Towards Robotic Laboratory Automation Plug & Play: Reference Architecture Model for Robot Integration

Ádám Wolf[a,b,*], Panna Zsoldos[c], Károly Széll[d], Péter Galambos[c]

[a] *Baxalta Innovations GmbH, a Takeda company, Industriestraße 67, A-1221 Wien, Austria*
[b] *Doctoral School of Applied Informatics and Applied Mathematics, Óbuda University*
[c] *Antal Bejczy Center for Intelligent Robotics, Óbuda University*
[d] *Alba Regia Technical Faculty, Óbuda University, H-8000 Székesfehérvár, Hungary*

Abstract

Supportive robotic solutions take over mundane, but essential tasks from human workforce in biomedical research and development laboratories. The newest technologies in collaborative and mobile robotics enable the utilization in the human-centered and low-structured environment. Their adaptability, however, is hindered by the additional complexity that they introduce. In our paper we aim to entangle the convoluted laboratory robot integration architectures. We begin by hierarchically decomposing the laboratory workflows, and mapping the activity representations to layers and components of the automation control architecture. We elaborate the framework in detail on the example of pick-and-place labware transportation - a crucial supportive step, which we identified as the number one area of interest in terms of laboratory robotics. Our concept proposal serves as a reference architecture model, the key principles of which were used in reference implementations, and are in line with international standardization efforts.

Keywords: Laboratory Automation, Mobile robotics, Autonomous manipulation, System integration, Plug and Play, Digital Twin

1. Introduction

The efficiency of pharmaceutical research and development (R&D) has been declining since 1950 [1]. This is due to an array of different factors, including the saturation of the market, regulators raising the requirements towards new drugs, and inefficient allocation of resources. To cope with the increase in complexity, laboratory automation solutions must provide more than just the capability of high throughput and brute-force experimentation. Advancements in experimental design approaches, modeling & simulation, and artificial intelligence

*Corresponding author

Email address: `adam.wolf@takeda.com` (Ádám Wolf)

(AI) based techniques are promising prospects in this regard. However, without the appropriate supportive (enabler) technologies of laboratory automation, such as interoperability solutions, robotics and human-in-the-loop approaches, the desired effect cannot be achieved. These new types of systems, however, present an increased complexity, compared to traditional laboratory automation platforms. To answer the challenge of integrating the increased number of components, we propose a reference architecture model (RAM), including a hierarchical decomposition of laboratory workflows, outlining the corresponding layers of the control architecture, and defining a taxonomy for lab robot activities.

The automation of life science laboratories is motivated by diverse factors, depending on the application. In high throughput (HTP) screening or quality control (QC) scenarios, for example, repetitive tasks must be performed in a structured environment with relatively low process variability. In contrast, a more dynamic R&D setup may pose a less structured and less constant environment. The former scenario can be addressed by traditional means of automation, such as customized and hard-programmed robotic cells and integrated automation lines [2, 3]. These solutions resemble special-purpose equipment in the automotive industry and in other production environments, where such custom-made cells serve a specific purpose, e.g., to perform a certain assembly sequence or end-of-line testing. These activities ensure a high throughput and high repeatability, but lack the flexibility that is necessary for dynamic workflows. For simple tasks, such as liquid handling and labware transportation, modular and easily re-configurable solutions already exist [4, 5, 6]. To perform more complex robotic activities flexibly, such as device- and tool interactions, novel robotics approaches must be adapted. These solutions include increasingly advanced perception, cognition, knowledge representation, and information-sharing capabilities.

Our Laboratory Automation Plug and Play (LAPP) concept addresses the challenges that arise in the context of robotizing flexible laboratory workflows [7]. As a blueprint, the framework aims to utilize already-established standardized solutions wherever possible and extends the stack by new elements where necessary. As such, we identified technologies to cover various needs of multi-layer lab automation systems. These range from SiLA [8] as a device communication protocol and ROS [9] as a robotics middleware, all the way to low-level robot-specific solutions, e.g., for navigation or motion planning. Combining these into a reference architecture model will ensure the highest compatibility between the various components of the system, thus achieving what we refer to as plug & play integration. LAPP primarily focuses on the robotics aspect of laboratory automation, more specifically on supportive robotics, such as pick-and-place transportation activities. This task is already addressed by multiple different solution providers, both in the form of bench-top and mobile robots. However, true plug & play compatibility of these systems is still not ensured. In [10] we proposed a digital twin (DT) approach as a means of a standardized information representation layer to facilitate the automatic interfacing of system components. According to our approach, each entity in the automated laboratory ecosystem has its own digital representation (digital twin or DT), which holds all of the relevant information about the entity that other elements of the system may need in order to interact with it. Distinguishing between *premises*, *devices*, *robots* and *labware* provides a framework where certain pieces of information can be allocated with respect to a certain entity. Certain properties are pre-defined by the vendor, whereas others are defined during commissioning, and yet another group of properties changes during operation. For the aforementioned pick-and-place transportation, most importantly, the geometrical relations, i.e., the positions of the components are represented.

This paper is the third piece in the series *Towards Laboratory Automation Plug & Play*, and it extends the LAPP framework with a reference architecture model (RAM). As such, we map the levels of the process and the levels of the automation system to each other. This way we provide a framework, which spans a structure for comprehending the complexity of these elaborate systems. LAPP in its entirety, and the RAM being a key part of it, aims to serve as a canonical framework. To achieve this, the two key aspects are providing an over-arching descriptive overview of the state-of-the-art and outlining a prescriptive blueprint for implementations. The goal is to simplify the integration of these complex systems, whose parts are usually contributed by multiple different solution providers.

We define LAPP-RAM with lab robotics in the focus, but covering the whole vertical of laboratory automation systems. We consider a broad variety of devices, such as storage, processing, and analytical devices, as well as liquid handlers. On the robotics side, we define complex activities that go beyond simple pick-and-place tasks and cover interactions with the environment, including laboratory devices, tools, and labware.

In Section 2 we provide an overview of state-of-the-art solutions for hierarchical decomposition in different domains, and then, inspired by these, we outline our reference architecture model in Section 3. Following that, in Section 4, we present our robotic activity representation framework. To assess the use cases for supportive robotics in life science laboratories, we summarize the results of a non-representative survey in Section 5. This shows, that most experts consider the pick-and-place transportation of labware as the most important robotic activity in the lab. We elaborate this activity based on the proposed framework. For additional laboratory robot capabilities, we propose a taxonomy. Finally, we provide an outlook on the implementation of the framework in Section 6 and conclude the paper in Section 7.

2. State of the art

In this section, we provide an overview of the present landscape in terms of system architectures in laboratory automation. We consider different approaches for hierarchical workflow representation, hierarchical and distributed system architectures, and representations for robotic activities.

2.1. Hierarchical workflow representation

As we discussed in [7], the representation of laboratory workflows is a very important aspect of automation. Ultimately, a process can only be automated when it is adequately described in a machine-readable form. For that, not only the sequence of actions but also their input parameters, conditional relations, and the control flow must be prescribed. This subsection provides an overview of the relevant state-of-the-art solutions that address this problem.

Hierarchical decomposition is an effective way of comprehending the vertical of complex multi-level systems in different domains. First, we consider two examples from project management.

Work Breakdown Structure (WBS) [11, 12] is a project management tool that provides a deliverable-oriented breakdown of processes. This is to aid scoping, cost estimation, and work-package assignment. As we define LAPP-RAM in Section 3, we adopt certain principles of WBS in the following fashion:

Similarities:

- Both structures serve the purpose of hierarchical decomposition of processes.
- Both definitions use mutually exclusive and collectively exhaustive (MECE) elements, in that they cover the corresponding activities unambiguously and explicitly.
- Their elements are outcome-oriented and agnostic toward implementation.
- WBS templates can be used to represent laboratory workflows with the LAPP-RAM notation.

Differences:

- A WBS breakdown usually has two to four levels, where the terminal element (the one that is not subdivided further) is a work package that a project member can meaningfully address.
- In LAPP-RAM, the depth, i.e., the level of the terminal element, cannot be strictly defined. Each level can be considered to be the deepest necessary abstraction from the perspective of a certain control element. e.g., from the perspective of the lab automation scheduler low-level activities, such as robotic motion and actuator primitives, are not relevant.
- WBS does not provide activity definitions in either a symbolic/abstract or executable/geometric fashion.
- In the context of LAPP-RAM, we define Robotic Activity Representations (RARs). Their primary purpose is to describe robotic activities for execution at the appropriate control level within the laboratory automation system.

Table 1: Hierarchical decomposition frameworks in project management. Work Breakdown Structure (WBS) [11, 12] and issue trackers (such as Jira) [13, 14]

Level	**WBS**	**Issue trackers**
7		Initiative
6	Level 1	Epic
5	Level 2	Task/story
4	Level 3	Subtask
3		
2		
1		

In the realm of hierarchical decomposition, a prime example can be found in issue trackers such as Jira [13, 14]. There the following levels are distinguished:

1. **Tasks**: At its core are *Tasks*. These denote individual work units that can be directly assigned to a designated person. They can exhibit various interrelationships, from dependencies to specific completion sequences.
2. **Subtasks**: A *Task* can be subdivided into *Subtasks*, which further detail the work.
3. **Stories**: On a broader scale, *Tasks* can be aggregated into *Stories*. A *Story* acts as an overarching task, encapsulating general information about a collection of *Tasks*.
4. **Epics**: Ascending further, we find *Epics*, which encapsulate the overarching objectives or themes of a project. It's common for *Stories* and *Tasks* within a single *Epic* to have interwoven dependencies.
5. **Initiatives**: At the zenith is *Initiatives*. These symbolize wider goals and might encompass multiple *Epics* from different teams, signifying a more extensive, collective aim.

