A Modular Architecture for Autonomous Robotic Logistics in Semi-Structured Environments

Xanthi S. Papageorgiou*, Anastasia Dimitra Lipitakis, Dimitris Kavroulakis, Thanos Giannakopoulos
Robotics & Cognitive Systems Unit, UBITECH, Limassol, Cyprus
{xpapageorgiou, dlipitaki, dkavroulakis, tgiannakopoulos}@ubitech.eu

***Abstract*—In the context of recent developments in logistics, the efficient and flexible deployment of autonomous robotic systems remains a significant challenge, particularly in semi-structured, flexible environments, typically encountered in small residences, warehouses, or medium-sized industrial facilities, where there is minimal potential for infrastructural and/or procedural enhancements to facilitate robotic automation. Such environments pose a significant problem to automated solutions, since the environment is flexible and partially unknown, cluttered and human-center. This paper presents a modular architecture for autonomous robotic logistics designed to enhance operational efficiency through adaptive control, real-time decision-making, and IoT integration. The proposed system employs a hierarchical architecture that separates high-level task planning from low-level motion control, facilitating scalability and simplifying task management. The architecture supports multiple autonomous robots capable of dynamic task allocation, path planning, and predictive control to improve reliability and minimise task execution time. This work contributes to the development of a novel class of robotic logistics systems that are capable of operating in semi-structured environments. These systems are distinguished by their ability to combine the advantages of traditional large-scale automation solutions with the flexibility and cost-effectiveness of small-scale robotic systems.**

***Index Terms*—Smart Autonomous Robotic Systems, Autonomous Multi Robot Transportion Systems, Modular Architecture, Automated Logistics**

I. Introduction

Many companies/organizations accommodated in small buildings involve a vast amount of material transport through hallways, elevators, basements, and customer/patient units. Logistic transporting robots are used in big hospitals, malls, and industrial areas, but there is no cost-effective autonomous logistic robotic system adapted to small residences, warehouses, or medium-sized industrial facilities. In healthcare institutions, several employees spend many man-hours transporting food, clothes, waste, and other materials. Although automation solutions exist for large scale structured environments, they do not exist for small buildings or semi-structured environments.

The goal is to develop and integrate an intelligent robotic indoor delivery system for wholesalers for pharmaceuticals, small residences, hotels, supermarkets/warehouses, and other small semi-structured buildings, that enhances operational efficiency in logistics without requiring extensive infrastructure modifications. The aim of this work is to bridge the gap between traditional warehouse automation and flexible, small-scale robotics solutions.

The main objective of this work is to develop a modular and scalable architecture that supports multiple autonomous robots for logistics tasks. This architecture ensures real-time decision making for task allocation, path planning, and navigation, as well as reliability and fault tolerance through adaptive task reassignment and predictive control. Also, based on this architecture, manual intervention in repetitive delivery tasks is reduced, enhancing productivity and safety.

II. Related Work

In recent years, the use of autonomous vehicles and robotic systems has expanded across various fields, driven by advancements in robotics and automation. As these systems are deployed in increasingly complex and dynamic environments, safety and flexibility have become critical design priorities, [1], [2]. To address these challenges, new architectures must emphasize real-time capability and modularity, enabling systems to adapt to diverse scenarios while maintaining robust performance. This has led to a growing demand for generic modular architectures that offer high flexibility, ensuring that robotic systems can be easily reconfigured or upgraded as requirements evolve.

One of the most complex challenges in robotics is Task and Motion Planning (TMP), which involves planning for robots operating in environments with numerous objects, requiring actions to navigate and manipulate their surroundings. TMP combines elements of discrete task planning, mathematical programming, and continuous motion planning, making it a highly demanding problem that cannot be effectively addressed by any single field alone, [3]–[6]. Task Planning, in particular, involves translating high-level specifications into low-level commands for the robot, a process that is both intricate and computationally intensive.

To address these challenges, researchers have proposed various solutions. For instance, an agent-based task management architecture has been developed to simplify the integration of intelligent robots, acting as middleware that provides a unified control view for distributed functional components, [7]. Similarly, robot task design and management systems have been created to streamline tool manipulation tasks, offering intuitive graphical interfaces for task description and execution, [8]. These advancements highlight the importance of modular and scalable architectures in managing the complexity of robotic systems.

