VOLUME 20, N° 1 2026
Journal of Automation, Mobile Robotics and Intelligent Systems

A Flexible Mobile Manipulator Architecture: A Case Study on Plasterboard Wall Preparation

Submitted: 6th November 2024; accepted: 24th February 2025

Łukasz Granat, Michał Bryła, Kuba Kamiński, Filip Jędrzejczyk, Sławomir Puchalski, Ingmar Kessler, Alexander Perzylo, Ángel Soriano

DOI: 10.14313/jamris-2026-002

Abstract:
Skilled labor shortage and a competitive market are challenges for many businesses. Hence, automation and robotization are well-established in the large-scale, well-structured industrial environments of many factories. Moreover, there is a push for safer and more easily programmable robots, so-called cobots, that would be more suitable for smaller-scale applications. However, the implementation and customization of complex robot systems with different sensors, tools, external systems, etc. in particular for the use cases of small and medium-sized enterprises (SMEs) are still often infeasible. Therefore, this paper describes the design, implementation, and testing of a flexible mobile manipulator architecture with the following three abstraction layers. A high-level semantic layer models and utilizes OWL ontologies of abstract manufacturing processes, specific workcell environments, linked context knowledge, and the current semantic world state. An intermediate translation layer collects and exchanges data from multiple system components via a unified internal database, which is used, e.g., in combination with behavior trees to convert symbolic, high-level actions into multiple parameter-driven low-level commands. A low-level control layer is implemented locally on a mobile robot to provide a unified interface of its potentially customized subsystems to the higher layers. This architectural approach facilitates the implementation of new use cases, e.g., by a system integrator, via flexibly adapting semantic representations and behavior trees on the model level without changing source code. The implemented system was deployed in five different real-world use cases of industrial partners in the VOJEXT project. This paper focuses on the use case of plasterboard wall preparation from the construction domain, which includes taping, spraying, and sanding operations, to evaluate the particular requirements of this application and how they can be met by the proposed robot control architecture.

Keywords: *Knowledge-based system, Interface and translation layer, Unified robot control, Semantic process descriptions, Reusable models, Use case adaption, Hardware and software abstraction, Human working environments, Building construction domain, Mobile manipulation*

1. Introduction

In the era of the Industry 4.0, automation of the production and handling of goods is a necessity, as the competitive market requires constant cost optimizations and efficiency improvements. Moreover, the labor market has been experiencing a shortage of skilled labor workers [6]. This situation encourages an increase in automation even for enterprises where robotization is not present or poorly developed to have robots or other machines perform at least parts of workers' activities. However, robots are perceived as complicated and potentially expensive to introduce to businesses, and for some sectors, the presence of qualified workers is still required during production processes, as it is very difficult to fully automate them.

In some sectors, like food delivery and logistics, there is a movement towards affordable automation [21], but many others, such as the construction domain, do not have easily available solutions for their type of operation. Multiple funding opportunities are available to modernize those enterprises, but without well-established system designs and paradigms, modernization progresses slowly.

To tackle this problem, highly flexible and modular solutions are needed, that are able to adjust to specific needs in different industrial sectors. This issue was addressed by the *Value Of Joint EXperimentation in digital Technologies for manufacturing and construction* (VOJEXT) project (https://vojext.eu), which aimed at providing a solution that uses robots, a variety of sensors, perception models, and advanced control systems to enable dynamic decisions made by cognitive subsystems and interaction with human workers. The main targets of the created solution are non-robotic companies, including both small artisan workshops and large multidisciplinary enterprises, as it allows to support qualified workers in performing their tasks by taking part in multiple different processes. Through boosting the workers' productivity, it may help to mitigate the labor gap and make businesses more competitive in the market. Additionally, use of mobile manipulators provides more versatility in human working environments than immobile solutions, such as robotic workcells or stationary machines.

There are multiple commercial solutions that aim to facilitate the deployment of robotic systems by providing a unified interface for controlling compatible hardware and planning the execution of tasks by defining sequences of atomic actions, such as Siemens SIMATIC Robot Integrator with SIMATIC Robot Library [22], ArtiMinds RPS [3], or Intrinsic Flowstate [11]. These kinds of tools usually provide a wide range of ready-to-use solutions for most common scenarios, which can be adapted to a particular scenario using dedicated GUIs.

However, the use of such tools is usually limited to compatible hardware, other devices cannot be easily integrated. Moreover, most commercial solutions are based on proprietary, monolithic closed software that cannot be easily extended by end users. Another issue is the lack of support for reasoning, i.e., rather than setting a goal for the system along with providing a set of knowledge about a process, an end user has to define all of the activities performed by the system in a step-by-step manner. This approach forces highly specialized solutions that focus on specific processes and cannot be easily transferred to different use cases. Moreover, it may require specialized knowledge and become overwhelmingly complex in a system with multiple assets.

As none of the existing solutions fit all of the requirements of the VOJEXT project, i.e., a flexible robotic system that is able to collaborate with a human operator, capable of reasoning, independent of a hardware vendor, and deployable in various scenarios, a novel solution was developed, based on the project partners' experience in their respective fields. It consists of three main layers: a knowledge-augmented high-level semantic layer, an intermediate control and translation layer, and a low-level mobile robot control layer. They enable the planning and execution of mobile manipulation tasks next to human workers based on formally represented automation knowledge, sensor data, and operator instructions, e.g., via hand gestures or interactions with a web-based human-machine interface (HMI).

The proposed control architecture was implemented, integrated with the rest of the VOJEXT system, and tested in multiple distinct use cases:

1) Handling and quality assurance of foam pillows (https://youtu.be/tyyFJBAa44Q)
2) Transport and collaborative handover of objects for logistics while connected to a factory system (https://youtu.be/-Ppr0rwjU0A)
3) Sensor data collection for quality assurance in an automotive OEM factory (https://youtu.be/PLMMWsvYgJc)
4) Plasterboard wall preparation in the building construction domain (https://youtu.be/PLMMWsvYgJc)
5) Support of workers in artisan floor tiles production (https://youtu.be/pdlQuxJiEbM)

Moreover, some of the control system modules were tested for other activities such as welding metal plates, pick-and-place operations in a carpenter shop, and mosaic assembly.

This paper presents the multi-layer architecture of the developed control system and its benefits, and assesses its applicability to the use case of robot-based plasterboard wall preparation.