Table 1. puts the examples from the project management application area next to each other in terms of the levels of the breakdown. We use a uniform level numbering across the paper, including the tables 1, 2, 3 and in Section 4, where we outline the proposed LAPP-RAM hierarchical decomposition. This represents a loose mapping between the hierarchical layers of the different frameworks.

Biological laboratory protocols possess distinct requirements that general-purpose solutions may not adequately address. The Laboratory Open Protocol (LabOP) has been introduced as a specialized representation for biological protocols [15].

The objectives of the LabOP framework are multifaceted:

- **Inter-project Communication**: LabOP facilitates seamless transfer and comprehension of protocol representations across disparate projects.
- **Reproducibility**: The framework is tailored to ensure that different laboratories, even those with varied equipment configurations, can reproduce these representations with high fidelity.
- **Clarity and Reusability**: LabOP's representations are both unambiguous and abstract, striking a balance to promote reusability across diverse biological contexts.
- **Broad Accessibility**: The framework is designed such that both automated systems and researchers can readily interpret and utilize the representations.

To ensure reliability and adaptability, LabOP leverages established technologies including the Unified Modeling Language (UML), Autoprotocol, and the Synthetic Biology Open Language (SBOL). Moreover, LabOP's remit extends beyond mere protocol representation; it encompasses execution records, resultant data, and the requirements of the execution framework itself. Notably, UML offers an agnostic approach to semantically model behavior, ensuring compatibility and coherence in diverse scientific domains.

LabOP is responsible for representing the biological workflows, where the implementation of a laboratory primitive is considered as a device-dependent black box. Lab robots are used in most cases, for supportive actions (e.g., labware transportation) which are not represented in the biological protocols. As such, neither does the Autoprotocol library include a specific primitive for labware transportation, nor does LabOP's extension. Besides supportive actions, lab robots would theoretically be suitable to fulfill certain meaningful activities, such as shaking the sample. We address both types of activities in Section 5.

When it comes to robotic activities, it is important to delineate the different levels of representation. Chu [16] and D. Nagy et al. [17] both provide hierarchical decompositions in this regard, while Vedula et. al [18] analyze the structure of surgical activities from a human surgeon's perspective. Although the field of application and the nomenclature are different, the corresponding levels can be identified, as presented in Table 2.

All of the frameworks and structures discussed in this subsection serve the purpose of the hierarchical decomposition of complex activities and/or the systems that execute them. In general, the upper layers influence the workings of the lower ones by means of control and the assignment of activities, while information flows from the bottom layers towards the upper ones.

It can be seen that hierarchical decomposition is often used in project management to plan the scope, resources, scheduling, and timelines. As seen in [7] and [10], the planning aspect can be bridged towards execution. This can be observed in the case of a Business Process Model and Notation (BPMN) process [20], which can be directly run on a process engine. When it comes to

Table 2: Hierarchical decomposition frameworks in surgical operations, robot surgery, service robotics and laboratory robotics.

Lvl	Vedula [18]	D. Nagy [17]	Leidner [19]	Chu [16]
7				
6	Procedure	Operation		
5	Task	Task	Action template	Task motion
4	Maneuver	Subtask		
3	Gesture	Surgeme	Operations	Motion element
2		Motion prmtv.	Geometric level	Motion step
1				

executing the process on an automation system, the levels of the process representation are often reflected in the levels of the control system. The levels of complex automation architectures are primarily delimited based on physical and technological considerations. We discuss this aspect in 2.2.

2.2. Strict multi-level structure vs. service-oriented architecture

Multi-level architectures are well-established and widely applied in conventional industrial automation scenarios. The ISA-95 standard [21] provides a hierarchical model and terminology for the integration of Enterprise Resource Planning (ERP), Manufacturing Execution Systems (MES), and Supervisory Control and Data Acquisition (SCADA) systems. On the bottom, the physical process is considered, which produces the desired outcome (product). This process is bi-directionally interfaced with the *Field* level, which is the level of sensors and actuators. On top of this, there is usually a *Direct control* layer containing the input-output (IO) modules and low-level logic implemented on a programmable logic controller (PLC) or microcontroller. The *Process supervisory* layer aggregates the information coming from the *Direct control* nodes and provides the operator with a human-machine interface (HMI). One step higher resides the *Process supervisory* layer, which is responsible for the monitoring and control of the production process. The *Production Supervision* (also called Manufacturing Operations Management) layer controls the workflow or recipe, maintains records, and optimizes the production process. The highest level in this distinction is that of the *Plant Management and Scheduling*, which establishes the plant schedule and controls factors, such as material use, delivery, and shipping and inventory levels. We drew an analogy between such industrial automation systems and laboratory automation architectures in [22].

In the context of the Industry 4.0 approach, the need for increased flexibility of the processes and for individualized production created the need for a new approach [23]. The reasoning is twofold. For information to reach the higher levels, it must propagate through multiple intermediary layers, often via multiple different protocols. Moreover, to adapt the system to a new process, the low-level control units (in most cases PLCs) must be manually reprogrammed, taking into consideration their interlinked co-dependencies. In contrast, Schnicke et al. [24] adopt a peer-to-peer service-oriented architecture (SOA), according to the core principles of Industry 4.0. Stateless services are implemented, which can be called with the appropriate parameters by another component or an orchestrator. For that, information must be represented in an appropriate DT framework. In this framework, the elemental component is the service provider, i.e., a functional unit, such as a device or machine. For each service provider, the offered capabilities are described in the form of a *ServiceTag*. The services can be organized into sequences, i.e., recipes that prescribe the steps for manufacturing a product. This makes it possible, that - in case of a change in the process - only the high-level process description must be adapted, and the low-level implementation steps are reconfigured by organizing them into a new order and/or changing their parameters.

Despite focusing on overcoming the strict multi-level (or automation pyramid) structure, Schnicke et al. consider multiple different ways of introducing hierarchical relations. First, they distinguish between *transforming services* and *supportive services*. The former means actions that actually change the product in some way, while the latter means non-transforming actions, such as transportation from one station to another. A high-level (abstract) recipe only prescribes the transforming actions that the product must go through, all in a device-agnostic fashion and without supportive actions. It is the orchestrator's role to turn this into an executable plan by taking into account the plant topology and the service providers' capabilities. This also means that the supportive services are filled in wherever needed, e.g., for a transfer between two stations. The second example of introducing hierarchical relations is the statement that services can be nested into each other, and designated (so-called *group control components*) can interface with other control components. By this, a multi-level structure can be implemented both in the process representation aspect and also by the means of control. However, this optionally-introduced hierarchical structure is not enforced to follow the strict levels of conventional automation pyramids. Thirdly, the possibility of exposing service details on multiple levels is mentioned. The given example is a transport service separated into *prepare*, *perform* and *reset* steps. Table 3 maps Schnicke's framework to a hierarchical decomposition.

A similar consideration was also one of the key principles for designing the SiLA 2 protocol [25]. SiLA enables service-oriented, peer-to-peer communication, which aligns with the principles of Industry 4.0. In this framework, service descriptions are called *feature definitions*. Although it is beyond the scope of the present article to discuss the SiLA 2 protocol comprehensively, the protocol will be a key building block in implementing LAPP. The feature definition framework and its design principles are also relevant on a conceptual level, thus, certain aspects will be discussed.

To facilitate the cooperation and collaboration of technical objects by means of virtual representation and connectivity, the Reference Architecture Model Industrie 4.0 (RAMI4.0) [26] was created by the Plattform Industry 4.0 network in Germany. For this purpose, a DT approach is described where the physical world is represented in the information world. The central notion is the

Table 3: Hierarchical levels in Schnicke's framework, mapped against the levels of LAPP (defined in Section 4)

Level	**Process**	**Protocol**	**Control architecture**
8			
7	Order		
6	Product recipes	BPMN	Orchestrator & process engine
5	Service	AAS	Control component (PLC)
4	Steps		
3			
2			
1			
0			

asset, which is broadly defined as an object which has value for an organization. It can be a physical entity (product, machine, machine element, etc.) or a virtual/logical entity (software, idea, archive, etc.).

The RAMI4.0 defines a structure to describe the main elements of an asset along three axes:

- Arthitecture (Layers): Information relevant to the role of the asset. Can be interpreted as elements of the DT.
- Life cycle & value stream: state at a particular time and location
- Hierarchy levels: Extension of the automation pyramid approach

The RAMI4.0 framework served as primary inspiration while outlining the LAPP-RAM.

2.3. Robotic activity representation

Out of the layers of the workflow decomposition (and of the system architecture) the closest to the physical process is the low-level side. As such, various physical activities can be implemented through robotization. To do this, the decomposition has to consider the specifics of the robot hardware, corresponding to a suitable level of granularity. In this context, different representation formats (aka protocols or languages) are used.

In its simplest form, a robot program is just a sequence of pre-taught movements and I/O operations, potentially with some basic logic operations, branching, and loops. Table 4 shows the various types of conventional robot movements (using Universal Robot's nomenclature [27]).

