Methods

Linus Witucki*, Marwin Madsen, Eric L. Wagemann and Mike Barth

Introduction of a unified robot intergration package

Einführung des Unified Robot Integration Package

https://doi.org/10.1515/auto-2025-0007
Received January 18, 2025; accepted March 21, 2025

Abstract: In modern manufacturing environments, robotic systems are increasingly becoming essential components of various production processes. The growing diversity of these systems require innovative approaches to simplify system integration. This contribution addresses the challenges posed by the heterogeneity of robotic systems, which often lead to complex engineering processes. To address this problem, the authors introduce a novel architecture called the Unified Robot Integration Package (URIP). URIP aims to accelerate system integration through modularization and encompasses three key aspects: control and orchestration, functionality description, and human-machine interface. By providing a unified description of robotic systems, URIP reduces the complexity associated with system integration, resulting in more efficient and flexible adaptation to changing requirements.

Keywords: robot; modularization; information modelling; ROS

Zusammenfassung: In modernen Fertigungsumgebungen werden robotische Systeme zunehmend zu wesentlichen Aspekten verschiedener Produktionsprozesse. Die wachsende Vielfalt dieser Systeme erfordert innovative Ansätze zur Vereinfachung der Systemintegration. Dieser Beitrag behandelt die Herausforderungen, die durch die Heterogenität robotischer Systeme entstehen und oft zu komplexen Entwicklungsprozessen führen. Um dieses Problem zu lösen, stellen die Autoren das Unified Robot Integration Package (URIP) vor. URIP zielt darauf ab, die Systemintegration robotischer Systeme durch Modularisierung zu beschleunigen und umfasst drei wesentliche Aspekte: Steuerung und Orchestrierung, Funktionsbeschreibung und Mensch-Maschine-Schnittstelle. Durch die Bereitstellung einer einheitlichen Beschreibung robotischer Systeme reduziert URIP die Komplexität der Systemintegration, was zu einer effizienteren und flexibleren Anpassung an sich ändernde Anforderungen führt.

Schlagwörter: Robotik; Modularisierung; Informationsmodellierung; ROS

***Corresponding author: Linus Witucki**, Institute of Control Systems, Karlsruher Institute of Technology, Kaiserstraße 12, D-76131 Karlsruhe, Germany, E-mail: linus.witucki@kit.edu. https://orcid.org/0009-0003-4973-9471
Marwin Madsen and Eric L. Wagemann, Karlsruher Institut für Technologie (KIT) Institut für Regelungs- und Steuerungssysteme (IRS), Karlsruhe, Germany. https://orcid.org/0009-0006-9953-2382 (M. Madsen). https://orcid.org/0009-0008-8420-644X (E. L. Wagemann)
Mike Barth, Karlsruhe Institute of Technology: Karlsruher Institut für Technologie, Institute for Control Systems, Fritz-Haber-Weg 1, 76131 Karlsruhe, Germany. https://orcid.org/0000-0003-2337-063X

1 Introduction

In a modern manufacturing environment, robot systems are becoming an integral part of an increasing number of different production processes [1]. Due to the specific requirements of different manufacturing processes, e.g. complex or repetitive mounting tasks, handling of small-scale parts, high-precision tasks, or even heavy-duty processes, the variety of robots is constantly increasing. In recent decades, robots and their environments have become more heterogeneous, with different types, configurations, and vendors. Such different types include articulated, delta, cartesian, SCARA, collaborative and soft robots. Due to differences in engineering basics, e.g. tools, languages, terms, and definitions, this variety often leads to a more complex engineering process for robot systems. In addition, robots of these types are regarded as incomplete machines. They are designed to be combined with additional components, such as an end effector. The resulting combined robot systems offer composite functionalities, as combinations of the individual functionalities of the components [2]. For example, a soft robot gripper attached to an articulated robot to handle fragile objects or objects with unknown geometry [3], can provide a pick-and-place functionality. This system integration can be a demanding task, when complex behavior is required. Often times this leads to the expensive outsourcing of the system integration. Currently, this custom engineering of robotic systems is a major cost factor for robotic automation, of up to 70 %, [4]. Additionally, as most robot systems are engineered for specific requirements and functionalities, a requirements change could immediately lead to a different robot type or auxiliary component and therefore additional integration cost.

One opportunity to reduce the complexity and accelerate the engineering of robot systems is modularization [5]. In the process industry, modularization concepts are already defined by the Module Type Package (MTP) [6]. Due to the differences between the production equipment of the process industry (pipes, vessels, pumps) and robots, it is not sufficient to directly use MTP to describe robot systems. This will be explained in detail in Section 2.

