
		SFR027 - Recognize the mixed pallet	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR074 – Interaction interface notifies the user if the robot needs help with the identification of a product, bag etc.	real-time		Web browser, Javascript, Angular, NodeJS
		SFR075 - Interaction interface notifies the user for help when a product falls on the floor	real-time		Web browser, Javascript, Angular, NodeJS
		SFR078 - Interaction interface notifies the user if any of the processes failed	real-time		Web browser, Javascript, Angular, NodeJS

		🟥 High Priority	**Technical Specifications**		
		🟦 Medium Priority			
		🟩 Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC2.3 Identification of the correct destination for the product on the shelf (via ESL recognition and assortment plan)	UR06,UR07,UR21, UR24,UR29,UR30, UR32,UR35,UR36, UR37,UR38,UR39, UR43,UR54,UR55, UR56,UR57,UR 59,UR60,UR61,UR 62,UR63,UR64,UR 65,UR68,UR69,UR 70,UR71,UR75,UR 83,UR84,UR85	🟥 SFR001 - Recognize the product to be replenished	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟩 SFR007 - Recognize electronic shelf labels and extract content	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟦 SFR024 - Recognize product assortment on shelves according to ESLs and per product defined area (by assortment plan)	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟥 SFR025 – Identify the correct destination on the shelf (through ESL) given the prior recognition of the mixed pallet and the respective product on the mixed pallet	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟦 SFR26- Inspect the state and replenishment feasibility of each package (per product)	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟥 SFR027 - Recognize the mixed pallet	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API

		SFR066 - Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR068 – Ability to complete restocking tasks for shelves within a mobility radius (parameter with default value e.g. 5 meters) from the mixed pallet	real-time	2D/3D Lidars, RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores Robotic Hands and Grippers	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API, C++
		SFR074 - Interaction interface notifies the user if the robot needs help with the identification of a product, bag etc.	real-time		Web browser, Javascript, Angular, NodeJS
		SFR075 - Interaction interface notifies the user for help when a product falls on the floor	real-time		Web browser, Javascript, Angular, NodeJS
		SFR076 - Ability to give a signal (either audio or visual) when robot's workspace is invaded by a human	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores, Proximity Sensors	OpenCV, PCL, Camera API, Web browser, Javascript, Angular, NodeJS
		SFR086 - Retrieves information about the ESL tags	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API
		SFR087 - If a mismatch identified, log the distance between the position of a product in the assortment plan and the position of the ESL	real-time	RGBD cameras, Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API

		🟥 **High Priority**	**Technical Specifications**		
		🟦 **Medium Priority**			
		🟩 **Low Priority**			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC2.4 Preparation of the products before replenishment	UR03,UR06,UR07, UR13,UR21,UR23, UR24,UR26,UR29, UR30,UR32,UR35, UR36,UR54,UR55, UR56,UR57,UR59, UR60,UR61,UR62, UR63,UR64,UR65, UR66,UR67,UR68, UR74,UR77,UR78, UR79,UR89,UR81, UR82,UR83,UR84, UR85	🟦 SFR002 - Recognize expired and damaged products	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟥 SFR042 - Place the products in the correct positions	real-time	RGBD cameras, Proximity Sensors	Robotic Hands and Grippers
		🟩 SFR051 - Ability to prepare the products/SKUs (e.g. removing cardbox or plastic foil toppers)	real-time	RGBD cameras, Proximity Sensors	Robotic Hands and Grippers
		🟥 SFR066 – Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC3.1 Preparing for loading	*UR1, UR2, UR3, UR4, UR5, UR15, UR19, UR21, UR22*	SFR028 - Recognize and track humans' trajectory	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR037 - The robot selects a desired position regarding the baggage cart	real-time	RGBD cameras, 2D/3D LiDAR	OpenCV, PCL, Camera API
		SFR067 – Robot pauses and resumes tasks according to user's commands	real-time		Web browser, Javascript, Angular, NodeJS
		SFR071 - User defining task and destination in the airport belts	real-time		Web browser, Javascript, Angular, NodeJS
		SFR072 - User commanding through interaction interface the robot to move to specified location (shelf or belt)	real-time, <10cm accuracy		Web browser, Javascript, Angular, NodeJS,

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC3.2 Recognizing Baggage for assigned flight	*UR6, UR7, UR8, UR23, UR24, UR25*	🟥 **SFR010 - Detects moving baggage on the conveyor belt**	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		🟥 **SFR062 - Manipulate the baggage to detect the bagtag**	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		🟥 **SFR084 – Robot stores/retrieves the bagtag information of the placed baggage in/from the database**	On-line	HDD >= 500GB, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores,	MySQL or MongoDB
		🟦 **SFR015 - Recognize damaged luggage for the arrivals**	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		🟦 **SFR016 - Recognize the luggage material (soft or hard)**	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API
		🟦 **SFR081 - Interaction interface alerts when no bagtag found**	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	Web browser, Javascript, Angular, NodeJS, Python, Pytorch, OpenCV, PCL, Camera API