In more complex robot applications, simple movements might need to be grouped and organized for modular re-usability. This enables a sequence of movements to be performed throughout the robot program multiple times, potentially on multiple different frames of reference.

Besides industrial robotics, an increasing number of new application fields are arising. One of these is robotic surgery, which represents a significantly

Table 4: Types of conventional robot movements

Name	**Meaning**	**Characteristics**
MoveJ	Joint move	Optimal joint-based movement with non-prescribed path
MoveL	Linear move	Prescribed linear path
MoveC	Circular move	Prescribed circular path
MoveP	Process move	Defined path and velocity (e.g., for welding or dispensing)

different set of requirements but utilizes many advantages of the fundamentals. As such, the robustness and reliability of robots is a key factor when it comes to teleoperated or (semi-)autonomous robotic surgery. The latter application means that during a robot-actuated surgery, i.e., when the surgeon controls the robot's movements, the robot could take over some simple tasks to relieve the surgeon. To achieve this, the tasks must be defined in a fully descriptive manner. As the basis of these representations, analyzing how a human surgeon performs the task can be considered [18]. Nagy et al. propose a framework for surgical subtask automation based on the Da Vinci surgical robot, for which they provide a hierarchical structure to handle the different granularity levels of surgical motions [17].

Laboratory automation represents special needs towards robotic solutions concerning the complexity and variability of the tasks [28]. Chu proposes the concept of Motion Elements (ME) specifically for the automation of a complex sample preparation laboratory workflow [16]. They consider robot movements from one position to another as a *motion step* and group them in reusable units called *motion elements*.

The AI.Laboratory solution paradigm of Knobbe et al. represents a framework for robot-centered laboratory automation systems [29]. It incorporates the so-called Lab Skill System (LSS), a taxonomical collection of laboratory skills consisting of the basic and the cell-specific subgroups.

Laboratory robots' capabilities are currently mostly limited to pick-and-place type transportation of labware by means of a parallel gripper [7]. For the robot to perform these actions autonomously, a suitable knowledge representation is fairly simple. It must include the position of the object with relation to the robot, the object's gripping frame, and the required gripper width. However, to broaden the scope of laboratory robot capabilities, new advancements in robotics technologies must be utilized. To address the need for robots that are capable of working with a variety of objects in an unstructured human environment approaches from service robotics must be adapted.

As such, a very important aspect is how robots deal with complex objects, including tools, i.e., objects that are used in combination with the robot's own end effector to perform certain actions on other objects. This is how humans perform actions by hand with the use of hand-held tools. For that, humans use the dexterity of their end effector: a human hand with five fingers is incredibly flexible. Also, the cognition aspect enables humans to perceive objects with a focus on their purpose. The corresponding capabilities and constraints are assigned to the objects intuitively. For robots to be able to achieve similar flexibility in performing actions with tools, suitable knowledge representations and the corresponding planning framework are needed, which utilizes these representations. Leidner addresses these needs in his early works and also in his dissertation [19, 30].

3. The LAPP Reference Architecture Model

Drawing inspiration from the concepts elucidated in Section 2, we present the LAPP Reference Architecture Model (LAPP-RAM). LAPP-RAM is specifically crafted for the realm of laboratory automation, with the overarching goal of serving as a comprehensive and exhaustive reference architecture, akin to the RAMI 4.0 model, encompassing both vertical and horizontal integration structures.

3.1. Hierarchical decomposition of laboratory automation systems and workflows

With the help of a multi-dimensional representation, we characterize the elements of the laboratory automation system along three aspects. For all of these aspects, we span a scale of abstraction and granularity.

1. **Process decomposition**: Layers of the workflow representation, with the aim of manual or automated execution
2. **Protocols**: Activity representation formats or languages
3. **Technical layers**: Layers of the control architecture

We define these dimensions by using the same level numbering as in tables 7, 8, 9, 10 and 11. Levels 1-5 are also color-coded.

3.1.1. Process decomposition

Activities within a laboratory workflow can be hierarchically decomposed by outlining the distinct levels of granularity. For this purpose, we utilize the principles of WBS, as discussed in 2.1. In the LAPP-RAM each layer represents a level of abstraction, assuming that from that certain perspective, it is not important how the activity is implemented on the lower level. As such, each representation layer is agnostic towards the lower (inferior) ones. This means that from the perspective of the scientific *Service* description it does not matter how the experiment itself is executed. Similarly, from the *Experiment* description, the device-level execution is abstracted away. Also, arriving at the robot-specific levels it can be deduced that a high-level *Task* description does not concern how the task is executed, instead, only the outcome is interesting. With these principles in mind we distinguish between the following levels of the process decomposition.

7) Service refers to the entirety of the laboratory's capabilities, e.g., high throughput and/or microscale services

6) Procedure can be an experiment, an assay, or a culture maintenance program

5) Task is an elemental action item, which is performed by a human or a certain device

4) Subtask is an intermediary layer that represent parts of a task, that accomplish minor landmarks

3) Motion sequence is defined for the specific case of robotics, whereby the robot performs a sequence of motions, e.g., in order to approach a handover site

2) Motion primitive is an elemental motion of a robot or other mechanism

1) Actuator primitive is an output excerpted by a certain actuator, e.g., robot joint, pump, etc.

0) (Physical process)

We will present detailed examples for liquid handling in Section 3.2, and for robotics in Section 4.

3.1.2. Protocols

Inspired by [31] we outline the layers of the workflow representation languages, which aim for manual or automated execution.

7) Service protocol (SP) defines the services that a certain (cloud) lab offers

6) Experiment design language (EDL) serves as an outcome-oriented recipe on the workflow- or assay-level. It enables transferring the protocol between different laboratories. Answers the question *"What to do?"*

5-4) Laboratory process language (LPL) (the *how*) serves as an execution plan, which is grounded to specific devices of a specific laboratory platform. It includes a sequence (or program) of tasks. Answers the question *"With what parameters on which devices?"*

3) Modular robot program (MRP), as a robot-specific layer, implements tasks and subtasks by serving as a wrapper around the low-level robot functionalities. It incorporates reusable and parameterizable motion sequences.

2) Low-level robot program (LRP) encodes primitive motions and simple IO logic.

1) Joint trajectories and IO (JTIO) refer to the hardware-oriented definition of actuator control, e.g., direct joint motions.

3.1.3. Technical layers

Besides the dimension of protocols and languages, we also discuss the levels of the control architecture, analogous to the industrial control systems described in Section 2.2. On the *Service* level the activities are orchestrated by the laboratory management layer, which includes LIMS and LES systems. This is analogous to MES in industrial automation. An experiment or assay is executed by the *Laboratory automation scheduler* layer, which interacts with the *Device-level* components through a specific high-level connectivity protocol. Normally, lower-level operations of the system are not exposed by means of the lab automation connectivity protocol, but in the case of certain implementations, it might be desired to open up the possibility for custom low-level control and/or debugging. As such, the *Subtask* and deeper levels are generally considered to be covered by the device controller computer, whereas the robot controller is responsible for the *Motion sequences* and *Motion primitives*.

Using our uniform numbering, we define the following technical layers:

7) Lab management (LIMS, ELN)

6) Automation scheduler (LES)

3)-5) Device control PC (DPC)

1)-2) Embedded controller (EC)

Table 5 maps the three hierarchical aspects of LAPP-RAM against each other. It is important to mention that the borders between the levels are not strict, and some implementations may fuse or bridge certain levels entirely. There are strong correlations between the three dimensions, meaning, for example, that a certain level of process activity is usually represented by a certain type of protocol and is executed on a certain level of the automation system. However, these relations are not strict either. Depending on various factors, such as the size of the laboratory and the existing boundary conditions of the infrastructure, implementations may vary in each aspect.

Table 5: The three hierarchical aspects of LAPP-RAM, mapped against each other

Level	**Process**	**Protocol**	**Technical**
8			
7	Service	Service pr.	LIMS, ELN
6	Procedure	EDL	LES
5	**Task**	LPL	DPC
4	**Subtask**		
3	**Motion sq.**	MRP	
2	**Motion sq.**	LRP	EC
1	**Actuator pr.**	JTIO	
0			

3.2. Adaptation to liquid handling activities

To understand the hierarchical layers from a general (non-robot specific) activity's perspective, we consider the example of a liquid handling task. As a crucial part of most assays, liquid handling means the transfer of certain volumes of liquids from a source container to a destination container. Also referred to as pipetting, liquid handling tasks can be carried out either manually by a human or automatically by a pipetting robot.

A more traditional and more high-throughput-focused form of automated pipetting is the use of gantry-type liquid handlers, such as Tecan, Hamilton, or Beckman Coulter machines. These units can be considered *Device* level entities that fulfill a certain *Task* as a part of an *Experiment.* A pipetting robot executes a so-called liquid handling script, which contains *Subtask* level commands, such as aspiration and dispensing, and in some cases also plate movements within the worktable of the machine. Also, additional devices might be integrated into the liquid handler robot. In this case, they play a direct role in the script by performing certain subtasks either on the sample itself, such as shaking or incubation, or on the labware, such as washing. In this manner, the context of a liquid handler, containing the integrated devices, can be considered as a robot cell, i.e., a stand-alone unit. These, by themselves, can also be integrated into a bigger system (analogous to a production line), by interfacing with an over-arching controller (or scheduler) and also being physically connected by the means of material transport. The same principle is repeated within their own context, where they act as an integration platform for their sub-modules and integrated devices. They implement both control (and information), and material flow. This means that the *Scheduler-Device* relation can manifest recursively to some extent, i.e., a *Subtask* can ascend to become a *Task* having its own *Subtask*s. This shows that the boundaries of the hierarchical decomposition are fluid.