Facing existing drawbacks, this work presents a novel architecture aimed at accelerating the engineering of robotic systems by means of modularization. This architecture is termed Unified Robot Integration Package (URIP). During the design phase of URIP, including a detailed state-of-the-art analysis, three crucial aspects of the engineering process have been derived:

1. control and orchestration,
2. functionality description,
3. human-machine interface.

This article outlines the overall structure and objectives of each aspect to explain the new concept for the first time, though detailed exploration is beyond its scope and will be the subject of future work.

The remainder of this article is structured as follows. In Section 2 related work is outlined from different domains. The structure of the URIP, divided into the mentioned aspects, is established in Section 3. An exemplary robot system is provided in Section 4. Lastly, a summary and an outlook for future research regarding the extension of the URIP are given in Section 5.

2 Related work

One of the major goals of the presented architecture is to establish a generic basis applicable to different sectors such as chemical, pharmaceutical, life science, food and beverage, oil and gas production plants, as well as the discrete manufacturing and assembly industry. For this, existing modularization efforts from different domains have been analyzed.

As mentioned in Section 1, a prevalent example of modularization in process plants is the MTP. The MTP, defined in the VDI/VDE/NAMUR 2658 [6], describes a modular process plant as an aggregation of process equipment assemblies (PEAs). A PEA can be, for example, process equipment such as distillation units [7]. Such a PEA is made up of functional equipment assemblies (FEAs). A FEA can be a controlled flow subsystem within the dosing unit. The monitoring and coordination of PEAs is implemented within a process orchestration layer (POL) [6]. The POL is usually a process control system that imports the MTP information models and uses them to orchestrate the overall configuration of the process plant, according to a process recipe. The MTP is designed specifically for process automation. For instance, the generation of standardized human-machine interfaces (HMIs), as defined by the MTP in part 2, only considers PEAs like pumps, tanks, or valves and their respective schematics. In order to integrate robotic systems into the MTP concept, all advertised functionalities must be wrapped in state machines. For granular functions like opening a gripper, this is not feasible, as most granular robotic functionalities are provided as remote procedure calls (RPCs), which can be directly invoked. When regarding higher level functionality like pick and place, a state machine becomes more advantageous. This results in the programming of the state machine, integrating all system components. Currently, the MTP does not provide assistance for this integration step, making it a manual process. This increases the burden on the module manufacturer. Therefore, the MTP concept is not directly suitable for robotic integration, necessitating the formulation of a streamlined approach of modular robotic systems.

The direct usage of the MTP with robotic systems is illustrated well in [8]. Here, the author utilizes the Robot Operating System (ROS) for hardware abstraction. The ROS offered robot functionalities are directly integrated with the Open Platform Communications Unified Architecture (OPC UA) in line with the MTP service structure by using *mtppy* [9]. The software *mtppy* provides a service framework for an MTP compliant state machine. This MTP adaptation considers robots only from the control aspect and does not regard HMIs. Additionally, the *mtppy* approach limits the consideration of complex robot systems, due to the rigid code structure. The reference implementation, in [8], directly integrates the functionalities of an articulated robot arm and an attached gripper into the POL. A *Move-Service* is implemented to trigger robot movement and a *Hand-Service* for gripper opening and closing. A composite functionality like pick-and-place therefore needs to be orchestrated directly in the POL. When considering the robot and gripper as a module offering a composite pick-and-place functionality, the timing of the services need to be implemented within the module's service. This example highlights that while it is possible to represent a robot using the MTP, there is currently no approach to structuring a robotic system and the advertised functionality.

In addition to MTP adaptations towards articulated robots, the MTP concept has already been utilized for process-oriented intralogistics units [10]. This concept involves implementing a Transport Coordination System (TCS) as a PEA and utilizing a Logistic Orchestration Layer (LOL) instead of the POL. The LOL manages orders, assets and transports [11]. The TCS communicates with a mission manager that coordinates transport tasks with automated guided vehicles (AGVs). This concept extends the scope of the MTP to AGVs [12]. However, the AGV is not directly considered as the mission manager, and the TCS abstract the AGVs functionalities. Therefore, this approach is not suitable for the direct representation of robotic systems.

So far, the MTP only considers communication via OPC UA [13]. OPC UA provides both cyclic Client/Server and event-based Pub/Sub communication. To provide the Pub/Sub pattern for MTP, OPC UA can be combined with the Message Queuing Telemetry Transport (MQTT) protocol [14]. These communication patterns are usually not provided by robots. Often, the communication method of a robot is based on the use of RPCs within a manufacturer-specific Application Programming Interface (API). To wrap these custom RPCs, ROS or ROS2 are commonly used [15]. ROS enables a Service-Oriented Architecture (SOA) of robot systems by wrapping the robot functionality into ROS services. There are also attempts to implement a software bridge between ROS and OPC UA [16]. These can be utilized to control robot functionality with non-ROS-enabled software nodes. In view of the diverse use of RPCs in robot systems, a modular concept RPCs and the ROS architecture must be taken into account.