It is structured as follows. Section 2 provides an overview of the system architecture and the following Sections 3, 4, and 5 describe the three essential abstraction layers of our design in more detail.

Section 6 discusses the characteristics of robots compatible with the requirements of the guiding use case of this work, explains the required functionalities of robots to work effectively with the presented solution, and provides several examples. Section 7 describes the experiments carried out as part of the plasterboard wall preparation use case, and discusses the intricacies of a deployment of our proposed solution in this domain. Finally, Section 8 concludes this work.

2. System Architecture

This work aims to enable complex autonomous robot systems for manufacturing and construction. The proposed approach was realized based on the system architecture depicted in Fig. 1, which consists of three abstraction layers to enable separation of concerns. These layers focus, respectively, on semantic knowledge modeling and reasoning, intermediate control and translation, and unified interfaces to both general hardware components as well as custom ones developed for specialized use cases. Each layer has a primary module with its own scoped area of activities, encapsulated level of details, and generalized interfaces of interaction.

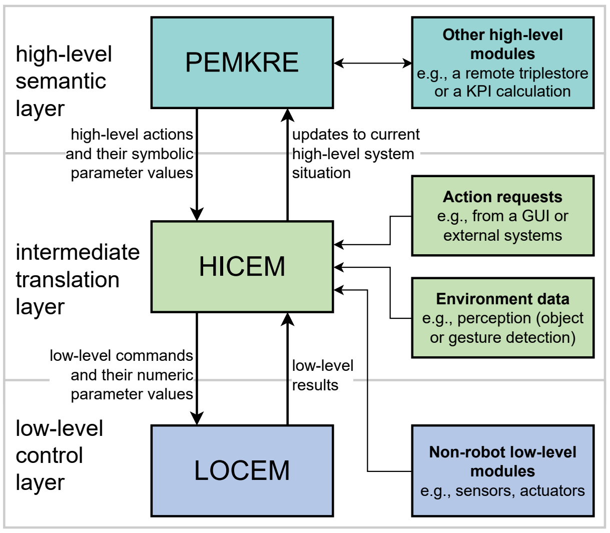
high-level semantic layer
PEMKRE
Other high-level modules e.g., a remote triplestore or a KPI calculation
high-level actions and their symbolic parameter values
updates to current high-level system situation
Action requests e.g., from a GUI or external systems
intermediate translation layer
HICEM
Environment data e.g., perception (object or gesture detection)
low-level commands and their numeric parameter values
low-level results
low-level control layer
LOCEM
Non-robot low-level modules e.g., sensors, actuators

Figure 1. Overview of the system architecture with its three abstraction layers and their primary modules PEMKRE, HICEM, and LOCEM

The high-level semantic layer contains the PEMKRE (Planning, Execution and Management Knowledge Reasoning Engine), which uses Semantic Web technologies such as OWL ontologies [24] and SPARQL requests [25].

It models manufacturing and construction processes by combining and parameterizing high-level actions similarly to flow charts as well as the current semantic world state of the robot and its work area. On request, it provides the next high-level action and its symbolic parameter values to the HICEM (High-level Control Engine Module), while being agnostic to how they are implemented in the lower layers, and enables the modeling of new use cases in the OWL ontologies without having to change any of the PEMKRE's source code. For example, in the semantic layer, new instances of existing high-level action types could be combined in an ontology-encoded process to solve a requested use case without having to change the lower layers. Conversely, the lower layers may provide an alternative implementation of an action type, as long as it adheres to the same interface type and achieves the same high-level effect. This layer may also contain additional modules, such as an embedded or remote triplestore, i.e., an RDF graph database [5], to persistently store, query, and update the OWL ontologies. Another high-level module may automatically calculate key performance indicators (KPIs) based on an executed production run's semantically logged process data in the triplestore.

The intermediate translation layer contains the HICEM, which uses one behavior tree per high-level action in combination with an internal database to translate one high-level action into multiple low-level commands to different low-level modules. During this procedure, it translates the symbolic parameters provided by the PEMKRE module to numeric ones, based on data from multiple sources, e.g., user requests from a GUI, detections from advanced perception modules, or data from external systems such as a company's database.

The low-level control layer contains the LOCEM (Low-level Control Engine Module), which internally uses different programming languages, frameworks, and middlewares to provide hardware abstraction and a consistent control interface to the higher layer, i.e., the HICEM. The LOCEM is able to focus on one subsystem, i.e., the robot with its mobile platform, articulated arm, gripper actuators, etc., and implements both atomic functions as well as specialized ones for new use cases. Hence, the LOCEM may internally include additional robot-related hardware and software components that are closely integrated into one of these functions. Examples include visual perception, e.g., of docking markers for the mobile platform, or specialized hardware integration, e.g., of a compressor for a spraying gun. The HICEM may communicate directly with non-robot low-level modules in this layer, for instance, to receive basic sensor readings from an industrial machine or to trigger production of the next part after the previous one was removed by the robot.

The following sections provide details about the primary module on each individual layer.

3. PEMKRE

The PEMKRE is directly connected to a triplestore, which persistently stores OWL ontologies that semantically model a robot system's high-level process flow, the current world state in a symbolic manner, and relevant context knowledge. These semantic description models formally represent target products, production processes, and manufacturing resources related to a domain and use case according to the PPR paradigm [2]. This provides a unified and reusable knowledge representation of their often heterogeneous data to enable autonomous robot systems [4].

The formal semantics of OWL ontologies enable the built-in reasoner of the triplestore employed in the system to automatically extend the resulting knowledge graph's set of asserted statements with a set of inferred statements. By working on this model-centric semantic level in the OWL ontologies, new use cases can be implemented without changing the PEMKRE's source code.