However, the growing scale and complexity of modern software projects present significant challenges for software architectural designers. The architectural design phase, which occurs early in the development process, is critical in shaping the system's ability to meet performance, safety, and flexibility requirements. Rather than focusing solely on detailed design decisions, designers must prioritize accurately describing real-world problems and translating them into clear system requirements. By adopting modular approaches, they can create architectures that address the intricate demands of TMP, ensuring scalability and adaptability for future applications in autonomous robotics, [9].

To this end, the main purposes of a conceptual architecture, [9]–[13], are: (i) To capture and facilitate analysis of the system in the context of its environment, incorporating key product features, requirements, and essential domain knowledge. (ii) To include information about the structural, behavioural, and functional characteristics of the product and avoid premature design decisions. (iii) To specify the major components and their interactions with the environment so that designers can identify conflicting requirements and non-functional constraints (e.g.embedded legacy systems, domain critical hardware etc.). (iv) To map each component to an environmental actor, a physical component, a software component, a legacy system, a domain concept, or a perceptible environmental action. The behaviours of the model can be traced to use case scenarios that match major features of the requirements. (v) To be modeled in a clear hierarchical structure with each level properly abstracted.

In this work, we propose a modular and scalable architecture for an intelligent multi-robotic autonomous delivery system designed to enhance operational efficiency in logistics. Our architecture is built to address the growing demands of modern supply chains, where speed, flexibility, and reliability are critical. By leveraging modular design principles, the system can be easily adapted to various operational environments and scaled to accommodate increasing workloads or additional robotic agents. The architecture integrates advanced capabilities such as real-time task allocation, dynamic path planning, and collaborative decision-making, enabling multiple robots to work together seamlessly in complex, semi-structured environments. This approach not only improves the efficiency of delivery operations but also ensures robustness and adaptability, making it suitable for a wide range of logistics applications, from warehouse management to last-mile delivery.

III. General Conceptual Architecture

Modern autonomous systems require a robust architecture integrating multiple components while maintaining flexibility and scalability, [14]. The architecture must address key challenges: dynamic task allocation, real-time robot coordination, and environmental adaptability. It provides a structured framework for system design and implementation through interconnected components and algorithmic modules.

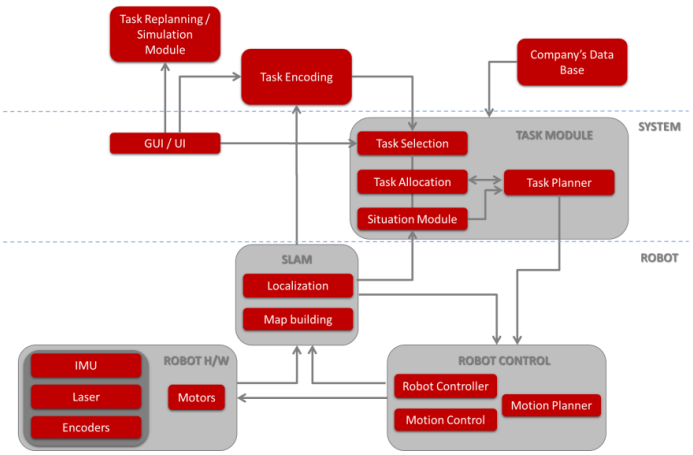

Fig. 1. The conceptual architecture of the system.

In Fig. 1, the conceptual architecture of the system is illustrated, which is composed of three primary elements: **System Components, System-Level Algorithmic Components, and System Operation**. These elements work cohesively to enable efficient task execution and adaptiveness in dynamic environments.

IV. Strategy View

The Strategy View presents the comprehensive conceptual framework that governs the system's architecture, emphasizing its multi-layered organization and the seamless integration of human stakeholders with advanced technological components. As depicted in Fig. 2, the architectural design establishes a well-defined hierarchy that originates from strategic stakeholder requirements and cascades through operational processes down to the technological enablers of autonomous delivery. This sophisticated yet flexible structure combines human oversight with autonomous robotic operations through carefully designed interfaces and control mechanisms.

At the base of the system are the **Facilities**, which represent the physical ecosystems where **Mobile Robots** perform their delivery functions. These robotic assets operate under an **assignment relationship** with **Robotic Solutions**, a critical layer that provides the intelligence and control capabilities. The **Robotic Solutions** layer incorporates two fundamental subsystems: the **Perception Modules** that process sensor data for environmental understanding, obstacle detection, and dynamic change recognition; and the **Control Modules** that transform delivery requirements into precise robotic movements and actions.