Another aspect that needs to be taken into account when modeling modularized robot systems is the PPR-Method, which splits a production entity into product (P), process (P), and resource (R) [17]. Its focus is on the definition of skills that can be supplemented to describe the relationship between processes and resources [18], which in turn allows a layer of abstraction [19]. For example, a skill can be "welding" – or "pick and place". This approach has already been implemented to orchestrate robot systems, using the so-called skill execution engines (SEE) [20]. Expending on the idea of PPR and production skills [2], [21], introduces the Capability Skill Service (CSS) model. The CSS model defines (a) capabilities as the description of implemented skills in a process-relevant context, (b) a skill as the implementation of a capability, and (c) services as a way to offer capabilities to external partners via a marketplace. For example, the capability "pick and place", utilizes implemented skills like "closing of a gripper" and "control of a robot". The capability can then be wrapped into an "automated material handling" service.

A comparison can be drawn between the CSS and MTP concepts. The skill describing the implementation of a functionality advertised by a system as defined by the CSS model can be compared to a procedure, defined as the realization of an abstract functionality in the MTP. A capability, the description and abstraction of a skill, can be compared to the service advertised by a PEA defined in the MTP [22].

Another method of modeling and abstracting intralogistics systems is the System Architecture for Intralogistics (SAIL) defined in [23]. SAIL describes how a material flow facility is subdivided into SAIL components. This architecture specifies the way functionalities are encapsulated, how interfaces are defined, and how control components are provisioned. This function-centric modeling approach promises to lower costs in all engineering phases by improving the reusability of components and, therefore, lowering engineering overhead for reconfiguration of systems [23]. However, the SAIL concept is specifically designed for conventional material flow technology such as conveyor belts or AGVs. Therefore, it specifically models the material flow through a production environment by defining, for example, the *Transport Coordination* and *Direction Control* functionalities. A welding robot, for example, cannot be modeled with such functionalities efficiently. It is not designed to use articulated robot functions.

In conclusion, many efforts of modularization already exist. Each of the presented concepts approach modularization from a different perspective and in different domains. As elaborated, each method has different reasons for why a direct application for robotic systems is not feasible. This establishes the need for action to adapt and combine aspects of the existing methods towards robotic systems.

3 Concept of a unified robot integration package

Similar to related work in the process industry and intralogistics, robot systems can be modularized. In this article, a package description is developed to describe such modular robotic systems. The proposed Unified Robot Integration Package (URIP) is an MTP-like package which models data used during engineering and operation. Therefore, URIP includes aspects of a robot-specific core concept such as control, functionality description, and HMI, during the engineering and operation of robot systems.

3.1 Requirements

In the following, several requirements for the URIP are formulated and set in relation to the overall goals of the concept.

To face the formulated challenge of heterogeneous, vendor- and type-specific robot systems, a major goal of the concept presented is to develop a generic, vendor-independent solution. This means a decoupling of software and hardware architectures. In order to accomplish this decoupling, a hardware abstraction layer is necessary [24]. This allows for reuse of software components between various robot systems of different types and vendors. Based on this hardware abstraction, robot controllers can be implemented and coordinated within a module. Therefore, a substantial requirement of the URIP concept is as follows:

REQ 1 Provision of hardware abstraction for robot controllers and implementation of module-wise component coordination.

To be able to integrate the engineered robotic module into an overarching orchestration system, a description of the module functionality is necessary. The related requirement is formulated as follows:

REQ 2 Integration of a generic description model for all orchestration-related functionalities that a module provides.

During the operation of the robot module, am HMI is necessary. The use of several subsystems leads to various subsystem-HMIs which need to be combined to a higher-level operator HMI. This leads to the following requirement:

REQ 3 Provisioning of an HMI, representing the robot module with all implemented (combined) functionalities.

In order to fulfill these requirements, the URIP concept defines three corresponding aspects: robot control, functionality description, and HMI which are combinable across multiple robot modules in order to describe a complex robot system. The URIP is therefore an aggregation of engineering data originating in different domains or expert teams. The focus of this article is on the discussion of the three identified requirements.

To get an understanding of the terms used in the sense of URIP, the following section first presents how a robot module can be decomposed into several subsystems and elementary components.

3.2 Module decomposition

The URIP uses the term *robot module* as a combination of one or multiple robotic components and optional auxiliary components. A robot component can be defined as a robot as defined in *ISO 10218* [25] or a mobile robot component following the definition of *ISO 13482* [26]. A robot component is integrated with a vendor-specific controller. Auxiliary components are assets which are attached to robots, and their controllers. They include, for example, safety equipment, like light curtains. This type of component does not include any standalone controller and is supposed to be integrated into the robot controller.