The *Subtask* level is defined as an activity segment that accomplishes minor landmarks. As such, it can be shown, why an Aspirate activity of a liquid handler is analogous to a GetLabware activity of a robot arm. In fact, a liquid handler robot by itself may perform pick-and-place tasks in the context of its own worktable, with the help of its robotic manipulator. In general, a liquid handler, in essence being a robot, performs *motion sequence*s, such as approach sequences or gripping. Going one level deeper, some liquid handlers are able to define so-called *motion vectors* or points, which are analogous to *Motion primitives.* On the deepest level, the axes of the gantry mechanics are controlled separately by the embedded controller in order to perform the desired motions. This is analogous to the joint trajectory control of robot arms. To stay with liquid handling, the control of the pumps can also be assigned to this level.

Table 6 maps liquid handling activities against the levels of the hierarchical workflow decomposition, along with the supportive robotic activity of labware transfer, which we elaborate in Section 4.

4. Robotic Activity Representations in LAPP

4.1. Design principles

As already stated in Section 1, the LAPP framework's primary focus is on supportive laboratory robotics. We outline a taxonomy, which provides a canonical system for naming and organizing present and future lab robot capabilities. These representations must be robot-agnostic in the sense that multiple different robots (from different vendors) must be able to implement them. However, the definitions of these capabilities must include appropriate constraints for the planner (or scheduler) to decide which robot can fulfill a certain task. This is a similar consideration to the bounding between the semantic and the geometric description in Leidner's Action templates (see Section 2.1). Moreover, the capability representations should align with the LAPP-DT concept, which necessitates that each robotic activity be depicted within the context of the corresponding laboratory asset, maintaining a relational perspective. This approach, akin to Leidner's object-oriented methodology, ensures comprehensive coverage. The representations must be agnostic towards the lower-level implementation by providing an appropriate level of abstraction and the use of symbolic definitions. These parameterizable representations will be used as commands during the execution of the workflow, each activity of the respective level being triggered by the super-ordinate control component.

4.2. Introduction to the taxonomy

We elaborate the taxonomy on the conceptual level, agnostic to the implementation. The framework should fulfill the following requirements:

- Form a semantic description for multi-level activities of laboratory automation
- Focus on laboratory robotics, including supportive activities, such as labware transport, and direct robotic implementations of primary activities, such as shaking
- Categorize these activities, based on their outcome and level of granularity
- Nest low-level activities into high-level ones

We will consequently use the definitions of the LAPP-DT [10], including the hierarchical definition of the coordinate frames (positions), unless otherwise stated.

5. Lab robot use cases

Robots are used for a wide range of tasks in life science laboratories [28, 32, 33]. We elaborate the LAPP robotic activity representations (RAP) with a focus on device-external robot arms - stationary (including rail-mounted) and mobile. The application field for these robots spans from labware transportation

[6] to advanced manipulation and even liquid handling [34]. The most ubiquitous of these are pick-and-place type labware transportation solutions, which are available off-the-shelf as market-ready products. Similar to the LAPP-DT concept [10], we also elaborate the LAPP RAP taxonomy by targeting this as a core capability. Additional robot capabilities are still less established, and most of the solutions are still in the research phase. Thus, the variability in the technical approaches is high. We also provide a structure to categorize these to-be-established capabilities and incorporate them in the taxonomy.

5.1. Survey

In June 2022, we conducted a targeted survey with the participation of seventeen experts in the field of laboratory automation. The majority of the participants identified themselves as *Users, laboratory experts* (seven), and *Robotics researchers* (six). In addition, the following groups were represented by one participant each: *Robot vendors*, *Lab software vendor*, *System integrator*, *Lab equipment vendor* and *Application engineer.* Twelve participants were active in the *Industry*, while five were in *Academia.* Fourteen were in *Research and development*, two in *Manufacturing*, and one in *Quality control.* The *Biologics* (six) and *Chemistry* (five) fields were represented by the majority, but *Software engineering*, *Bioinformatics*, *Biochemistry*, *Robotics*, *Mechanical engineering* and *Lab automation* were also marked. See the detailed analytics in the supplementary materials.

The participants were asked, for which use cases they already use or provide robots. *Gantry-type liquid handling* and *labware transportation, benchtop robots* (eleven each) were the top use cases, while *labware transportation, floor-level mobile manipulators* (six) and *Liquid handling, human-like* (four) also proved to be widely used. *Collaborative assistance, benchtop* (two), *Collaborative assistance, mobile*, *labware transportation, other (e.g., drones)* and *State monitoring, mobile robot* were also marked.

Following that, use cases were rated in regard to how likely the participants were to consider using or launching them in the future, based on their current state and future perspective. *Labware transportation; floor-level mobile manipulators* proved to be the most popular (4,1 rating on average), then *labware transportation; benchtop robots (e.g., PreciseFlex)* and *Collaborative assistance; benchtop* followed (both with rating 4,0).

In the final section of the survey, participants were asked to rate and suggest laboratory robot activities in the form of commands. The supplementary table `Commands_survey.xlsx` lists the commands in the order of decreasing average rating. It can be seen, that the experts rated the commands the highest that belong under the *LabwareTransfer* task. This justifies the decision that LAPP is elaborated primarily to address this use case. The next subsection presents how the pick-and-place type labware transfer task is broken down and represented as LAPP-RARs. The commands (RARs) marked with *Future capabilities* exceed the state-of-the-art capabilities of standardized lab robots. We do not elaborate on them in the present paper, but we cite relevant research and development endeavors, where applicable. During processing the results, we named each

Table 6: Breakdown for liquid handling, robot arm, mobile robot, and conveyor activities.

Lvl.	**Liquid handler**	**Robot arm**	**Mobile robot**	**Conveyor**
7	microscale services			
6	microscale chromatography workflow			
5	liquid transfer	labware transfer		
4	aspirate	get labware, put labware		
3	approach site	move to site	drive to station	-
2	motion vectors	move to arm waypoint	navigate to intermediary	move tray to desired position
1	pump control	joint control	base velocity commands	motor or magnet control

command by a *Command Identifier* and specified the corresponding hierarchical level.

5.2. Pick and place activities as LAPP RARs

In this section we elaborate on a comprehensive representaion of the simple pick & place labware transportation activities, using the LAPP RARs and LAPP DTs. In our previous paper [10] we described how a transportation task is canonically implemented as a sequence of robotic steps. In the code snippet 1 and in Table 2, we map this against the hierarchical levels presented in 3. The indentation represents the decomposition levels defined in 3. The *Service* and *Procedure* levels are omitted, as a LabwareTransfer *Task* is non-specific to *Services* and *Procedures*, instead, it is a reusable unit of high-level lab robot activity. Tables 7, 8, 9, 10 and 11 list the Robotic Activity Representations (RARs) that constitute the pick-and-place labware transfer task. Parameters of each command are set in *italic*. This list is exhaustive towards the pick & place labware transportation task.

The pseudocode snippet 1 and the Table 2 present the breakdown of a labware transfer task, as performed by a mobile manipulator. The implementation regarding a stationary robot can be achieved by omitting the activities related to autonomous mobile robots (AMRs) (marked with *), such as navigation. It can be seen, that the activities of the robot arm and those of the AMR are analogous.

In table 6 we present an exemplary breakdown of a laboratory service, according to the hierarchical levels, putting liquid handling, robot arm, mobile robot, and conveyor activities besides each other. It can be seen that a labware transfer task (and its corresponding subtasks) can be implemented by different means.

Throughout the following tables, we discuss *Task* (Table 7), *Subtask*(Table 8), *Motion sequence* (Table 9), *Motion primitive* (Table 10) and *Actuator primitive* (Table 11) level RARs in more detail.

Table 7: Task-level Robotic Activity Representations for pick-and-place labware transfer

Command ID	**Description**
LabwareTransfer	Robot moves an object (of a certain *labware-Type* e.g., `DEEP96`) from a predefined *source* site (a.k.a. nest) to a desired *destination* site. Depending on the distance between the two sites, this can be completed by a stationary, rail-mounted, or mobile manipulator.
TeachBaseWaypoint	Store the current map position as an *armWaypoint* for the AMR base.
TeachArmWaypoint	Store the current arm position as an *armWaypoint.* In the case of stationary robots, this is expressed with relation to the world coordinates, whereas for mobile manipulators fiducial markers are used as a device-attached reference.
TeachLabware	Store current finger position (width) or current finger force under a specific *labwareType*
MaintainBattery	Maintain the battery charge level above a set *minPercentage* by sending the robot to Charge, if the battery charge level drops below the threshold.

The *Task* and *Subtask* levels of the breakdown show a significant overlap with the SiLA base features *LabwareTransferManipulatorController* and *Labware-TransferSiteController* [35]. These definitions were developed as an endeavor of the SiLA Robotics Working Group, with the lead and active participation of the authors. The features are device agnostic, which means that they cover an array of different means of active labware transfer. In most cases, the active party in a labware transfer is a stationary robot arm, but conveyor belts and built-in manipulator arms are also covered.