These technological components directly support the **Autonomous Robotic Delivery Process**, the system's core operational workflow. This process integrates multiple coordinated functions including intelligent **Task Management** for optimal scheduling, adaptive **Tasks Replanning & Encoding** for dynamic adjustments, reliable **Executing Tasks** for physical delivery operations, and comprehensive **Tasks Supervision** for continuous monitoring. The system delivers two primary

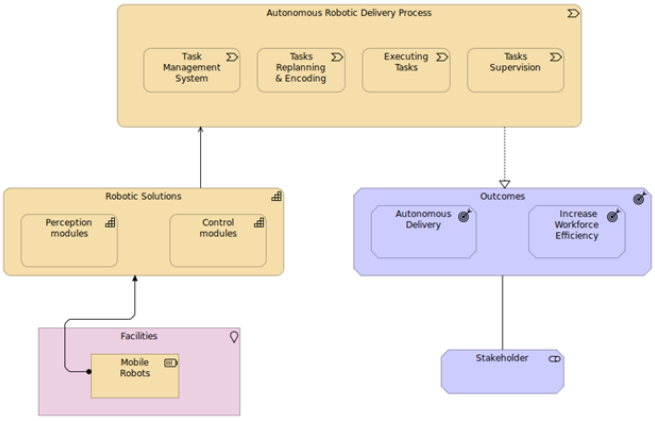

Fig. 2. Strategy Layer of conceptual architecture.

outcomes: fully **Autonomous Delivery** capabilities that operate without human intervention, and measurable **Increased Workforce Efficiency** by automating routine tasks, both of which directly address stakeholder expectations through a clear **realization relationship**.

V. Business Layer

The Business Layer, illustrated in Fig. 3, defines the organizational processes, roles, and interactions that ensure both efficient delivery operations and robust security management. This layer serves as the crucial link between strategic business objectives and technical implementation, featuring sophisticated coordination mechanisms between human operators and automated systems.

Central to this layer are the **Business Actors**, particularly the Delivery Supervisors who oversee the **Autonomous Delivery Process** and **Emergency Officers/Security Administrators** who manage the **Security Management Process**. These roles interact with the system through specialized interfaces that enable informed decision-making and exception handling.

The **Autonomous Delivery Process**, implemented through the **Autonomous Delivery Service**, follows a complete workflow from initial Delivery Approval (triggered either by human operators or automated systems), through intelligent **Task Management** that optimizes allocation and scheduling, adaptive **Task Replanning** that responds to operational changes, physical **Delivery Execution** by the robotic fleet, to continuous **System Monitoring** that ensures performance and compliance. Each phase is supported by dedicated subsystems and data flows that maintain operational integrity.

Parallel to delivery operations, the **Security Management Process** provides comprehensive safety assurance through its **Incident Response** mechanisms. Triggered by various inputs including the **Emergency Button** or automated anomaly detection, this process enables rapid response to emergencies or security breaches. The **Security/Safety Compliance Service** underpins this process with specialized tools for incident reporting, response coordination, and compliance tracking.

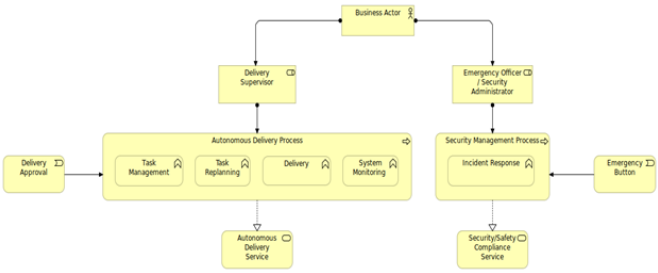

Fig. 3. Business Layer of conceptual architecture.

VI. Application Layer

The application layer integrates business workflows with technical execution, enabling predictive maintenance and operational management. As shown in Fig. 4, it combines process management, task execution and security monitoring through coordinated services and components.