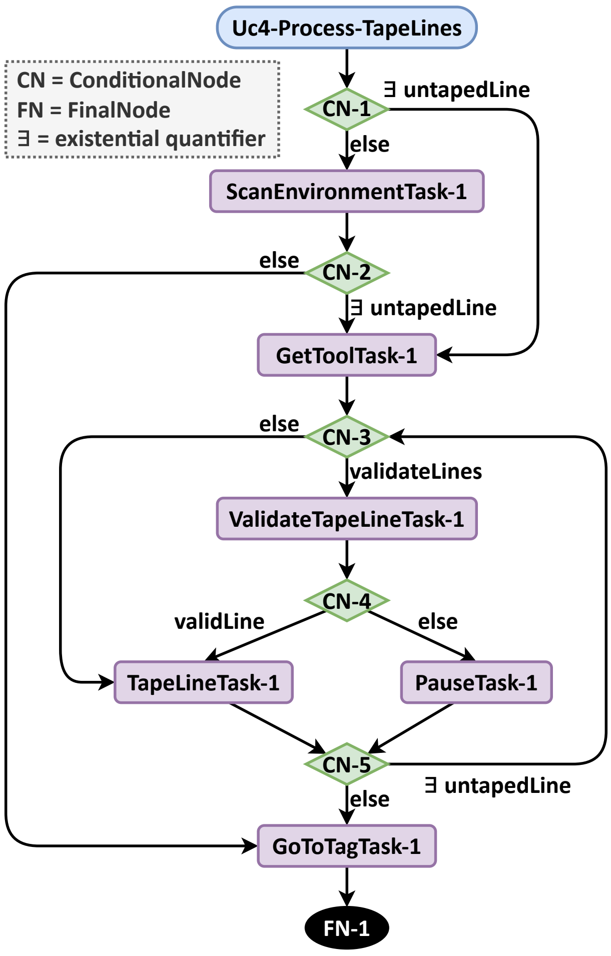

Figure 2. Visualization of an ontology-encoded abstract process with tasks, control nodes, and SPARQL-based semantic conditions for a plasterboard taping process. Parameters and semantic effects of tasks are not shown

Fig. 2 shows the task-level instances and their relations in an abstract process ontology for the first of the three parts (i.e., taping, spraying, sanding) in the plasterboard wall preparation use case (see Section 7). Such abstract processes model the workflow of a robot system similarly to flow charts, which are also intuitively understandable to non-robotics experts.

Abstract processes are abstractions that are distinct from both the current world state and the multiple executions of a single workflow over time. These abstract processes or other semantic models can be generated automatically [15] or defined in an intuitive GUI [16], but this is out-of-scope for this work. By using the semantic query language SPARQL, branching conditions are modeled within the ontology as requests to the current semantic world state of the robot system [13]. The PEMKRE evaluates them dynamically at runtime, while also mapping object-level abstract task parameters to the actual instances in the current semantic world state, to generate an executable specific process ontology.

The abstract process in Fig. 2 is about the automated taping of plasterboard wall joint lines. The other abstract processes in this use case for the spraying and sanding of the compound on the wall areas follow the same structure, only with different task and parameter types. The robot's first task in the abstract process is to scan the environment for lines and areas to add these new wall feature instances to the current semantic world state. If there are still any unprocessed wall features left in the semantic world state from previous process executions, the robot can skip this task for the moment. If there are none at all after the scanning, the mobile robot moves to a home location and the process ends early. Otherwise, the robot system gets the taping tool. From the perspective of the high-level semantic layer, the abstract process is agnostic in regard to how this is done. During initial development, the robot system may request the assistance of a human operator for this, but in the future it may instead be equipped with an automatic tool changer. After getting the tool, the abstract process enters its main loop. There, if the *validate Lines* option is set to *true*, the robot first performs a "dry run" of the taping for the next wall joint line so that a human operator can check it beforehand. If the operator indicates to the robot that the line was valid, it actually tapes the line, and if not, pauses so that the operator can tape this particular line manually. Once all lines are taped, the robot moves to a home location and the process ends.

During process execution, the PEMKRE provides several services to the HICEM. They allow to, e.g., parameterize and evaluate predefined SPARQL requests stored within the OWL ontologies, get the next high-level action and its parameter values by evaluating the ontology-encoded process, or set a new status for the current task. The PEMKRE automatically derives a high-level action's parameterization from the object-level task parameters in the corresponding specific process description and linked context knowledge.

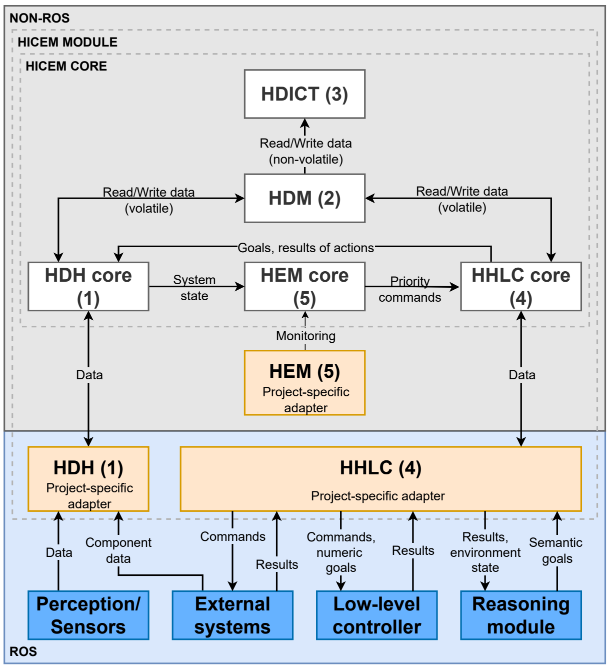

Figure 3. Internal architecture of the HICEM and the data flow between its components

It is transmitted as an N-Triples-formatted string, which also enables the HICEM and the PEMKRE's semantic description models to implement new action types without changing their common interface.

4. HICEM

4.1. HICEM Concept

The main concept of the HICEM is to interface two conceptually independent parts of a system: the high-level semantic layer and the low-level control layer (i.e., the robot subsystem and other low-level modules such as external sensors or actuators) by providing two main capabilities.

First, based on high-level decisions and factors that influence the system's state, the module can execute a low-level plan using behavior trees [9, 12], which are models used to plan and execute deterministic sequences of actions.

They are represented in a hierarchical tree, where each node defines a specific behavior. The tree is traversed from the root to the leaves, with the current state of the system guiding the flow through the nodes.

Second, the module collects important data for both layers and maintains its translatability. The semantic layer has to deal with only the data that is relevant to it, such as identifiers of objects detected by a sensor or human-readable tags for robot poses. Based on a high-level action received from it, the HICEM selects all the data that is relevant to the low-level control layer of the system, such as the pose of a detected object, and executes a configured sequence of low-level commands.

4.2. HICEM Internal Architecture

The HICEM is a ROS-based [14] implementation of a controller, which is built on top of a generic library called the HICEM Core. It is split into functional parts, as shown in Fig. 3, that provide universal mechanisms for low-level commands planning, action execution, monitoring of system health, data collection, and data storage. These components are described in the following paragraphs.