The combination of multiple robot components constitutes a robot module. The example in Figure 1 illustrates how a mobile robot manipulator can be decomposed into sensible parts, based on the previous explanations. When a robot module is combined with additional components, a hierarchical structure is applied. Each component and combination of components can be represented as a robot module with a unique URIP.

3.3 Core concept aspects

The following sections describe the core concepts incorporated with the URIP, starting with the control aspect,

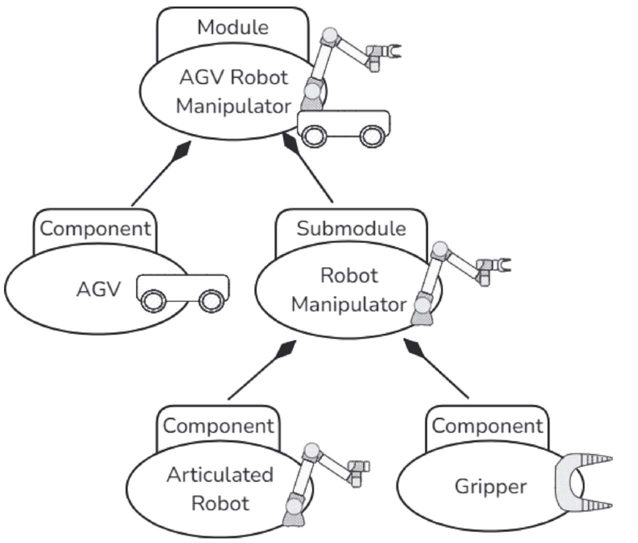

Figure 1: Definition of subdivisions by means of an exemplary AGV robot manipulator.

followed by the description of the robot's functionalities and the module HMI concept.

3.3.1 URIP-aspect 1: robot control

This section will elaborate on how the URIP implements a comprehensive, abstract, and modular robot control mechanism.

When integrating a robot system, it becomes clear that the control methods of robot systems vary drastically with each vendor. Each robot provider implements a custom controller structure for their robots, including custom programming languages and APIs. As mentioned in Section 1, the multitude of vendor-specific solutions leads to a drastic overhead in engineering effort. To alleviate this fact, the overarching goal of URIP control component is to provide a comprehensive, vendor-abstracted robot control system. This system aims at providing a unified interface for controlling various robotic systems, regardless of the manufacturer. In this way, the exposed functionalities of the robot module satisfy the requirement REQ 1.

As depicted in Figure 2, in the core, the control aspect is based on the API offered by the vendor, which exposes the functionality offered by the vendor. In order to achieve a

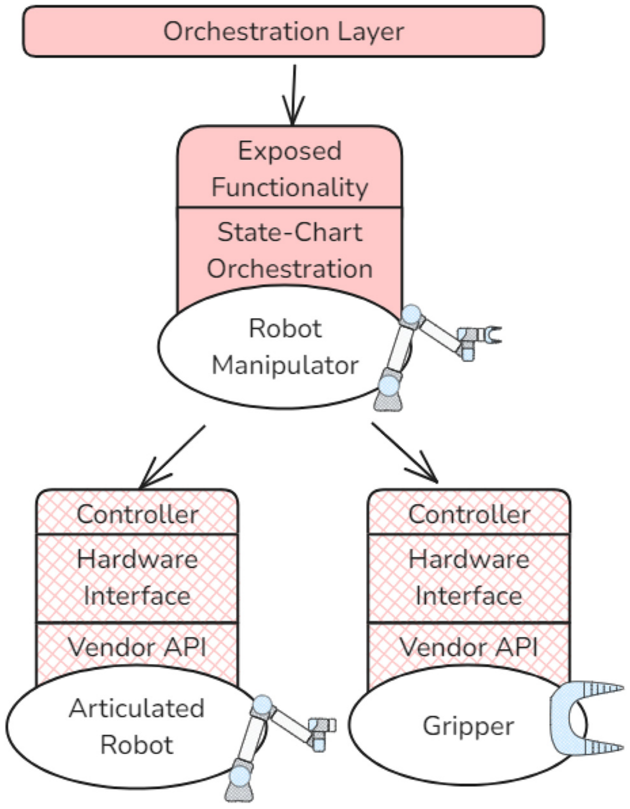

Figure 2: Control aspect of the URIP.

generalized robot integration, a hardware abstraction layer is necessary, standardizing the robot interface. On top of this hardware interface, generic robot controllers can be implemented, that latch onto the hardware interface functionality, offering control over granular robot functionality, such as Tool Center Point (TCP) or joint trajectory movement. Due to the standardized nature of the hardware interface, implemented controllers can be reused across different robot systems. Such controllers can be provided for each robot component of a robot module.