This layer interacts seamlessly with the Business Layer through the **Delivery Management Interface**, ensuring alignment between business oversight and operational execution. The **Delivery Supervisor** role in the Business Layer connects to the Application Layer via the **Delivery Management Interface**, which acts as a bridge to serve the **Delivery Management Component**. The **Monitoring System Component** includes several critical subcomponents that enable efficient management and execution of delivery tasks by consolidating real-time system data. This component ensures that delivery and security operations are continuously assessed for performance and compliance. The **Monitoring System Component**, within the **Delivery Management Component**, is intricately connected to the **Monitoring System Process**, which includes several key monitoring sub-processes. These sub-processes provide real-time insights and data integration to ensure the efficient functioning of both delivery and security operations. The Monitoring System Process encompasses the following sub-processes:

1. **Monitoring the Completed Tasks Process:** Tracks and verifies the completion of assigned tasks, ensuring that all delivery goals are met according to plan. It has a **realization relationship** with the **Task Replanning Service**, which utilizes the data to refine and dynamically adapt task plans. This implementation relationship ensures that real-time monitoring data is used to dynamically adapt and refine task plans to address any deviations or unexpected conditions in the operational environment. This sub-process is also connected to the Business Layer through the **Task Replanning** function, ensuring that completed tasks inform future planning and scheduling.

2. **Monitoring the Status of Robots Process:** Observes the operational status of robots, identifying any potential issues or downtime in real time. It has a **realization relationship** with the **Monitoring the Status of Robots Service**, enabling continuous robot performance evaluation. Additionally, it is connected to the Business Layer through the **System Moni-**

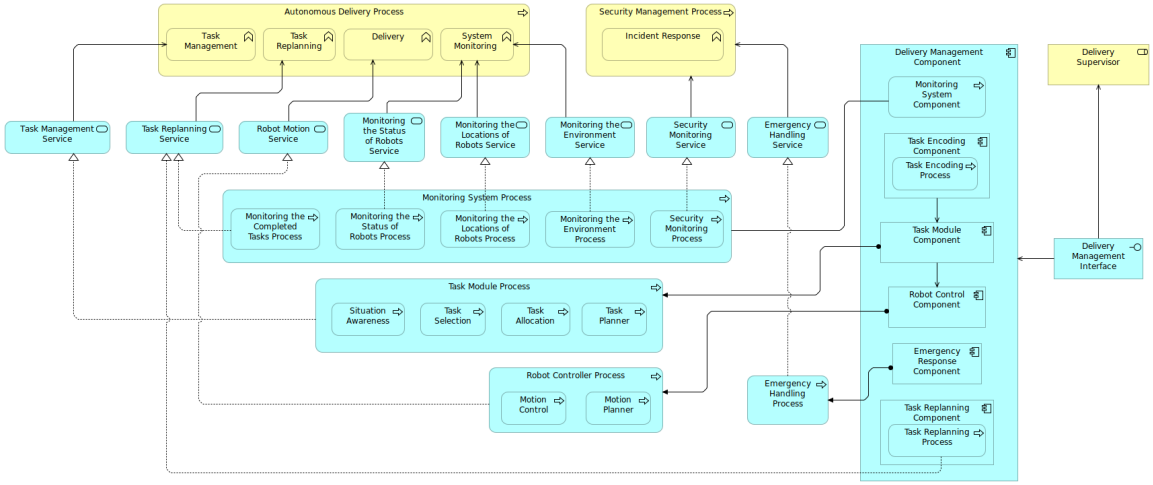

Fig. 4. Application Layer of conceptual architecture.

toring function, providing stakeholders with a comprehensive view of the robots' operational health.

3. **Monitoring the Locations of Robots Process:** Tracks the geographic and operational locations of robots, enabling effective coordination and spatial planning. It has a realization relationship with the **Monitoring the Locations of Robots Service**, which ensures accurate location tracking. Furthermore, it is connected to the Business Layer through the **System Monitoring** function, supporting high-level oversight and decision-making.

4. **Monitoring the Environment Process:** Evaluates the environment for dynamic changes, such as obstacles or safety hazards, that might impact delivery operations. It ensures the system remains adaptable and responsive to environmental changes.

5. **Security Monitoring Process:** Oversees security-related events, such as potential breaches or safety violations, ensuring compliance with security protocols. It has a **realization relationship** with the **Security Monitoring Service**, which enables real-time security evaluation. The Security Monitoring Process is also directly connected to the Business Layer through the **Security Management Process**, ensuring that security concerns are integrated into high-level operational management.