HDH, HICEM Data Handler, (1) in Fig. 3 It consists of the HICEM Data Handler Core and its adapter, which provides use case specific interfaces (e.g., ROS topics and services). It collects data from other system modules and converts it into a format that allows to store it in the HDM and HDICT. It also collects system health data and supplies it to the HEM.

HDM, HICEM Data Manager, (2) in Fig. 3 It is an internal component that handles internal data storage. It implements a simple volatile database, which stores data until the HICEM is closed and manages the non-volatile database (HDICT) by providing an interface to it for the other parts of the HICEM and organizing the data in it, e.g., by removing outdated entries.

HDICT, HICEM Dictionary, (3) in Fig. 3 It is an internal component that implements a non-volatile database to store data relevant to the HICEM and is used to preserve data between its runs. It provides an interface for the HDM to add, remove, update, and read data, independently from the database implementation. It may store, e.g., system configuration, environment description, state of the system, a high-level goal, or a feedback from a safety module.

HHLC, HICEM High Level Controller, (4) in Fig. 3 It consists of the HICEM High Level Controller Core and its adapter, which provides use case specific interfaces (e.g., ROS topics and services). It implements an interface to a high-level layer and a low-level controller. The HHLC can access the internal databases (either the volatile one in the HDM or the non-volatile one in the HDICT) via the HDM. It also contains a Behavior Tree Engine that is an implementation of the BehaviorTree.CPP library [7]. This engine provides a mechanism of low-level actions planning using behavior trees. The behavior tree models are composed of a tree of nodes, which correspond to generic or use case specific functions, each responsible for an atomic action, e.g., executing a low-level command. The behavior tree nodes often use data stored in the HDM to set the arguments of a to-be-called low-level command, change the type of command, or change a sequence of executed nodes.

HEM, HICEM Environment Monitor, (5) in Fig. 3 It is a project specific adapter of the HICEM Environment Module. It continuously reads statuses of all the modules present in the system from the HDM, processes them, and, if needed, calls the HHLC to stop the currently running behavior tree.

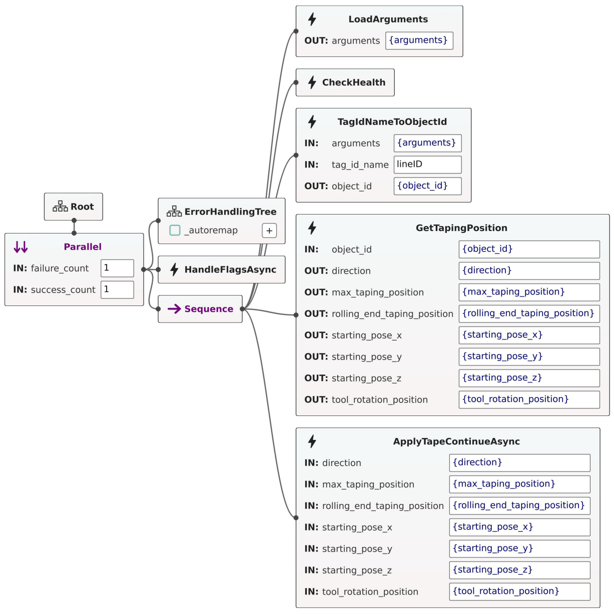
Root
Parallel
IN: failure_count 1
IN: success_count 1
ErrorHandlingTree
_autoremap
HandleFlagsAsync
Sequence
LoadArguments
OUT: arguments {arguments}
CheckHealth
TagIdNameToObjectId
IN: arguments {arguments}
IN: tag_id_name lineID
OUT: object_id {object_id}
GetTapingPosition
IN: object_id {object_id}
OUT: direction {direction}
OUT: max_taping_position {max_taping_position}
OUT: rolling_end_taping_position {rolling_end_taping_position}
OUT: starting_pose_x {starting_pose_x}
OUT: starting_pose_y {starting_pose_y}
OUT: starting_pose_z {starting_pose_z}
OUT: tool_rotation_position {tool_rotation_position}
ApplyTapeContinueAsync
IN: direction {direction}
IN: max_taping_position {max_taping_position}
IN: rolling_end_taping_position {rolling_end_taping_position}
IN: starting_pose_x {starting_pose_x}
IN: starting_pose_y {starting_pose_y}
IN: starting_pose_z {starting_pose_z}
IN: tool_rotation_position {tool_rotation_position}

Figure 4. Behavior tree of the TAPE_LINE high-level action

4.3. Example Interaction with the PEMKRE and LOCEM

As an example of the HICEM's interaction with the other two primary modules in this work, the line taping process from one of the use cases in the VOJEXT project is described in the following.

To start a process, the HICEM needs an external input, e.g., from an operator using a GUI. The input must contain the IRI (Internationalized Resource Identifier) of the process that will be started. Using the data input, the HICEM requests the PEMKRE to initialize the process in the semantic layer, and then requests the next action for the system to perform. The response from the PEMKRE, i.e., a high-level action with the required parameter values, is matched to an existing behavior tree, which is in this example the one corresponding to the line taping, that is loaded up afterwards.

The response also contains additional arguments, e.g., the ID of the robot to be used and the ID of the line to be taped, which are used as inputs for the specific behavior tree nodes. Next, the HICEM notifies the PEMKRE that the received action is in progress and executes the behavior tree.

Not all of the nodes result in the execution of low-level commands in the LOCEM, as some of them handle internal HICEM data processing. For example in Fig. 4, "CheckHealth" checks the system health and if a crucial module reports an unrecoverable error, the action is aborted. "GetTapingPosition" maps the arguments received from the PEMKRE to the numerical values required by the LOCEM, by (in this case) calculating the approach point for the robot to tape the line based on its received ID. "TagIdNameToObjectId" loads the taping arguments required by the LOCEM from the HDM.

Using the data from the initial steps, the HICEM employs the action client to send a command to the LOCEM that contains the command string and additional data. Afterwards, the HICEM waits until the LOCEM returns a response. On a success response, the behavior tree finishes as successful, and the HICEM sends to the PEMKRE that this particular action has finished as a success. Otherwise, the behavior tree finishes as a failure and the HICEM sends this result to the PEMKRE. While executing a behavior tree, HICEM's internal flags are constantly monitored, as well as external signals from a GUI, e.g., to stop the current process.