The decomposition and sequence that we present in this paper adapt the SiLA features to mobile manipulator robots. See section 6 for more details on how the various aspects of the LAPP framework are incorporated in real-life implementations and are manifested in standardization endeavors, such as the activities of the SiLA Robotics Working Group.

We do not discuss representations lower than actuator primitive (such as position and force control activities), because they are not relevant from the perspective of the laboratory automation workflow. Instead, they are implemented on the low-level (embedded) robot control components.

In Figure 1, we visualize the decomposition of the labware transfer task into subtasks, and further down into low-level RARs. The breakdown considers a mobile manipulator robot, consisting of an AMR base, a robot arm, and a gripper.

Table 8: Subtask-level Robotic Activity Representations for pick-and-place labware transfer

Command ID	**Description**
PrepareForInput	Prepare for picking.
GetLabware	Pick up a piece of labware (of a specific *labware-Type* from a designated *source* site.
InternalPlace	Place labware on the robot's built-in storage (hotel).
PrepareForOutput	Prepare for placing.
InternalPick	Pick labware from the robot's built-in storage (hotel).
PutLabware	Place a piece of labware (of a specific *labware-Type* to a designated *destination* site.
Charge	Send the robot to its charging station, make it to dock and start charging.

Table 9: Motion sequence-level Robotic Activity Representations for pick-and-place labware transfer

Command ID	**Description**
MoveThroughSequence	Robot arm performs the complete approach motion sequence (by visiting the corresponding armWaypoints in sequence) to finish at a given target *armEndpoint* (typically a site or its safe approach position).
DriveThroughSequence	Mobile base (AMR) performs the complete approach motion sequence (by visiting the corresponding baseWaypoints in sequence) to finish at a given target *baseEndpoint* .
OpenGripper	Robot opens the parallel gripper to prepare for gripping or to release the grasped object.
Grip	Robot grips an object (of a specific *labware-Type*) with a parallel gripper.
Dock	Dock the AMR base to a specific *station* (e.g., with a laser scanner target)
Undock	Undock from a *station.*

Table 10: Motion primitive-level Robotic Activity Representations for pick-and-place labware transfer

Command ID	**Description**
MoveToArmWaypoint	Robot arm moves to a specified *armWaypoint* by the means of a joint-optimized *movement* (MoveJ) or via a linear path (MoveL).
DriveToBaseWaypoint	AMR drives to a specified intermediary map position (*baseWaypoint*).
SetFinger	Gripper closes to a specified *force* or *width.*

Table 11: Actuator primitive-level Robotic Activity Representations for pick-and-place labware transfer

Command ID	**Description**
SetJointPosition	Set a specific *joint* of the kinematic chain to a specific *angle* or linear *position* (depending on the type of joint).
SetFingerJoint	Set a specific gripper *joint* to a specific *angle* or *position* (depending on the type of gripper).

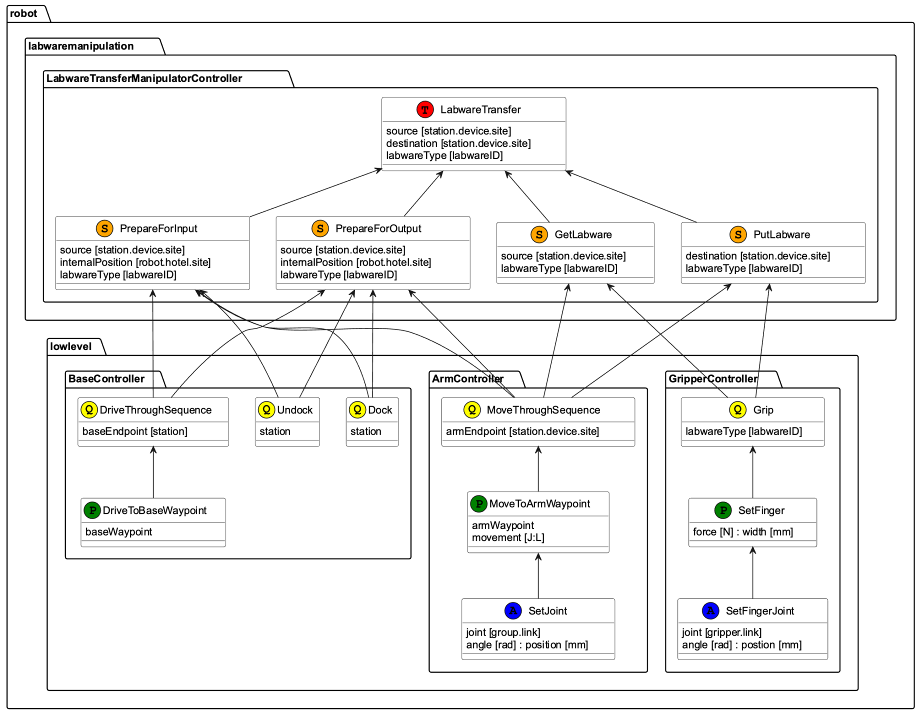


Figure 1: UML-like representation of the LabwareTransfer RARs. Namespaces represent a way of categorizing the RARs. The arrows represent how a high-level RAR is composed (broken down into) low-level RARs. Task (T), Subtask (S), Motion sequence (Q), Motion primitive (P) and Actuator primitive (A) are represented as classes. Parameters are shown as class members, with the data type in square brackets []. Overloaded parameters (in the case of polymorphic commands) are separated by a colon :

In the pseudocode snippet 1 and in Table 2, we present the sequence of a typical labware transfer activity, performed by a mobile manipulator. In a typical scenario, the scheduler would coordinate the *LabwareTransfer* task by calling the *PrepareForInput*, *GetLabware*, *PrepareForOutput* and *PutLabware* subtasks on the robot, in sequence. In the meantime, the source and the destination device should also be prepared for output or input, respectively. In the case of a mobile manipulator, the *PrepareForInput* subtask includes undocking from the previous station (e.g., charger), driving to the next station via a sequence of waypoints, and docking there. After this, the arm needs to be unfolded into a safe/approach position. Note that this motion sequence is also performed by stationary robots.

When the source device is ready for output, the *GetLabware* subtask command makes the arm perform the final approach. This is either a fixed position, relative to the robot base (in case of stationary robots) or a relative position (in the case of mobile manipulators), as outlined in [10]. The gripper then engages with the labware, and then the arm is retracted to a manipulation-ready standby position. This is followed by a MoMa-specific internal subtask to place the labware on the on-board storage (hotel). This RAR can also be considered as a subtask, under another subtask. The self-composition relation is displayed in Figure 1 with a horizontal arrow.

When the *GetLabware* subtask is completed, the source device is notified, and the robot goes on to prepare for placement at the destination device. In case of a MoMa, this involves undocking from station 1, driving to station 2 via a sequence of waypoints, docking, and picking up the labware from the hotel. (Note the self-composition relation, similar to the placement on the hotel.) Then, the arm approaches a safe position to prepare for the actual placement upon command from the scheduler. This, again, is the same in the case of stationary robots. Finally, the *PutLabware* subtask is entirely the same for stationary and mobile robots, and it involves the final approach, releasing the labware and retracting it to a standby position.

Source Code 1: Decomposition of a pick-and-place labware transfer task of a mobile manipulator robot

```
Legend
------------------------------------------------------------------
Indentations represent the levels of the decomposition hierarchy:
Task
    Subtask
        MotionSequence
            MotionPrimitive
                ActuatorPrimitive
* Starred activities are specific for mobile manipulators
------------------------------------------------------------------

LabwareTransfer (station_1.device_1.site_1, station_2.device_1.site_1, DEEP96)
```

```
PrepareForInput (self.site_1, DEEP96)
    *Undock (charger_1)
    *DriveThroughSequence (station_1)
        *DriveToBaseWaypoint (station_1.baseWaypoint_1)
        (...)
        *DriveToBaseWaypoint (station_1.baseWaypoint_n)
        # Final baseWaypoint = station_1
    *Dock (station_1)
    MoveThroughSequence (station_1.device_1.site_1_safe)
    # Safe position
        MoveToArmWaypoint (station_1.device_1.site_1.armWaypoint_1, J)
            SetJoint (joint_1, <angle>)
            (...)
            SetJoint (joint_n, <angle>)
        (...)
        MoveToArmWaypoint (station_1.device_1.site_1.armWaypoint_n-1, J)
            # Safe position, aka. site_approach
GetLabware (station_1.device_1.site_1, DEEP96)
    MoveThroughSequence (station_1.device_1.site_1)
        # Handover position
        MoveToArmWaypoint (station_1.device_1.site_1.armWaypoint_n, L)
        # Final armWaypoint = site_1
    Grip ()
        SetFingers (<force>)
    MoveThroughSequence (manipulation-ready)
        MoveL (device_1.site-approach_1)
        MoveL (device_1.device-approach)
        MoveL (manipulation-ready)
    *InternalPlace (self.hotel.site_1, DEEP96)
PrepareForOutput (self.site_1, DEEP96)
    *Undock (station_1)
    *DriveThroughSequence (station_2)
        *DriveToBaseWaypoint (station_1.baseWaypoint_1)
        (...)
        *DriveToBaseWaypoint (station_1.baseWaypoint_n)
        # Final baseWaypoint = station_2
    *Dock (station_2)
    *InternalPick (self.hotel.site_1, DEEP96)
    MoveThroughSequence (station_2.device_1.site_1_safe)
    # Safe position
        MoveToArmWaypoint (station_2.device_1.site_1.armWaypoint_1, J)
        (...)
        MoveToArmWaypoint (station_1.device_1.site_1.armWaypoint_n-1, J)
        # Safe position, aka. site_approach
PutLabware (station_2.device_1.site_1, DEEP96)
    MoveThroughSequence (station_2.device_1.site_1)
    # Handover position
```

```
            MoveToArmWaypoint (station_2.device_1.site_1.armWaypoint_n, L)
            # Final armWaypoint = site_1
        OpenGripper ()
        MoveThroughSequence (manipulation-ready)
            MoveL (device_1.site-approach_1)
            MoveL (device_1.device-approach)
            MoveL (manipulation-ready)
```

5.3. Advanced robot activities

In table S12 of Appendix S2, we present the extended RAR taxonomy. Here, the scope is broadened beyond the simple pick-and-place labware transfer, and such capabilities are also included that are not yet fulfilled by state-of-the-art laboratory robots. Where appropriate, we cite relevant ongoing research and prototype products. The structure is a non-exhaustive proposal, which is subject to future extensions and amendments, depending on the advancements in lab robot capabilities.