For complex robot behavior and multi-robot modules, coordination of all submodules and components is necessary to provide the needed robot functionality. To this end, choreography or orchestration methods can be utilized.

Both concepts are rooted in computer technology and web service architectures. Orchestration utilizes a single coordination process, which describes the flow of operations. Choreography is a decentralized method of coordinating multiple systems, with direct peer to peer communication between each system [27].

With regard to automation technology, the usage of orchestration is more common, than choreographies [28].

When implementing a choreography of multiple controlled systems, each system needs the knowledge about the following systems functionalities. Therefore, the overall system structure must be clear during the implementation of granular functionalities. For example, when implementing an articulated robot with a gripper attachment using the choreography method, the robot controller interacting with the gripper is specific for that attachment. If the robot is afterward used with a completely different, or even without a gripper, the controller implementation needs to be redone. Based on this reasoning, the authors recommend module internal orchestration within a superordinate system, as depicted in Figure 2. However, the possibilities of choreography implementations for modular robot systems need to be explored in further research.

The orchestration will be provided by module-specific statecharts, where granular robot control is used to coordinate robot functionalities. By adopting a modular approach to statecharts and utilizing orthogonals as defined by Harel [29], it is ensured that the implementation of robot components behavior within statecharts is reusable, given the generic interface of the granular robot controller. In this context, orthogonality refers to the ability to split a state into independent concurrent sub-states, each representing a different aspect or subsystem. For example, orthogonality can be used to handle different hardware components such as sensors, cameras, and actuators independently, but concurrently.

Implementation-wise, the *ros2_control* stack [30] can be followed to implement the hardware interface and the reusable robot controller. ROS2 provides a widely popular middleware, defining standardized interfaces and provides a large library of robotics tools. A component using ROS2 to communicate with other components is called a node. The communication patterns can follow a Pub/Sub or call-response (ROS-Service/Action) pattern.

When a module with multiple robot components is concerned, each will need a dedicated *ros2_control* node. Each of these can be deployed on different or the same hardware within the module. An exemplary distribution of an articulated robot manipulator is depicted in Figure 3. The robot and gripper are connected separately to a vendor offered hardware controller and an I/O-Link interface. Both systems provide an API, which can be utilized by the depicted real-time operating system (RTOS) enabled computer. On the RTOS-system two ROS2 controllers are implemented, one generally available joint trajectory controller (JTC) node (1), and a separate gripper controller node (2). These controllers establish a ROS2 action and service based interface within the ROS2 network. A superordinate orchestration system utilize these in order to offer complex pick and place behavior. Regarding this architecture, only elementary components will provide a hardware interface and direct controller, as indicated by the crosshatch pattern in Figure 2. All (sub-)modules implement the described orchestration of the internally advertised functionalities of comprised components. Implementation-wise, this can result in a ROS2 workspace with a controller workspace for each elementary module and state chart workspaces for each (sub-)module.

In order to implement the statecharts, *SMACC2* [31] can be used. With this approach, state charts similar to those defined by ISA-88 [32] can be implemented. The implementation of such a module statechart is described in [33]. A resulting exposed functionality can be serviced to a high-level orchestration system that can commission the robot module functionality via the standardized transition signals, which can be provided as ROS2 services in the appropriate states.

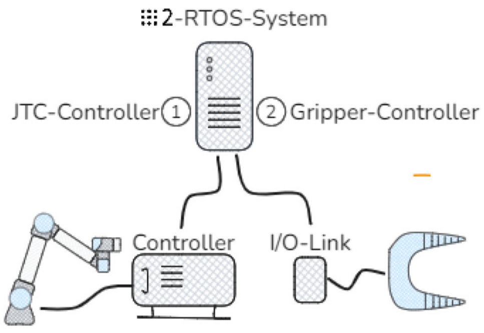

Figure 3: Exemplary controller deployment architecture.

With this method, the orchestration of robot modules is possible, similar to [8], [20], but providing only the functionalities of modules relevant to the high level process rather than the functionalities of granular robot component.

In conclusion, the requirement for provisioning a hardware abstraction for controller and orchestration implementation is fulfilled through modular robot orchestration based on a generic granular robot controller and the utilization of hardware and vendor abstraction via the hardware interface. The ROS2 control stack can be followed to implement the hardware interface and reusable robot controller. For the Harel statechart implementation, the ROS2 package, SMACC2, can be used to interface with the generic ROS2 controllers. This comprehensive method ensures an effective approach to flexible, reusable, and vendor-agnostic functionality of robot modules. The ROS2 and *SMACC2* implementations proposed in this section have been tested in the example explained in Section 4.