5. LOCEM

The LOCEM is an abstraction layer of software that acts as an interface between low-level robot systems and higher-level control systems. The LOCEM allows to interact with the complex control structure of robot subsystems in a simple manner.

Those subsystems can be described as a Physical Interface Module (PIM) and consist of, e.g., mobile base navigation and localization, a robotic arm movements planner, or a safety module.

An example of the dataflows in the LOCEM can be seen in Fig. 5. Commands received by the LOCEM are sent as actions via one of the interfaces supported by the LOCEM and used by the robot subsystems, which have multiple internal protocols, messages types, and formats. The atomic actions shown in Fig. 5 can be, e.g. a simple movement from point A to B, changing a digital output state, or requesting information from a sensor. This layer also has the capability to interact with multiple atomic actions using a single ROS action call. This feature is not a primary function of the LOCEM, but it may be used to simplify tasks that are always supposed to use the same set of atomic actions.

The LOCEM consists of several subcomponents that can work independently. The first one is a service interface that is designed for quick actions that do not require a long response time. The second one is an action interface that is designed for long-lasting actions that require status monitoring, e.g., for mobile base or robotic arm movements. The LOCEM also includes three more subcomponents: a procedure server interface that has similar functionality to the ROS action server interface but is designed for low bandwidth and slower-in-response atomic actions, as well as a subscriber interface and publisher interface that are used to respectfully listen to and broadcast information without confirmation.

The LOCEM accepts commands sent by the HICEM as strings with arguments that are converted by the subcomponents to ROS message types used by the corresponding robot system components. Such a message corresponds to a name specified at the beginning of the string. The order of the given arguments and available options are specified in an Application Programming Interface (API) documentation. A ROS topic-based LOCEM node provides constant status feedback during execution and information about the result of a finished command.

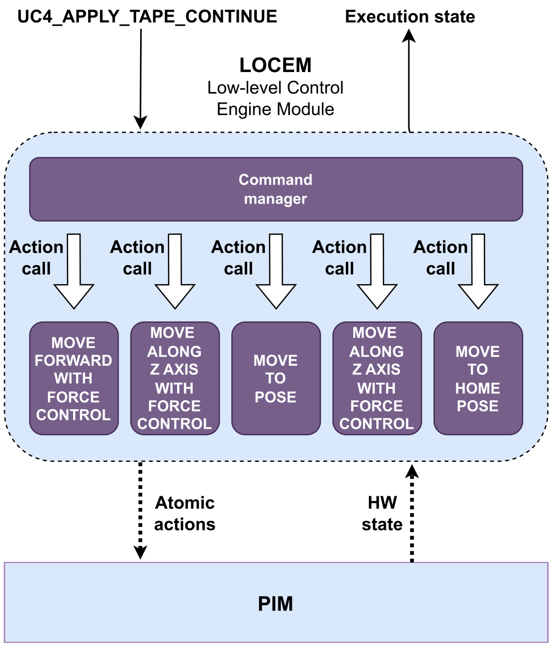

Figure 5. Invocation and monitoring of atomic actions by the LOCEM for one the higher-level actions sent by the HICEM

A ROS service-based LOCEM node provides just the result. The mentioned subcomponents were developed, validated, and demonstrated during the VOJEXT project.

The LOCEM is primarily designed to work with Robot Operating System (ROS) [15] based robot environments. In the ROS ecosystem, the types of the transferred data are essential for communication between microservices. Frequently, those types are custom, which creates a necessity to share and maintain on both ends up-to-date custom message packages for each ROS node that an external system wants to interact with. Moreover, this approach forces to expose potentially sensitive parts of a system to the public. To solve those problems, LOCEM nodes act as translator for ROS interfaces with one defined message type. This interface unification is one of the main benefits of the LOCEM as it provides simplicity of usage while preserving the safety of robot systems.

Information received by the LOCEM can be directly passed as ROS messages for further processing, such as a navigation goal for a mobile platform. However, messages can also be translated to other communication protocols, such as MODBUS for a safety system or TCP/IP for communication with a robotic arm.

This means that a target atomic action does not need to be written in a specific programming language, framework, or environment. Hence, higher-level control layers also need only one middleware regardless of the technologies used inside of a robot software.

6. Characteristics of Compatible Robots

Within the VOJEXT project, the presented architecture was proven to work correctly with robots providing the following primary features: a mobile platform with the capability to autonomously plan a path to a given target, a robotic arm with sufficient size and payload for completing the needed tasks, and safety sensors. An example of this is the RB-KAIROS+ [19], which is a mobile manipulator designed for indoor material handling. The robot consists of an RB-KAIROS [18] mobile platform with a mounted Universal Robots e-Series robotic arm. Robotic arms are usually mounted stationary next to machines or workstations, which limits their workspace to only that place. The RB-KAIROS+ allows for the expansion of the robot workspace as the robotic arm can move to multiple locations thanks to its mobile base. This enables the robot to, e.g., operate on large parts or perform pick-and-place operations in large areas. This autonomous mobile robot (AMR) utilizes omnidirectional wheels that make reaching tight spaces possible. It is equipped with safety scanners and other safety measures to comply with recent industrial standards (EN 60204-1:2006/A1:2019, EN ISO 12100:2010-11, EN ISO 13849-1:2015, EN ISO 13849-2:2012, EN ISO 13850:2015, EN 60204-1:2006-6, EN 1175-1:1998+A1:2010, 2006/42/EC).

Scaling software to different hardware configurations is challenging and requires significant effort. During the course of the VOJEXT project, the presented architecture, including the LOCEM, was deployed on seven different robots that used three different types of safety scanners. There were also two different types of robotic arms (UR5e and UR10e) in four different configurations: single robotic arm in the middle of mobile platform, single robotic arm in the middle of mobile platform on pedestal, single robotic arm in front of mobile platform, and two robotic arms in front of mobile platform. All robots were based on one mobile platform model with multiple hardware adjustments to specific use cases.

For all of the mentioned robots, the system architecture (see Section 2) with its PEMKRE, HICEM, and LOCEM modules operated in the same way. Moreover, this architecture can be transferred to other robot models with limited effort required, as long as their internal subcomponents provide functionalities similar to already implemented ones.

A good example of this could be the RB-VOGUI+ [20] robot that is designed for outdoor operation. It uses typical wheels with outdoor tires mounted on a suspension that can rotate each wheel separately. This makes RB-VOGUI+ mobile platform navigation control significantly distinct from the RB-KAIROS+. However, the same LOCEM interfaces could be run on the RB-VOGUI+ robot as in principle its controllers provide the same atomic actions.