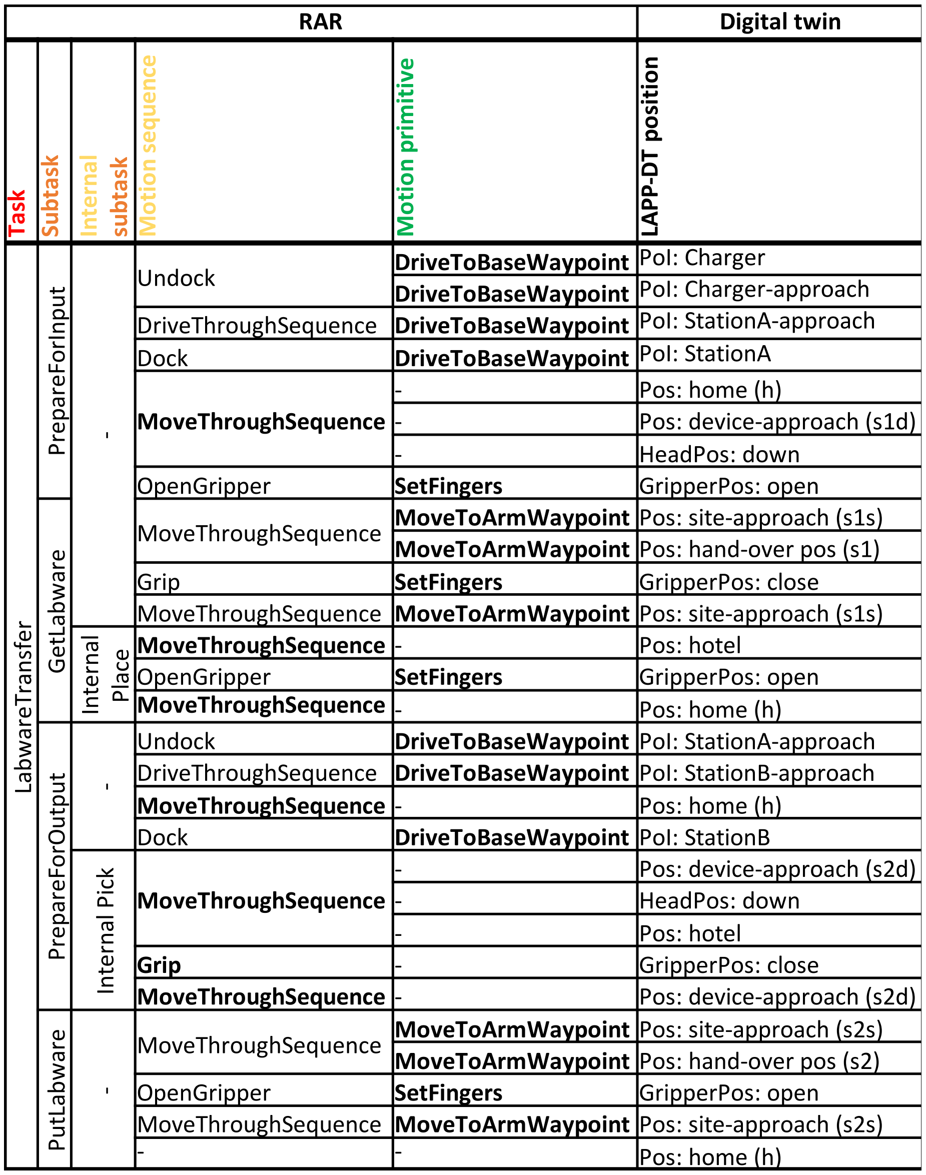
6. Outlooks on the implementation of LAPP

6.1. Technologies

The LAPP framework is a technology-agnostic concept, which provides a blueprint for the integration of supportive robots in life science laboratories. The proposed system architecture can be implemented by a variety of different building blocks. However, to achieve the envisioned plug & play compatibility, the interfaces between these components must be standardized. This involves, as the most important aspect, the communication protocols that enable interoperability of laboratory devices, laboratory robots, and schedulers. At the time of the writing of the article, two major initiatives are present.

The Standardization in Laboratory Automation Consortium (SiLA) [8] is developing a specification and reference implementations for an interoperability standard in laboratory automation. The framework includes a semantic layer, called feature definitions, which can be utilized to implement the proposed task- and subtask-level activities.

As an initiative within the OPC Foundation, the OPC-UA Laboratory and Analytical Device Standard LADS is aiming for a similar goal [36, 37]. The OPC-UA protocol itself is already widespread in industrial automation, especially in process control scenarios. The framework itself has an extensive array of different specifications and extensions, that make it possible to implement several aspects of the LAPP concept. Most prominently, the position representations, as proposed in [10], could be implemented with the Relative Spatial



RAR					Digital twin
Task	Subtask	Internal subtask	Motion sequence	Motion primitive	LAPP-DT position
LabwareTransfer	PrepareForInput	-	Undock	**DriveToBaseWaypoint**	PoI: Charger
				DriveToBaseWaypoint	PoI: Charger-approach
			DriveThroughSequence	**DriveToBaseWaypoint**	PoI: StationA-approach
			Dock	**DriveToBaseWaypoint**	PoI: StationA
			MoveThroughSequence	-	Pos: home (h)
				-	Pos: device-approach (s1d)
				-	HeadPos: down
			OpenGripper	**SetFingers**	GripperPos: open
	GetLabware		MoveThroughSequence	**MoveToArmWaypoint**	Pos: site-approach (s1s)
				MoveToArmWaypoint	Pos: hand-over pos (s1)
			Grip	**SetFingers**	GripperPos: close
			MoveThroughSequence	**MoveToArmWaypoint**	Pos: site-approach (s1s)
		Internal Place	**MoveThroughSequence**	-	Pos: hotel
			OpenGripper	**SetFingers**	GripperPos: open
			MoveThroughSequence	-	Pos: home (h)
	PrepareForOutput	-	Undock	**DriveToBaseWaypoint**	PoI: StationA-approach
			DriveThroughSequence	**DriveToBaseWaypoint**	PoI: StationB-approach
			MoveThroughSequence	-	Pos: home (h)
			Dock	**DriveToBaseWaypoint**	PoI: StationB
		Internal Pick	**MoveThroughSequence**	-	Pos: device-approach (s2d)
				-	HeadPos: down
				-	Pos: hotel
			Grip	-	GripperPos: close
			MoveThroughSequence	-	Pos: device-approach (s2d)
	PutLabware	-	MoveThroughSequence	**MoveToArmWaypoint**	Pos: site-approach (s2s)
				MoveToArmWaypoint	Pos: hand-over pos (s2)
			OpenGripper	**SetFingers**	GripperPos: open
			MoveThroughSequence	**MoveToArmWaypoint**	Pos: site-approach (s2s)
			-	-	Pos: home (h)

Figure 2: Decomposition of the labware transfer sequence. For the positions, we use the nomenclature and signs of [10]

Locations information model [38]. The LADS information model (as an OPC-UA companion specification) adds a specific layer on top of OPC-UA that makes the integration of laboratory devices possible.

In general, tasks and subtasks being triggered by the scheduler need to be interpreted and executed by the laboratory devices' and robots' dedicated control units. Below this layer, the implementation may vary, based on the type of the device's or robot's control system. To break these activities down into geometric motion sequences, the LAPP-conform approach would be to implement parameterized motions and use a position database (the LAPP-DT) to query the specific coordinates for each hand-over position and their intermediary approach sequences. In the case of industrial robot arms, a bridge might need to be introduced between the scheduler and the embedded controller of the robot. This unit must interpret the tasks and subtasks, fetch the coordinates from the DT, and trigger the motions on the embedded controller with these as parameters.

The Robotics Working Group of the SiLA Consortium (SRWG) is actively working on implementing several aspects of the LAPP concept. As such, the LabwareTransferController feature definition [35] implements the Labware transfer sequence, as outlined in 5.2.

During the 4th BioSASH hackathon, a reference implementation was created [39, 40, 41]. There, two different robot arms (a Universal Robots UR 3 and a PreciseFlex PF400 were integrated successfully, based on the proposed system architecture and activity representation models. As a technological backbone, the SiLA standard was utilized.