These monitoring sub-processes work together to provide a holistic view of the operational and security status of the system. For instance, if the **Monitoring the Completed Tasks Process** detects delays or failures, the **Task Replanning Service** can immediately adjust task schedules to maintain operational efficiency. Similarly, insights from the **Monitoring the Status of Robots Process** and the **Monitoring the Locations of Robots Process** allow the **System Monitoring** function in the Business Layer to ensure that robots are functioning optimally and are well-coordinated. The **Security Monitoring Process** feeds critical security data into the **Security Management Process**, enabling swift and effective responses to potential threats.

The integrated feedback between monitoring processes, Task Replanning Service, and Business Layer ensures system efficiency, reliability and security in dynamic environments. This synergy of real-time monitoring, adaptive replanning and strategic management maintains consistent performance while meeting all security and business requirements.

The **Task Encoding Component**, through its **Task Encoding Process**, translates high-level tasks from the Delivery Supervisor into formats that can be processed and executed by the system. This, in turn, activates the **Task Module Component**, which orchestrates task allocation, planning, and execution. The **Task Module Component** is a central part of the delivery workflow, managing the flow of information and instructions to the **Robot Control Component**. The **Robot Control Component** handles real-time robotic actions, ensuring precise movement and task execution.

The **Task Module Component** within the **Delivery Management Component** is activated by the Task Module Process, which includes a set of key processes that ensure efficient task planning and execution. These processes include:

1. **Situation Awareness:** Gathers contextual information about the operational environment to provide a comprehensive understanding of current conditions.

2. **Task Selection:** Prioritizes tasks based on their importance and urgency, selecting the most critical ones for immediate execution.

3. **Task Allocation:** Assigns tasks to the appropriate robots or resources, ensuring optimal utilization of available assets.

4. **Task Planner:** Develops detailed plans for the execution of tasks, integrating situational data and operational constraints.

The **Task Module Process** has a **realization relationship** with the **Task Management Service**, enabling it to efficiently manage task allocation, prioritization, and execution. This connection ensures that all tasks are dynamically managed and adapted to changing operational requirements. Similarly, the **Robot Control Component** interacts with the **Robot**

Controller Process, which includes:

1. **Motion Control Process:** Directs the robotic actuators, ensuring precise execution of movements required to carry out assigned tasks.

2. **Motion Planner Process:** Optimizes routes and actions for robots, considering operational constraints and environmental conditions.

The **Robot Controller Process** has a **realization relationship** with the **Robot Motion Service**, which directly supports the Delivery function in the **Autonomous Delivery Process**. This connection ensures that robots can execute delivery tasks efficiently and navigate dynamic environments with precision.

In addition, the **Emergency Response Component** within the **Delivery Management Component** interacts with the **Emergency Handling Process**, which oversees the response to critical incidents, faults, or emergencies. This process has a **realization relationship** with the **Emergency Handling Service**, ensuring that security incidents and emergencies are addressed promptly and effectively. The **Emergency Handling Service** is closely linked to the **System Monitoring** function in the **Autonomous Delivery Process** and the **Security Management Process** in the Business Layer, creating a comprehensive framework for managing both operational and security challenges.

Furthermore, the **Task Replanning Component** within the **Delivery Management Component** also has a direct realization relationship with the **Task Replanning Service**, enabling dynamic adjustments to plans based on monitoring data. This feedback loop between the monitoring sub-processes, task replanning and the Business Layer ensures that the system remains efficient, reliable, and secure, even in dynamic or unpredictable scenarios.

VII. Technical Layer

The technical layer forms the infrastructure supporting both application and business layers, integrating hardware, software, and services for autonomous navigation, task execution, and system monitoring. As shown in Fig. 5, it comprises essential components and nodes that enable core system functionalities.

The **Infrastructure Node** forms the technological foundation, integrating software and hardware components to support system operations. Built on **Linux** for stability and scalability, it utilizes the **Robot Operating System (ROS)** for core robotic functionalities including motion planning, sensor integration, and task execution. **Rosbridge** enables web-ROS integration, while visualization tools (**GzWeb**, **RViz**) provide real-time monitoring capabilities. The development stack includes **React** and **Django** for interface and backend services, connected to the Delivery Management Interface. A central database maintains critical operational data (task statuses, robot positions, environmental conditions) ensuring data persistence and accessibility.