7. Plasterboard Wall Preparation Case Study

This section describes the deployment of the implemented robot system in one of the use cases during the VOJEXT project, i.e., the preparation of plasterboard walls in the construction domain. This use case consists of covering joints between plasterboard walls with tape, spraying compound on walls, and sanding

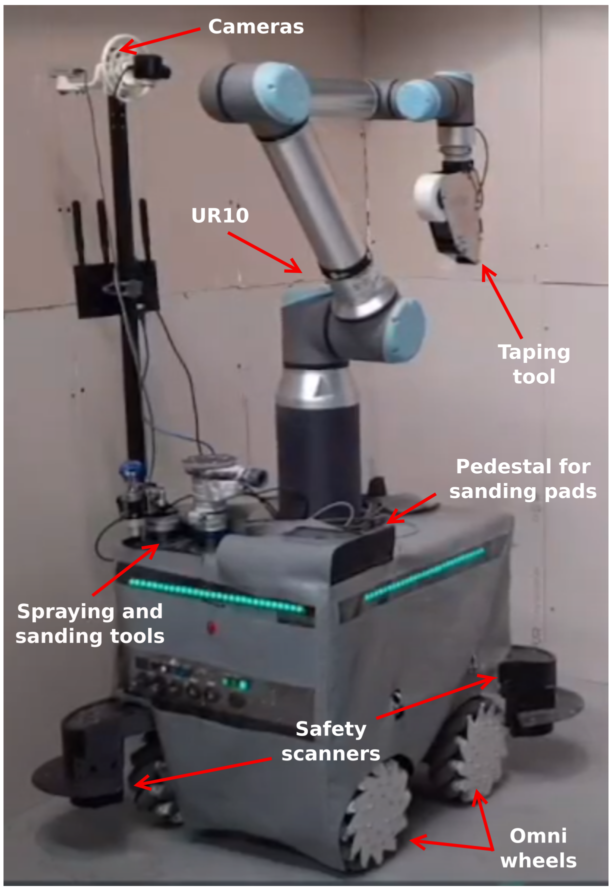

Figure 6. RB-KAIROS+ robot and its customized equipment for a plasterboard wall preparation use case during the VOJEXT project

the surface of the applied compound to smooth it after it has dried. All of these processes also include human-robot collaboration for changing the tools of the robotic arm.

The requirements of this wall preparation use case were defined by one of the VOJEXT project's industrial partner — Acciona [1]. Successful operation required the following process goals to be met:

1) Proper tool selection (based on process chosen by worker)
2) Tape application on plasterboard wall joints (required worker evaluation)
3) Spraying joints and tape with compound
4) Spraying wall with compound (required worker evaluation)
5) Smoothing of dried compound to achieve a uniform layer thickness

To achieve the given goals, the RB-KAIROS+ robot presented in the previous section was equipped with three different types of tools. Compound spraying was performed using a Graco 26C624 spraying gun [10]. Smoothing of applied material was performed using an OnRobot solution [13] that was available on the market. Taping of plasterboard wall joints is not covered well in research and market-ready solutions (FerRobotics [8], Straub Design Company [23], Robo-Tape [17]) in terms of robotic operation, as those tools

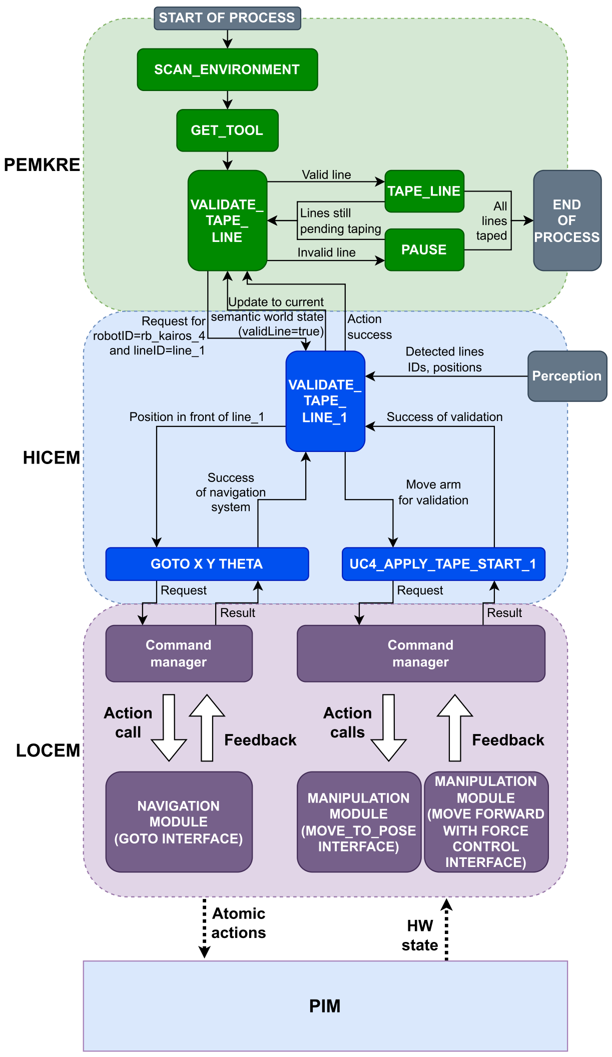
START OF PROCESS
SCAN_ENVIRONMENT
GET_TOOL
PEMKRE
VALIDATE_TAPE_LINE
Valid line
TAPE_LINE
Lines still pending taping
All lines taped
END OF PROCESS
Invalid line
PAUSE
Update to current semantic world state (validLine=true)
Request for robotID=rb_kairos_4 and lineID=line_1
Action success
Detected lines IDs, positions
Perception
VALIDATE_TAPE_LINE_1
Position in front of line_1
Success of validation
HICEM
Success of navigation system
Move arm for validation
GOTO X Y THETA
UC4_APPLY_TAPE_START_1
Request
Result
Command manager
Action call
Feedback
Action calls
LOCEM
NAVIGATION MODULE (GOTO INTERFACE)
MANIPULATION MODULE (MOVE_TO_POSE INTERFACE)
MANIPULATION MODULE (MOVE FORWARD WITH FORCE CONTROL INTERFACE)
Atomic actions
HW state
PIM

Figure 7. Simplified excerpts from the workflow and dataflow within and between the layers for the taping process. The spraying and sanding processes are performed analogously

support only tapes with a width smaller than 50 mm or were designed for thick, more stiff tape types than the one used by Acciona at their construction sites. Therefore, available robot-ready, tape-applying solutions were not designed to work in the conditions specified for this use case by Acciona. For these reasons, a custom taping solution was developed for this use case.