Besides the reference implementation of the BioSASH hackathon, robot vendor ABB developed a prototype for a SiLA-based control interface for their GoFa collaborative robot, also utilizing the LAPP concept [42].

Besides stationary robots, which are offered by multiple industrial robot vendors, mobile manipulators may also be employed for labware transportation. The LAPP concept is agnostic towards the type of robot and covers both stationary and mobile manipulation. The latter, however, poses additional challenges, mainly due to the additional level of complexity. As summarized in our previous work [7], multiple vendors provide such platforms already. Moreover, extensive academic research is ongoing to address the need for generic manipulation skills in unstructured environments.

To enable the research and development of advanced mobile manipulation capabilities, the Robot Operating System needs to be supported, as this middleware framework is ubiquitous in robotics research. To address this need, the SiLA Robotics Working Group created a reference implementation for a SiLA-ROS bridge [43].

6.2. Future work

A subgroup of the SRWG is currently working on implementing an ontology, which will involve the information models proposed in [7].

Besides, a case study is currently in progress, where the LAPP RAM and the LAPP DT will be implemented, based on SiLA, the LabwareTransferController feature definition, the SiLA-ROS bridge, and the previously-mentioned ontology.

From a future perspective, knowledge representation and symbolic planning techniques can be introduced to a LAPP-conform ecosystem by swapping the layers from the motion sequences down. Instead of predefined positions, advanced perception and cognitive robotics techniques will determine the robotic actions, based on high-level requests coming from the lab execution system [44]. Also, the protocol definitions for lab workflows are envisioned to be separated into a high-level (scientific) protocol and a lab-specific executable program, which is grounded to specific instruments, recreating the symbolic planning aspect on multiple levels.

7. Conclusion

Our motivation with the present paper was to create a reference architecture model for the integration of supportive robotic solutions in life science laboratories. To create a suitable framework, we drew inspiration by evaluating state-of-the-art hierarchical decomposition approaches from different domains. Based on that, we elaborated the Laboratory Automation Plug & Play Reference Architecture Model (LAPP-RAM). Our framework includes a hierarchical decomposition of the laboratory workflows and a breakdown of the corresponding elements of the control architecture and representation protocols. We elaborated in detail on the robotic activity representations (RARs) for pick-and-place labware transfer and created an extended taxonomy for advanced lab robot capabilities. Finally, we provided an outlook on implementing the different aspects of the framework.

Conflict of interest statement

Ádám Wolf is employed with "Baxalta Innovations GmbH", a wholly owned subsidiary of "Takeda Pharmaceutical Company Limited", and owns stock of Takeda and/or its subsidaries/affiliates.

Acknowledgements

This work was funded by Baxalta Innovations GmbH, a Takeda company.

This work was supported by the Doctoral School of Applied Informatics and Applied Mathematics, Óbuda University.

Péter Galambos and Károly Széll thankfully acknowledge the financial support of this work by the project no. 2019-1.3.1-KK-2019-00007 implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme. Péter Galambos is a Bolyai Fellow of the Hungarian Academy of Sciences. Péter Galambos is supported by the UNKP-23-5 (Bolyai+) New National Excellence

Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

References

[1] J. W. Scannell, A. Blanckley, H. Boldon, B. Warrington, Diagnosing the decline in pharmaceutical R&D efficiency, Nature Reviews Drug Discovery 11 (3) (2012) 191–200. doi:10.1038/nrd3681.

[2] Lab automation - Goldfuß engineering GmbH.
URL https://www.goldfuss-engineering.com/en/lab-automation.html

[3] Concept — Robotic Biology Institute Inc. — RBI.
URL https://rbi.co.jp/en/concept/

[4] Andrew Alliance - Pipetting tools and software for the scientific lab.
URL https://www.andrewalliance.com/

[5] Biosero Acceleration Lab, [cited 2021 Oct 18].
URL https://biosero.com/integrations/acceleration-lab/

[6] S. Kleine-Wechelmann, K. Bastiaanse, M. Freundel, C. Becker-Asano, Designing the mobile robot Kevin for a life science laboratory, in: RO-MAN 2022 - 31st IEEE International Conference on Robot and Human Interactive Communication: Social, Asocial, and Antisocial Robots, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 870–875. doi:10.1109/RO-MAN53752.2022.9900786.

[7] A. Wolf, D. Wolton, J. Trapl, J. Janda, S. Romeder-Finger, T. Gatternig, J.-B. Farcet, P. Galambos, K. Széll, Towards Robotic Laboratory Automation Plug & Play: The "LAPP" Framework, SLAS Technology (12 2021). doi:10.1016/J.SLAST.2021.11.003.
URL https://www.sciencedirect.com/science/article/pii/S2472630321000224

[8] SiLA Rapid Integration.
URL https://sila-standard.com/

[9] ROS: Home.
URL https://ros.org/

[10] T. T. Wolf, S. Romeder-Finger, K. Széll, P. Galambos, Towards Robotic Laboratory Automation Plug & Play: Teaching-free Robot Integration with the LAPP Digital Twin (5 2022). doi:10.48550/arxiv.2205.08210.
URL https://arxiv.org/abs/2205.08210v2

[11] Systems Engineering Fundamentals, Defense Acquisition University Press, 2001.

[12] S. A. Brotherton, R. T. Friend, E. S. Norman, Applying work breakdown structure to project lifecycle, Denver, CO., 2008.
URL https://www.pmi.org/learning/library/applying-work-breakdown-structure-project-lifecycle-6979

[13] Difference and use cases of Jira issue types: Epic... - Atlassian Community.
URL https://community.atlassian.com/t5/Jira-articles/Difference-and-use-cases-of-Jira-issue-types-Epic-vs-Story-vs/ba-p/1655157

[14] Epics, Stories, Themes, and Initiatives — Atlassian.
URL https://www.atlassian.com/agile/project-management/epics-stories-themes

[15] B. Bartley, J. Beal, M. Rogers, D. Bryce, R. P. Goldman, B. Keller, P. Lee, G. Bioworks, V. Biggers, J. Nowak, Building an Open Representation for Biological Protocols, bioRxiv (2022) 2022.07.05.498808doi:10.1101/2022.07.05.498808.
URL https://www.biorxiv.org/content/10.1101/2022.07.05.498808v1https://www.biorxiv.org/content/10.1101/2022.07.05.498808v1.abstract

[16] X. Chu, Automation Strategies for Sample Preparation in Life Science Applications, Ph.D. thesis (2016). doi:10.18453/rosdok{_}id00002364.

[17] T. D. Nagy, T. Haidegger, A DVRK-based Framework for Surgical Subtask Automation, Acta Polytechnica Hungarica 16 (8) (2019). doi:10.12700/APH.16.8.2019.8.5.

[18] S. S. Vedula, A. O. Malpani, L. Tao, G. Chen, Y. Gao, P. Poddar, N. Ahmidi, C. Paxton, R. Vidal, S. Khudanpur, G. D. Hager, C. C. G. Chen, Analysis of the structure of surgical activity for a suturing and knot-tying task, PLoS ONE 11 (3) (3 2016). doi:10.1371/journal.pone.0149174.

[19] D. Leidner, C. Borst, G. Hirzinger, Things are made for what they are: Solving manipulation tasks by using functional object classes, IEEE-RAS International Conference on Humanoid Robots (2012) 429–435doi:10.1109/HUMANOIDS.2012.6651555.

[20] BPMN Specification - Business Process Model and Notation.
URL https://www.bpmn.org/

[21] ISA 95.00.01-2010 Enterprise-Control System Integration - Part 1: Models and Terminology.

[22] A. Wolf, P. Galambos, K. Széll, Device Integration Concepts in Laboratory Automation, in: INES 2020 - IEEE 24th International Conference on Intelligent Engineering Systems, Proceedings, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 171–177. doi:10.1109/INES49302.2020.9147171.

[23] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Business and Information Systems Engineering 6 (4) (2014) 239–242. doi:10.1007/S12599-014-0334-4/FIGURES/1.
URL https://link.springer.com/article/10.1007/s12599-014-0334-4

[24] F. Schnicke, T. Kuhn, P. O. Antonino, Enabling industry 4.0 service-oriented architecture through digital twins, Communications in Computer and Information Science 1269 CCIS (2020) 490–503. doi:10.1007/978-3-030-59155-7{_}35/FIGURES/2.
URL https://link.springer.com/chapter/10.1007/978-3-030-59155-7_35

[25] Standards — SiLA Rapid Integration.
URL https://sila-standard.com/standards/

[26] Reference Architecture Model Industrie 4.0 (RAMI4.0) English translation of DIN SPEC 91345:2016-04 (2016).

[27] Universal Robots e-Series User Manual UR10e Original instructions (en) UR10e User Manual (2009).

[28] H. Fleischer, K. Thurow, Automation Solutions for Analytical Measurements: Concepts and Applications, John Wiley & Sons, 2017.

[29] D. Knobbe, H. Zwirnmann, M. Eckhoff, S. Haddadin, Core Processes in Intelligent Robotic Lab Assistants: Flexible Liquid Handling, IEEE International Conference on Intelligent Robots and Systems 2022-October (2022) 2335–2342. doi:10.1109/IROS47612.2022.9981636.