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC3.3 Loading of allocated baggage on the cart	*UR5, UR6, UR9, UR11, UR13, UR14, UR16, UR17, UR18, UR19, UR23, UR27, UR28, UR29, UR32*	SFR012 - Detects each baggage separately identifying its shape and dimensions	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR014 - Recognize when a baggage cart is full or empty	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR019 - Recognize the free space at the target destination (baggage cart or belt) to place the luggage	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR021 - Recognizes the location for the baggage placement	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API
		SFR057 - Place the luggage within the boundaries of the belt/cart	Real-time	2D/3D LiDAR, RGBD cameras	OpenCV, PCL, Camera API
		SFR063 - Ability to avoid collision with moving or static objects	Real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR064 - Ability to avoid collisions with itself in bimanual tasks	Real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR065 - Detects the optimal contact points in boxes, bags and products to grasp the item	Real-time	RGBD cameras	OpenCV, PCL, Camera API, Pytorch, Python
		SFR084 - Robot stores/retrieves the bagtag information of the placed baggage in/from the database	On-line	HDD >= 500GB, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	MySQL or MongoDB
		SFR088 – Keep count of the number of baggage loaded	On-line	HDD >= 500GB, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	MySQL or MongoDB
		SFR089 - Scans the bag tag (when placing the baggage) for BTRS (Baggage	real-time	RGBD cameras	OpenCV, PCL, Camera API

		Tracking and Reconciliation System) purposes			
		SFR009 - Detect the appropriate baggage cart	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR050 - Grasp the luggage that made from soft material gently	Real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR055 - Place the luggage made of hard material below the luggage made of soft material	Real-time	RGBD cameras, 2D/3D LiDAR	OpenCV, PCL, Camera API,
		SFR060 - Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism	Real-time	2D/3D LiDAR, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR080 - Interaction interface alerts when the bag it is manipulating is out of its capabilities (too heavy, too long, no handles, out of reach etc.).	Real-time		Web browser, Javascript, Angular, NodeJS

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC3.4 Movement back to resting position/charging point	*UR1, UR2, UR31*	**SFR031 - Platform 2D/3D localization**	real-time, < 10 cm (RPE)	2D/3D Lidars, RTK-GPS	
		SFR032 - Platform Path planning ability	real-time		Python
		SFR033 – Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras, GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR034 - Ability to move on the pavement area alongside the conveyor belt	Real-time	2D/3D Lidars, RGBD cameras	
		SFR066 - Robot pauses its current task if employee or customer is too close to it	Real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR028 - Recognize and track humans trajectory	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	OpenCV, PCL, Camera API, Pytorch, Python
		SFR036 - Navigate avoiding restricted areas	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR069 - Decides from where it can leave the pavement track (either because it has been instructed to by the operator or if it needs to be charged)	Real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR035 - Ability to step up to the pavement area (height 15cm) around the conveyor belt (with or without the use of a ramp)	<= 15cm height, > 5° angle	2D/3D Lidars, RGBD cameras	OpenCV, PCL, Camera API

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC4.1 Movement from resting position/charging station to the target position	*UR1, UR2, UR5, UR19, UR21, UR22*	SFR031 - Platform 2D/3D localization	real-time, < 10 cm (RPE)	2D/3D Lidars, RTK-GPS	
		SFR032 - Platform Path planning ability	real-time		
		SFR033 - Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras, GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores,	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR034 - Ability to move on the pavement area alongside the conveyor belt	real-time	2D/3D Lidars, RGBD cameras	
		SFR066 – Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR028 - Recognize and track humans' trajectory	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR036 - Navigate avoiding restricted areas	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR067 – Robot pauses and resumes tasks according to user's commands	real-time		Web browser
		SFR071 - User defining task and destination in the airport belts	real-time		Web browser
		SFR072 - User commanding through interaction interface the robot to move to specified location (shelf or belt)	real-time, <10cm accuracy		Web browser
		SFR035 - Ability to step up to the pavement area (height 15cm) around the conveyor belt (with or without the use of a ramp)	<= 15cm height, > 5° angle	2D/3D Lidars, RGBD cameras	OpenCV, PCL, Camera API

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC4.2 Preparing for unloading	*UR2, UR3, UR4, UR15, UR19, UR21, UR22*	SFR09 - Detect the appropriate baggage cart	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR015 - Recognize damaged luggage for the arrivals	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR028 - Recognize and track humans' trajectory	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	OpenCV, PCL, Camera API, Pytorch, Python
		SFR029 - Detects proximity of moving baggage tractor with loaded carts	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR037 - The robot selects a desired position regarding the baggage cart	real-time	2D/3D Lidars, RGBD cameras	OpenCV, PCL, Camera API
		SFR058 - Handle luggage only for the flight that has been allocated to	real-time	RGBD cameras	OpenCV, PCL, Camera API