Additionally, a vertical pedestal was mounted on the mobile platform to increase the reach of the robotic arm to meet the requirements of this use case. The customized robot and its equipment can be seen in Fig. 6.

The main objective of the performed tests is the successful completion of the tasks. To achieve this goal, the system needs to be able to complete tasks in a mostly autonomous manner. The system was run in two different environments: at an Acciona workshop facility and in a Robotnik test area. This allowed to test the performance of the proposed solution with varying environmental factors.

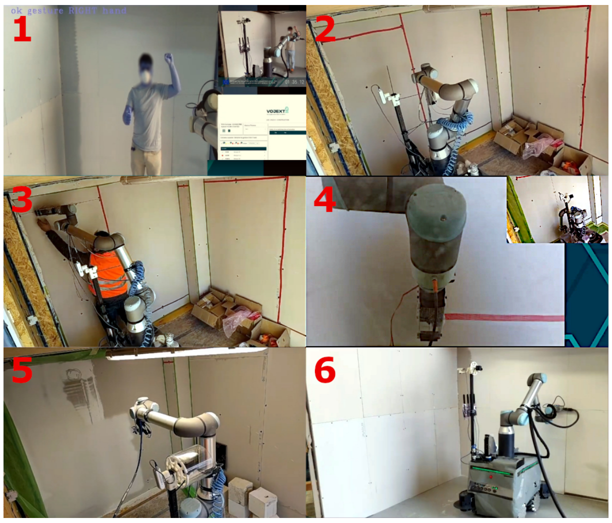

Figure 8. Chronological sequence of actions related to demonstration runs of the robot performing the plasterboard wall preparation processes

Each of the tasks was performed in the way presented in Fig. 7 and Fig. 8. The latter figure presents all operations that are part of the plasterboard wall preparation. At the beginning of its individual taping, spraying, and sanding processes, the robot scans (if currently needed) the environment for joint lines and areas on the surrounding walls and the proper tool is selected for the current process. Next, the corresponding tool is mounted by a worker on the robot's end-effector (subfigure 1). Then the mobile platform drives to a position where the taping can start (subfigure 2). The system is able to apply tape on wall joints automatically, but before each of the taping operations a worker may optionally evaluate whether the joint would be properly covered and whether the robot is well positioned to complete the task (subfigure 3). The robot covers the joint with tape (subfigure 4) and sprays the compound on the wall (subfigure 5). The last part of the use case is sanding the wall to make its surface even (subfigure 6).

Fig. 7 shows the line validation task within the taping process as an example. The PEMKRE module uses a process ontology that is similar to a flow chart and combines a set of required tasks to achieve the process. The PEMKRE sends a "VALIDATE_TAPE_LINE" high-level action with the ID of the next line (here "1") from its semantic world state, which is also modeled as an ontology. The behavior tree defined for this action type in the HICEM is executed, which calls low-level commands in the LOCEM, when robot operations are required. In the case presented in Fig. 7, those are the "GOTO" and "UC4_APPLY_TAPE_START" commands. The first command sends the mobile platform to coordinates in front of line "1". Those coordinates were derived based on a previous environmental scan and stored in the HICEM. The second command moves the robotic arm to a position for evaluation by the worker. In case of successful validation, the HICEM passes the positive result to the PEMKRE, which then returns the "TAPE_LINE" action to proceed with the actual taping.

In case of failure, the worker may perform the taping of this line manually and the PEMKRE returns the next "VALIDATE_TAPE_LINE" action with the ID of another line. The LOCEM interprets inputs from the HICEM and calls corresponding interfaces, i.e., the navigation interface for the "GOTO" command and the manipulation interfaces for the "UC4_APPLY_TAPE_START" command.

During the tests of the system, the use case partner Acciona decided that spraying compound across the whole wall surface provides sufficient quality to not repeat this task separately for both the joints and the whole wall surface. Using the designed system architecture, the robot managed to also smooth the wall surface to achieve the required thickness. For the tests performed at the Robotnik site, a video from an associated webinar is available online (https://youtu.be/PLMMWsvYgJc). The same hierarchical architecture was also successfully used in the other use cases defined in the VOJEXT project. Those use cases presented different challenges, such as compatibility with an already existing factory management system, integration of external devices like punching, injection, and molding machines, as well as worker state detection.

Successful implementation in a variety of different use cases confirms the correct operation of the control system architecture.

8. Conclusion

The proposed solution was successfully implemented and tested in a building construction use case, whose domain is particularly affected by the skilled labor shortage and lack of available automation solutions. The solution was also deployed in other use cases that were provided by other end users in the VOJEXT project and achieved successful operations in each one. The implementation of these use cases, which usually involved multiple external systems, advanced perception, and human interaction, was made feasible by the flexible abstraction layers of the robot system architecture. While it was not demonstrated, in principle, that the solution is scalable, it can be easily applied to a large number of different scenarios. The presented architecture is transferable and may be implemented for multiple mobile manipulator robots with a high level of hardware and software customization. This would require further research, work, and involvement of well-established robotic solution providers and integrators.

AUTHORS

Łukasz Granat* – Robotnik Automation S.L., Ronda Auguste y Louis Lumière 8, 46980 Parque Tecnológico Paterna, Valencia, Spain, e-mail: lgranat@robotnik.es.

Michał Bryła – Łukasiewicz Research Network – Industrial Research Institute for Automation and Measurements PIAP, Al. Jerozolimskie 202, 02-486 Warsaw, Poland, e-mail: michal.bryla@piap.lukasiewicz.gov.pl.

Kuba Kamiński – Łukasiewicz Research Network – Industrial Research Institute for Automation and Measurements PIAP, Al. Jerozolimskie 202, 02-486 Warsaw, Poland, e-mail: kuba.kaminski@piap.lukasiewicz.gov.pl.

Filip Jędrzejczyk – Łukasiewicz Research Network – Industrial Research Institute for Automation and Measurements PIAP, Al. Jerozolimskie 202, 02-486 Warsaw, Poland, e-mail: filip.jedrzejczyk@piap.lukasiewicz.gov.pl.