3.3.2 URIP-aspect 2: descriptive aspect

The overarching goal of this aspect is to achieve seamless integration of module functionalities within an over all ROS Orchestration Layer (RosOL). This integration aims to ensure that various modules, regardless of their specific functionalities, can work together cohesively within a production environment.

Integration is heavily dependent on the implementation details of module functionalities, such as procedures and parameters. This dependency creates challenges in achieving a standardized approach to integration, as each module may have unique requirements and interfaces.

To address this issue and therefore satisfy REQ 2, the authors propose the integration of a generic description model for all orchestration-related functionalities that a module provides. This solution involves using the CSS [2] approach. As described in Section 2, the CSS method is a structured approach to defining and implementing functionalities within industrial systems. In this method, a capability is an implementation-independent specification of a function in industrial production to achieve an effect in the physical or virtual world. A capability may be implemented by one or more skills. A skill is an executable implementation of an encapsulated function specified by a capability.

For modular robotic systems, this method can be adapted. A robot module, consisting of submodules and components, exposes which can be described as capabilities. These capabilities choreograph the functionalities of submodules and components within the module hierarchy. Therefore, the module capability is considered a composite capability. Only the elementary components of each (sub-)module implement the skills out of which the (sub-)modules form their capabilities. For example, if a composite capability involves picking, moving, and placing an object, it will utilize the skills associated with each of these actions (pick, move, place), which are implemented within the elementary components, articulated robot and robot gripper, rather than implementing a new skill for the entire sequence. The described capability and skill decomposition is depicted in Figure 4. The goal of accessing and orchestrating the composite capabilities of the robot module entails the need for information models that describe the capabilities of the module, the elementary skills of the components and the connections of the skill interface. This information model will facilitate the standardized description and integration of module functionalities within the orchestration system.

To be able to access the capability and skill descriptions, an Asset Administration Shell (AAS) [34] implementation is advantageous. The CSS model outlines the use of an AAS submodel for describing capabilities, based on references to skills and additional metadata such as descriptions [2]. This capability submodel is currently under development [35]. To describe a skill within the AAS, it is recommended to use the AAS submodel element "operation" or a function block (FB), which represents an independent submodel [2]. In Section 3.3.1, it is proposed to implement the robot functionality with ROS2. However, currently there is no submodel to reference ROS2 functionalities within the AAS. Therefore, the development of a ROS2 compatible functionality description as depicted in Figure 5 is necessary.

A ROS-FB submodel, which represents the functionality provided by a URIP defined robot module, must entail the data presented in Table 1. Additionally, in order to be able to call the described functionality, a description of the skill interfaces, containing the data listed in Table 2, is also necessary. The detailed definition of the ROS-FB submodel, although within the scope of the author's research, is out of the scope of this article and will therefore be published in future work.

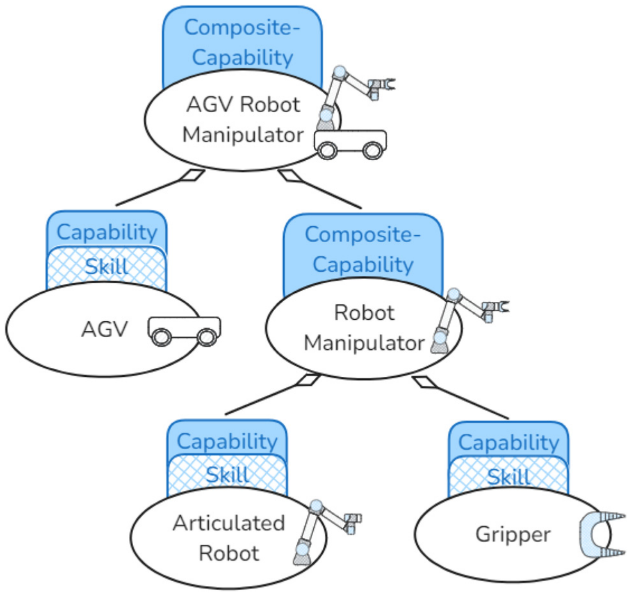

Figure 4: Capability and skill decomposition example.

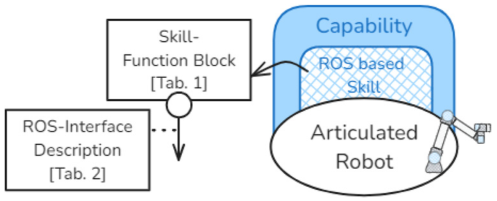
Skill-Function Block [Tab. 1]
Capability
ROS based Skill
ROS-Interface Description [Tab. 2]
Articulated Robot

Figure 5: Skill description of ROS functionalities.