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC4.3 Unloading from Cart to Conveyor belt	*UR5, UR6, UR7, UR8, UR9, UR10, UR11, UR12, UR13, UR14, UR16, UR18, UR23, UR24, UR25, UR27, UR28, UR29, UR32*	SFR012 - Detects each baggage separately identifying its shape and dimensions	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR014 - Recognize when a baggage cart is full or empty	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR019 - Recognize the free space at the target destination (baggage cart or belt) to place the luggage	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR021 – Recognizes the location for the baggage placement	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR030 – The robot decides which baggage to remove with least disturbance to surrounding baggage	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR045 – Place momentarily the boxes on its equipped table or another interim storage position		RGBD cameras	OpenCV, PCL, Camera API
		SFR057 – Place the luggage within the boundaries of the belt/cart	real-time	RGBD cameras	OpenCV, Camera API
		SFR063 – Ability to avoid collision with moving or static objects	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR064 – Ability to avoid collisions with itself in bimanual tasks	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR065 – Detects the optimal contact points in boxes, bags and products to grasp the item	real-time	RGBD cameras	OpenCV, PCL, Camera API, Pytorch, Python
		SFR084 – Robot stores/retrieves the bagtag information of the placed baggage in/from the database	On-line	HDD >= 500GB, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	MySQL or MongoDB

		SFR088 – Keep count of the number of baggage loaded	On-line	HDD >= 500GB, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	MySQL or MongoDB
		SFR089 – Scans the bag tag (when placing the baggage) for BTRS (Baggage Tracking and Reconciliation System) purposes	real-time	RGBD cameras	OpenCV, Camera API
		SFR016 - Recognize the luggage material (soft or hard)	real-time	RGBD cameras, GPU: Nvidia with min memory 12GB at 15000MHz, minimum of 3000 CUDA cores	OpenCV, PCL, Camera API, Pytorch, Python
		SFR050 - Grasp the luggage that made from soft material gently	real-time	RGBD cameras, 2D/3D LiDAR	OpenCV, PCL, Camera API,
		SFR055 – Place the luggage made of hard material below the luggage made of soft material	real-time	RGBD cameras, 2D/3D LiDAR	OpenCV, PCL, Camera API,
		SFR056 - Place the luggage with a distance between them on the reclaim belt for the arrivals	real-time	RGBD cameras, 2D/3D LiDAR	OpenCV, PCL, Camera API,
		SFR058 - Handle luggage only for the flight that has been allocated to	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR060 - Ability to grasp all types of products within the test sample safely using a versatile gripping mechanism	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR080 - Interaction interface alerts when the bag it is manipulating is out of its capabilities (too heavy, too long, no handles, out of reach etc.).	real-time		Web browser, Javascript, Angular, NodeJS,

		High Priority	Technical Specifications		
		Medium Priority			
		Low Priority			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC4.4 Moving to next Baggage cart	*UR1, UR2, UR5, UR17, UR19*	SFR031 - Platform 2D/3D localization	real-time, < 10 cm (RPE)	2D/3D Lidars, RTK-GPS	
		SFR032 - Platform Path planning ability	real-time		
		SFR033 - Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras, GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores,	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR034 - Ability to move on the pavement area alongside the conveyor belt	real-time	2D/3D Lidars, RGBD cameras	
		SFR066 – Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		SFR09 - Detect the appropriate baggage cart	real-time	RGBD cameras	OpenCV, PCL, Camera API
		SFR028 - Recognize and track humans' trajectory	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	OpenCV, PCL, Camera API, Pytorch, Python
		SFR036 - Navigate avoiding restricted areas	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		SFR082 - The robot alerts the operator if no other cart is found, goes in standby mode and awaits input from operator.	real-time		Angular, NodeJS, Javascript, Web browser

		High Priority (🟥)	Technical Specifications		
		Medium Priority (🟦)			
		Low Priority (🟩)			
Sub Use Case	**Related Requirements**	**System Functional Requirements**	**Performance Requirements**	**Hardware Requirements**	**Software Requirements**
SubUC4.5 Movement back to resting position/charging point	*UR1, UR2, UR17, UR31*	🟥 SFR031 - Platform 2D/3D localization	real-time, < 10 cm (RPE)	2D/3D Lidars, RTK-GPS	
		🟥 SFR032 - Platform Path planning ability	real-time		
		🟥 SFR033 - Platform human-aware navigation	real-time	2D/3D Lidars, RGBD cameras, GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores,	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟥 SFR034 - Ability to move on the pavement area alongside the conveyor belt	real-time	2D/3D Lidars, RGBD cameras	
		🟥 SFR066 – Robot pauses its current task if employee or customer is too close to it	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		🟦 SFR028 - Recognize and track humans' trajectory	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	OpenCV, PCL, Camera API, Pytorch, Python
		🟦 SFR036 - Navigate avoiding restricted areas	real-time	GPU: Nvidia with min memory 12GB at 15000Mbps, minimum of 3000 CUDA cores, RGBD cameras	Python, PyTorch, Tensorflow, OpenCV, PCL, Camera API
		🟦 SFR069 - Decides from where it can leave the pavement track (either because it has been instructed to by the operator or if it needs to be charged)	real-time	2D/3D Lidars, RGBD cameras, Proximity Sensors	OpenCV, PCL, Camera API
		🟩 SFR035 - Ability to step up to the pavement area (height 15cm) around the conveyor belt (with or without the use of a ramp)	<= 15cm height, > 5° angle	2D/3D Lidars, RGBD cameras	OpenCV, PCL, Camera API