Sławomir Puchalski – Łukasiewicz Research Network – Industrial Research Institute for Automation and Measurements PIAP, Al. Jerozolimskie 202, 02-486 Warsaw, Poland, e-mail: slawomir.puchalski@piap.lukasiewicz.gov.pl.

Ingmar Kessler – fortiss, Research Institute of the Free State of Bavaria associated with Technical University of Munich, Guerickestraße 25, 80805 München, Germany, e-mail: ikessler@fortiss.org.

Alexander Perzylo – fortiss, Research Institute of the Free State of Bavaria associated with Technical University of Munich, Guerickestraße 25, 80805 München, Germany, e-mail: perzylo@fortiss.org.

Ángel Soriano – Robotnik Automation S.L., Ronda Auguste y Louis Lumière 8, 46980 Parque Tecnológico Paterna, Valencia, Spain, e-mail: asoriano@robotnik.es.

*Corresponding author

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program (H2020-DT-2019-2) under grant agreement 952197 (VOJEXT).

References

[1] Acciona. "ACCIONA | Business as unusual". https://www.acciona.com. Accessed 2024-07-23.

[2] A.F. Cutting-Decelle, R.I.M. Young, J.J. Michel, R. Grangel, J. Le Cardinal, and J.P. Bourey, "ISO 15531 MANDATE: A Product-process-resource based Approach for Managing Modularity in Production Management", *Concurrent Engineering*, vol. 15, no. 2, 2007, 217–235, 10.1177/1063293X07079329.

[3] ArtiMinds. "ArtiMinds RPS - Intuitive Robot Software for Advanced Robotics". https://www.artiminds.com/robotics-software-and-services/robot-programming-suite-basic. Accessed 2024-08-01.

[4] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues, "Knowledge for Intelligent Industrial Robots". In: *AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI*, vol. SS-12-02, 2012, 1–6, https://lup.lub.lu.se/record/4679237.

[5] R. Cyganiak, D. Wood, and M. Lanthaler. "RDF 1.1 Concepts and Abstract Syntax". W3C Recommendation, W3C, 2014. https://www.w3.org/TR/rdf11-concepts.

[6] Directorate General for Communication of the European Commission. "Tackling labour and skills shortages in the EU". https://commission.europa.eu/news/tackling-labour-and-skills-shortages-eu-2024-03-20_en. Accessed 2024-07-23.

[7] D. Faconti. "BehaviorTree.CPP 4.6. Framework to create BehaviorTrees". https://github.com/BehaviorTree/BehaviorTree.CPP/blob/2a8a226fbbd99f524f0796e0d8a3145773c61c06/README.md. Accessed 2024-07-23.

[8] FerRobotics Compliant Robot Technology GmbH. "Active Taping Kit ATK, robotic masking, tape applicator, end effector". https://www.ferrobotics.com/en/services/products/active-taping-kit-atk-robotic-masking-tape-applicator-end-effector. Accessed 2024-07-23.

[9] R. Ghzouli, T. Berger, E. B. Johnsen, S. Dragule, and A. Wąsowski, "Behavior trees in action: a study of robotics applications". In: *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering*, vol. 1, 2020, 196–209, 10.1145/3426425.3426942.

[10] Graco. *311053L, Manual, Automatic Airless Spray Guns, Instructions, English*, 2020. https://www.graco.com/content/dam/graco/tech_documents/manuals/311/311053/historic/311053EN-L.pdf.

[11] Intrinsic. "Intrinsic Flowstate". https://www.intrinsic.ai/flowstate. Accessed 2024-08-01.

[12] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of Behavior Trees in robotics and AI", *Robotics and Autonomous Systems*, vol. 154, 2022, 1–18, 10.1016/j.robot.2022.104096.

[13] I. Kessler and A. Perzylo, "Flexible Modeling and Execution of Semantic Manufacturing Processes for Robot Systems". In: *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2024, 1–8, 10.1109/ETFA61755.2024.10710791.

[14] OnRobot. "Sander Tool for Collaborative Robots | Sanding & Polishing | EOAT". https://onrobot.com/en/products/onrobot-sander. Accessed 2024-07-23.

[15] Open Robotics. "ROS - Robot Operating System". https://www.ros.org. Accessed 2024-07-23.

[16] A. Perzylo, I. Kessler, S. Profanter, and M. Rickert, "Toward a Knowledge-Based Data Backbone for Seamless Digital Engineering in Smart Factories". In: *2020 IEEE 25th International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, 164–171, 10.1109/ETFA46521.2020.9211943.

[17] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2016, 2293–2300, 10.1109/IROS.2016.7759358.

[18] RoboTape. "RoboTape: Tape Better, Faster, Smarter". https://robotape.com. Accessed 2024-07-23.

[19] Robotnik. "RB-KAIROS AMR with 250 Kg of payload and omnidirectional wheels for indoor applications". https://robotnik.eu/products/mobile-robots/rb-kairos-2/#rb-kairos. Accessed 2024-07-23.

[20] Robotnik. "RB-KAIROS+ UR10e Autonomous Mobile Manipulator for indoor applications with a 12,5 kg payload on its arm". https://robotnik.eu/products/mobile-robots/rb-kairos-2/#rb-kairos-plus-ur-ten. Accessed 2024-07-23.

[21] Robotnik. "RB-VOGUI+ Mobile Manipulator for R&D applications in outdoor environments". https://robotnik.eu/products/mobile-robots/rb-vogui-en/#rb-vogui-plus. Accessed 2024-07-23.

[22] J. Romeo. "Robotics for the Rest of Us: Automation in Small and Midsize Businesses". https://www.robotics247.com/article/robotics_for_the_rest_of_us_automation_in_small_and_midsize_businesses. Accessed 2024-07-23.

[23] Siemens. "Easy robot integration with SIMATIC". https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/highlights/robot-integration.html. Accessed 2024-08-01.

[24] Straub Design Company. "SD-610 Robot-Mounted Tape Applicator". https://straubdesign.com/new-products/sd-610-robot-mounted. Accessed 2024-07-23.

[25] W3C OWL Working Group. "OWL 2 Web Ontology Language Document Overview (Second Edition)". W3C Recommendation, W3C, December 2012. https://www.w3.org/TR/owl2-overview.

[26] W3C SPARQL Working Group. "SPARQL 1.1 Overview". W3C Recommendation, W3C, 2013. https://www.w3.org/TR/sparql11-overview.