The **Robot Hardware (Robot H/W)** represents the physical layer of the system, comprising essential components such as **Motors**, which facilitate robotic movement under the control of the motion control process. The **Encoders** provide feedback on motor rotations, thereby ensuring precise localisation and motion execution. The **Inertial Measurement Unit (IMU)** tracks the robot's orientation and acceleration, thereby aiding in navigation and maintaining stability. The **Light Detection and Ranging (LIDAR) Laser** generates environmental maps and detects obstacles, thereby enabling safe and accurate navigation. A physical **Button** serves as an emergency interface for handling urgent interventions or faults.

The **Hardware Abstraction Function** serves to bridge the gap between the hardware and higher-level services. Its role is to translate raw data from the motors, encoders, IMU, and laser sensors into actionable insights for the system. This abstraction is crucial for ensuring seamless communication between the hardware layer and the software-driven processes.

The **SLAM (Simultaneous Localization and Mapping)** module is of great importance in enabling autonomous operation of the robot. It integrates the functionalities required for localisation and mapping, thereby enabling the estimation of the robot's position within its environment and the generation of real-time maps. The data is then fed into the **Positional Information Service**, which consolidates the location and orientation details for use by the **Robot Controller Process**. Concurrently, the Navigation Service provides assistance to the **Navigation Process** within the **Robot Controller Process**, enabling robots to devise and implement optimal routes, even in dynamic environments.

The **Task Module Process** in the application layer is supported by the **Task Assignment Service**, which allocates tasks to robots based on operational constraints. This process ensures effective task management by dynamically adapting to real-time operational requirements.

Within the **Robot Controller Process** in the application layer, robotic movements are executed with the help of the **Positional Information Service, Navigation Service**, and hardware abstraction. The **Motion Control Process** directs robotic actuators for precise execution, while the Motion Planner Process develops efficient movement plans by integrating positional data and environmental constraints.

The **Task Encoding Process** in the application layer, converts high-level task descriptions into machine-readable formats, ensuring that tasks are seamlessly executed by the robotic systems. Meanwhile, the **Task Replanning Process**, powered by the **Replanning Service** in technical layer, dynamically adjusts task plans based on real-time monitoring data and inputs from the **Emergency Handling Process**. This adaptability allows the system to maintain operational continuity and efficiency, even in the face of unexpected changes. The **Emergency Handling Process**, triggered by the **Emergency Button Service**, ensures a swift and effective response to security incidents or operational disruptions.

Through direct integration with monitoring and security services, the system effectively mitigates potential threats and emergencies. These interconnected components ensure reliable performance by combining real-time data analytics, robust hardware, and adaptive software to achieve operational