Table 1: ROS function block submodel elements.

Element	Type	Example
Reference ID	ID	45dj-6qe3-sfo0-h23j
Name	String	Joint movement
Description	String	Joint based movement of the robot
Parameters	List<Ref>	Angle, velocity, acceleration
Prerequisites	List<String>	Robot not moving, angle within range
Post-requisite	List<String>	Robot not moving, angle within tolerance
Interface	Ref	k35h-kj36-s24f-1lk9

Table 2: ROS interface submodel elements.

Element	Type	Example
Reference ID	ID	k35h-kj36-s24f-1lk9
ROS-node name	String	JTC-controller
ROS-IP	String	196.0.0.1
ROS-domain	Int	0
ROS-method	Enum	Action
Topic name	String	JTC-controller/follow_joint_trajectory
Message type	String	control_msgs/action/FollowJointTrajectory.action

3.3.3 URIP-aspect 3: human machine interface aspect

The goal of this section is to explore the development and implementation of HMIs for modular robot systems. HMIs play a crucial role in ensuring that operators can interact and control robotic systems effectively, facilitating seamless operation and integration within flexible production environments.

One of the primary challenges in the development of HMIs for modular robot systems is the complexity that arises from the diverse functionalities and configurations of different modules. Each module may have unique control requirements, parameters, and procedures, making it difficult to create a standardized interface that can accommodate all variations. This complexity requires an advanced HMI that can adapt to the specific needs of each module.

To address this challenge and satisfy REQ 3, the integration of an automatically generated HMI, similar to the HMI of the MTP, is proposed. The proposed approach leverages the control implementation discussed in Section 3.3.1 and the descriptive aspect elaborated in Section 3.3.2. The automatic generation of HMIs is crucial for managing intricate interactions between modules, allowing operators to monitor and control systems with ease. This method minimizes the manual effort required to design and update HMIs, reducing the potential for errors and enhancing overall system efficiency.

The HMI visualizes the state of the robot module system, makes the module capabilities available to the operator, and incorporates the decomposition of the system. The HMI is provisioned for the overall robot module, each submodule and elementary component. This ensures exposure of all relevant functionalities for different scenarios like normal operation or maintenance, where granular control over specific components might be necessary.

In Figure 6, a mock-up HMI of the exemplary composite robot module from Section 3.2 is shown.

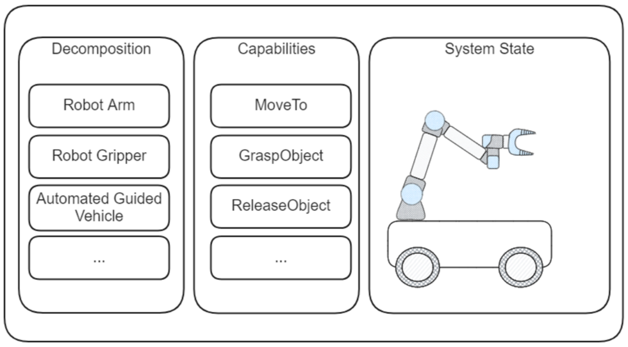

Figure 6: Exemplary mock-up HMI of AGV based robot manipulator module.

The following paragraphs explain the elements of a URIP HMI.

3.3.3.1 HMI element: system state

Utilizing the information models for robotic systems, which incorporate 3D models, the system state can be visualized for each robot (sub-)module and component directly. An example of such an information model would be the Simulation Description Format (SDF) [36] or the URDF mentioned in section Section 3.3.1. These information models can be used to render robotic components within a module. The renderings can be controlled to visualize the current system state of a component, for example, the joint angles of an articulated robot. Such visualizations can be implemented using ROS2 and the ROS-Visualization (RViz) [37]. In order to display the immediate robot environment, more elaborate tools like the robotics simulation tool Gazebo [38] can be used. This approach is well integrated with the ROS2 control aspect described in Section 3.3.1.

3.3.3.2 HMI element: capabilities

As described in Section 3.3.2 the advertised module functionality can be described with the CSS model. The elaborated approach can be used to advertise the appropriate module functionality in the HMI. Each capability can be displayed and interacted with regard to the description and the necessary parameters. Such module capabilities might be callable by an operator through the HMI, for example, to start and stop grasping functionality provided by an articulated robot with a gripper attachment. The description entails all relevant information, like the interface type, message type, namespace and endpoint in order to automatically create a client node. As described in Section 3.3.1, a node represents a ROS2 capable software program, that can communicate with other nodes, for example by calling services.