[30] D. S. Leidner, Cognitive Reasoning for Compliant Robot Manipulation, Doctoral Thesis (2017) 211.
URL http://www.springer.com/series/5208

[31] A Model for Biotech Laboratory Infrastructure — by Maximilian Schulz — Ageless Society.
URL https://ageless.blog/a-model-for-biotech-laboratory-infrastructure-81c9b96a88c8

[32] P. Courtney, New trends in intelligent robotics in the laboratory, European Pharmaceutical Review 21 (2) (2016) 36–38.
URL www.euroc-project.eu

[33] A. Wolf, K. Széll, A review on robotics in life science automation, in: AIS 2019 14 th International Symposium on Applied Informatics and Related Areas organized in the frame of Hungarian Science Festival 2019 by Óbuda University PROCEEDINGS, 2019, pp. 106–111.

[34] H. Fleischer, R. R. Drews, J. Janson, B. R. Chinna Patlolla, X. Chu, M. Klos, K. Thurow, Application of a Dual-Arm Robot in Complex Sample Preparation and Measurement Processes, Journal of Laboratory Automation 21 (5) (2016) 671–681. `doi:10.1177/2211068216637352`.

[35] feature_definitions/org/silastandard/instruments/labware/manipulation · master · SiLA2 / sila_base · GitLab.
URL `https://gitlab.com/SiLA2/sila_base/-/tree/master/feature_definitions/org/silastandard/instruments/labware/manipulation`

[36] A. Brendel, F. Dorfmüller, A. Liebscher, P. Kraus, K. Kress, H. Oehme, M. Arnold, R. Koschitzki, Laboratory and Analytical Device Standard (LADS): A Communication Standard Based on OPC UA for Networked Laboratories, Advances in Biochemical Engineering/Biotechnology 182 (2022) 175–194. `doi:10.1007/10{\_}2022{\_}209/COVER`.
URL `https://link.springer.com/chapter/10.1007/10_2022_209`

[37] OPC 30500-1: Laboratory and Analytical Devices.
URL `https://opcfoundation.org/documents/30500-1/`

[38] Industrial Automation - Relative Spatial Location - 5 Relative Spatial Location Information Model overview.
URL `https://reference.opcfoundation.org/RSL/v100/docs/5`

[39] Successful hackathon in the bioSASH project: video series now online!
URL `https://www.biolago.org/de/blog/details/successful-hackathon-in-the-biosash-proejcjt-video-now-online.html`

[40] bioSASH - Pick & place labware transportation with benchtop and mobile robots - YouTube.
URL `https://www.youtube.com/watch?v=49YD7MzXBL8`

[41] bioSASH final result of hackathon Pick & place labware transportation with benchtop and robots - YouTube.
URL `https://www.youtube.com/watch?v=G2hN5eSVpAI`

[42] Post — LinkedIn.
URL `https://www.linkedin.com/posts/lukas-bromig_futurelabs-sila-labofthefuture-activity-7069780030683250688-h1qw/?utm_source=share&utm_medium=member_desktop`

[43] SiLA2 / sila_robotics / sila_ros · GitLab.
URL `https://gitlab.com/SiLA2/sila_robotics/sila_ros`

[44] M. Beetz, L. Mösenlechner, M. Tenorth, CRAM - A Cognitive Robot Abstract Machine for everyday manipulation in human environments, IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings (2010) 1012–1017`doi:10.1109/IROS.2010.5650146`.

[45] Disinfection solution with UV-equipped mobile robots — OMRON, Europe.
URL `https://industrial.omron.eu/en/news-events/news/disinfection-solution-with-uv-equipped-mobile-robots`

[46] A. Schmelzer, V. Miegel, A Novel Approach To Advanced Cleaning In The Pharmaceutical Industry; 2 Worlds – 1 Robot (4 2022).

[47] A. Wolf, P. Troll, S. Romeder-Finger, A. Archenti, K. Széll, P. Galambos, A Benchmark of Popular Indoor 3D Reconstruction Technologies: Comparison of ARCore and RTAB-Map, Electronics 9 (12) (2020) 2091. `doi:10.3390/electronics9122091`.
URL `https://www.mdpi.com/2079-9292/9/12/2091`

Supplementary Data for: Towards Robotic Laboratory Automation Plug & Play: Reference Architecture Model for Robot Integration

Ádám Wolf [a,b,*], Panna Zsoldos[c], Károly Széll [d], Péter Galambos [c]

[a] *Baxalta Innovations GmbH, a Takeda company, Industriestraße 67, A-1221 Wien, Austria*
[b] *Doctoral School of Applied Informatics and Applied Mathematics, Óbuda University*
[c] *Antal Bejczy Center for Intelligent Robotics, Óbuda University*
[d] *Alba Regia Technical Faculty, Óbuda University, H-8000 Székesfehérvár, Hungary*

Contents

S1. Results of the survey

See `Commands_survey.xlsx`.

S2. LAPP-RAP - Extended taxonomy

Table S12: Namspaces

Namespace	**Description**
cleaning / CleaningController	Cleaning operations, e.g., area coverage with vacuum cleaner or UV desinfection robot [45, 46]
cleaning / SprayController	Spray a surface with a liquid, e.g., for desinfection
cleaning / WipeController	Wipe a surface, e.g., with a sponge [30]
devicemanipulation / BayController	Set a bay storage (e.g., carousel) to a desired position, so that a certain site can be accessed.

*Corresponding author
Email address: `adam.wolf@takeda.com` (Ádám Wolf)

devicemanipulation / HatchController	Manipulate various hatches of devices, e.g., drawers, doors, sliding door, lids. Latching must be taken into account.
devicemanipulation / UIController	Interface with the physical user interface of a device, e.g., by push buttons, rocking switches, sliders, knobs etc.
humaninteraction / SpeechService	Human-robot interaction through text-to-speech. Let the robot communicate with the human through speech.
labwaremanipulation / CapController	Apply and remove screw caps: cap and decap.
labwaremanipulation / ClampController	Control the flow in a tube by applying a clamb externally.
labwaremanipulation / ConnectorController	Connect lines, e.g., with aseptic connectors.
labwaremanipulation / labelController	Robot labels a labware for simple marking e.g., with a marker. Plotter-like operation.
labwaremanipulation / LabwareTransferController	Pick-and-place labware transfer [35, 44]
labwaremanipulation / LidController	Lid and delid. Covers that are not screw-on. In its simplest form just place the lid over the container. In more complex cases the operation of some locking mechanism might be necessary, e.g., snap, latch or bayonet.
labwaremanipulation / LidFlipController	Open and close flip lids.
labwaremanipulation / PackagingController	Pack and unpack labware, remove or add single-use packaging. e.g., peal off packaging before loading the labware in a device, or pack it before shipment.
labwaremanipulation / SlideInController	Subtask-level pick and palce actions where the labware must be slid onto a platform when inserted into a device / site.
labwaremanipulation / TrolleyController	Move wheeled object, e.g., a trolley by docking to it or grasping its handle and pushing or pulling it.
labwaremanipulation / TubeHandlingController	Tube handling, e.g., loading into tube welder. Handle tangling tubes with advanced perception, e.g., tactile sensing and / or vision. Set-up-and-leave: tubes in predefined fixtures.
lowlevel / ArmController	Low-level (action primitive), not task-specific control of the robot arm. E.g., move rotation, move linear, approach.

lowlevel / BaseController	Low-level (action ptimitive), not task-specific control of the mobile base. e.g., navigat intermediary.
lowlevel / GripperController	Low-level (action primitive), not task-specific control of the gripper (end effector).
maintenance / BatteryController	Battery management, e.g., send to charge.
maintenance / ConfigurationController	Configuration management, e.g., change end effector tool.
maintenance / InitializationService	Reinitialize the robot
maintenance / MapService	Maintain the map of the premises and the base positions for robot navigation.
maintenance / PositionService	Maintain the device's position (e.g., nests).
maintenance / ProgramController	Exposes the ability to start, stop and pause predefined programs saved in the robot. For example: a hard-coded robot program.
maintenance / StatusProvider	Provide information about the health, availability and current activities of the robot.
maintenance / TeachingService	Teach robot positions.
maintenance / ValidationService	Validate a robot program in regard to collaborative requirements.
perception / BarcodeProvider	Scan a barcode with an on-board camera or barcode scanner.
perception / LiquidLevelProvider	Detect the liquid level in a container, e.g., by computer vision.
perception / ObjectDetectionProvider	Provide a known or unknowh object's pose by detecting it in a 2D or 3D image. Typical use case is grasp detection or object tracking.
perception / PhotoProvider	Take a photo with an on-board camera, e.g., for audit or remote troubleshooting purposes.
perception / PresenceProvider	Use a sensor to detect the presence of an object.
perception / ShapeProvider	3D reconstruction of a room or object. [47]
samplemanipulation / PipetteController	The robot performs a liquid transfer with a pipette, (hand-held or integrated). [29]
samplemanipulation / PourController	The robot performs a liquid transfer by pouring from one container to another one.

samplemanipulation / ShakeController	The robot mixes a sample by performing the predefined periodic motion (e.g., rocking or shaking). Frequency is limited compared to vortex shaker devices.
samplemanipulation / StirController	The robot mixes a sample with a submerged tool.
samplemanipulation / VortexController	The robot places the sample on a vortex mixer device's platform to perform the shaking. Admittance or soft gripper is necessary.