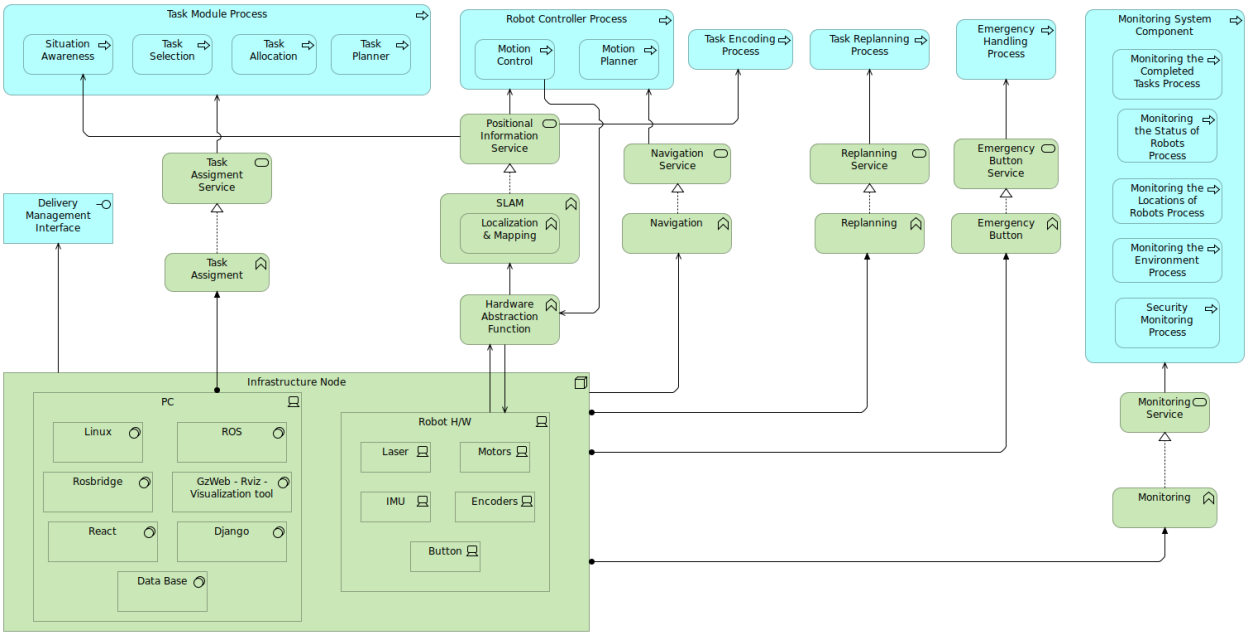

Fig. 5. Technical Layer of conceptual architecture.

objectives.

VIII. Conclusions

In this paper, we presented a modular architecture for autonomous robotic logistics, designed to enhance operational efficiency in semi-structured, flexible environments. The proposed system employs a hierarchical structure that separates high-level task planning from low-level motion control, enabling scalability and simplifying task management. Our solution effectively addresses the unique demands of dynamic, human-centered environments, such as small residences, warehouses, and medium-sized industrial facilities, without requiring significant infrastructural changes. This work contributes to a novel class of robotic logistics systems that combine the scalability of traditional automation with the flexibility and cost-effectiveness of small-scale robotic solutions.

Acknowledgment

This work is a part of "Smart Autonomous RObotic deliveries for small loGIstics" – AROGI, ENTERPRISES/0223/SubCall1/0117 project, funded by the European Union Recovery and Resilience Facility of the NextGenerationEU instrument, through the Research and Innovation Foundation.

References

[1] T. Schöpping, T. Korthals, and M. Hesse, "Generic architecture for modular real-time systems in robotics," 01 2018, pp. 413–420.

[2] A. D. Hristozov, E. T. Matson, J. C. Gallagher, M. Rogers, and E. Dietz, "Resilient architecture framework for robotic systems," in *2022 International Conference Automatics and Informatics (ICAI)*, 2022, pp. 18–23.

[3] C. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, 05 2021.

[4] F. Lagriffoul, N. T. Dantam, C. Garrett, A. Akbari, S. Srivastava, and L. E. Kavraki, "Platform-independent benchmarks for task and motion planning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3765–3772, 2018.

[5] M. M.G and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, 12 2017.

[6] G. Rajendran, U. V, and B. O'Brien, "Unified robot task and motion planning with extended planner using ros simulator," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7468–7481, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821001695

[7] J. Lee and B. Kwak, "A task management architecture for control of intelligent robots," 08 2006, pp. 59–70.

[8] K. Kanayama, M. Mizukawa, S. Iwaki, S. Matsuo, T. Okada, and Y. Nakamura, "A robot task design and management system for industrial applications," in *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97*, 1997, pp. 687–692.

[9] H. Liu and D. Gluch, "Conceptual modeling with the object-process methodology in software architecture," *Journal of Computing Sciences in Colleges*, vol. 19, pp. 10–21, 01 2004.

[10] A. Mahmood and F. Montagna, "System of systems architecture framework (sosaf) for production industries," 07 2012, pp. 543–548.

[11] P. Bernus, O. Noran, and A. Molina, "Enterprise architecture: Twenty years of the geram framework," *Annual Reviews in Control*, vol. 39, pp. 83–93, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578815000097

[12] P. Pelliccione, E. Knauss, R. Heldal, S. M. Agren, P. Mallozzi, A. Alminger, and D. Borgentun, "Automotive architecture framework: The experience of volvo cars," *Journal of Systems Architecture*, vol. 77, pp. 83–100, jun 2017. [Online]. Available: https://doi.org/10.1016/j.sysarc.2017.02.005

[13] A. Almeida da Costa Júnior, S. Misra, and M. Soares, *A Systematic Mapping Study on Software Architectures Description Based on ISO/IEC/IEEE 42010:2011*, 06 2019, pp. 17–30.

[14] X. S. Papageorgiou, D. Kavroulakis, D. Ntalaperas, and T. Bouras, "A task restricted hierarchical control scheme facilitating small logistics," in *2nd Workshop on Mobile Manipulation and Embodied Intelligence at ICRA 2024*, 2024. [Online]. Available: https://openreview.net/forum?id=uAxow4DkZi