3.3.3.3 HMI element: submodules

As described in Section 3.2, a module consists of submodules and components. Each submodule provides a different granularity of the module functionality. In order to access the capabilities of a specific submodule, the HMI needs to provide a hierarchical structure. Therefore, each submodule and component need to be equipped with separate HMIs. A decomposition menu will make the composite system's HMI accessible. In this way, the HMI can penetrate all layers of a module and access data from all submodules, components, and the module itself.

3.4 Additional concept considerations

During the operation of a system, the aspects of safety and security are of utmost importance. To ensure safe and secure operation, it must be considered throughout the engineering lifecycle.

The safe operation of a robot system is provided predominantly through risk assessment processes. Risk assessment is performed according to the ISO 12100 [39] standards and the robot-specific consideration of *ISO 10218* [25]. These standards represent guidelines that enable the identification of safety issues in a robotic system and show how to elevate these issues through inherent, technical, or organizational measures. Additionally, the functional safety of the provided functionality must be assured by the system. The functional safety of machinery is standardized in the *IEC 62061* [40] and *IEC 61508* [41].

When handling complex multi-component production systems, a key aspect considering safety is the management of complex safety interconnections between subsystems. Based on the similarities in system architecture, this section refers to the research of [42]. Within this research, an interconnection model for safety in multi-component manufacturing systems is provided. This includes possible points of failure, organizational precautions, and implementation of active as well as passive system elements. This approach can be utilized to model the safety relations between components of a robot module, and can therefore in later stages be implemented into the URIP concept.

Additionally, in the context of a robot system in an Industrial Automation and Control System (IACS), security similarly to safety is also based on risk assessment considerations. The *de facto* standard here is the IEC 62443 [43]. However, its processes and requirements are not inherently conceived in the context of modularization [44]. For example, the underlying risk assessment requires experts from a wide range of fields, and reconfiguring the robot components would require repeating this complex process for the new environment. If reconfiguration requires a complete renewal of the assessment, the desired modularity of the system would be lost. Therefore, new methods, like automating the security risk assessment [45], need to be developed. For this, the interlinking and existing risk assessment approaches need to be adapted. In the context of the proposed implementation of ROS2 in Section 3.3.1, security is largely based on the security specification of the Data Distribution Service (DDS) [46]. The Secure ROS2 (SROS2) tool set provides a framework for enhancing security. SROS2 establishes the mentioned aspects of authentication, access control, and cryptography within the robot module. However, SROS2 also relies on the presence of certificates, which must be managed effectively within the modular system. Therefore, dedicated research, like [47] is needed with respect to the security management of modular systems.

In order to operate complex multi component robotic systems safely and securely, it is important to further develop methods of the automated and interconnected risk assessment. The solutions mentioned in this section mark the link to concurrent work presented by the authors which needs to be part of the overall URIP architecture presented in this paper.

The following section provides an exemplary application of the presented URIP in order to illustrate the feasibility and functionality.

4 Exemplary concept application

In this section, an exemplary robot module is presented. With this, the usage of the introduced URIP concept is outlined. The overall functionality of the exemplary robot module is the automated collaborative drilling of a work piece, as depicted in Figure 7.

Although drilling is not a natural task for collaborative systems due to the necessary system rigidity, the example is chosen because it is easy to explain and illustrate. The decomposition of the module is shown in Figure 8. The main components are two robots with specific end effectors, one end effector is a common gripper, and the other is a drill head. As illustrated in Figure 7, both robots are cooperating with each other in order to accomplish the drilling task.

Each elementary component, in the component-layer of Figure 8, implements an individual skill and exposes a corresponding capability. The drill attachment implements a ROS2 based controller to *activateDrill* and *deactivateDrill* the drill. Similarly, the gripper attachment implements the *closeGripper* and *openGripper* skills. Both robot components use the *ros2_control* implementation of a *joint_trajectory_controller*. Therefore, the robot movement is configured joint wise with angle, speed and acceleration parameters. In the submodule-layer of Figure 8 the granular skills of the composite components are orchestrated using

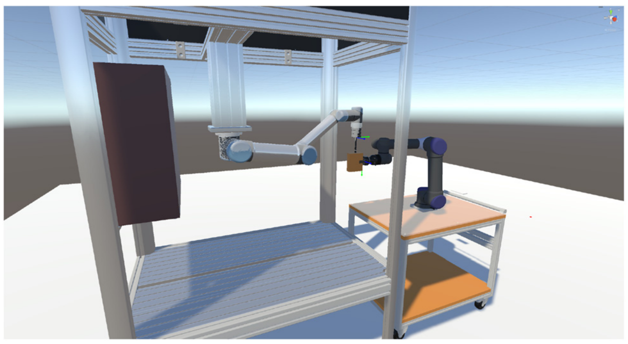

Figure 7: Simulation of the exemplary robot drilling module.

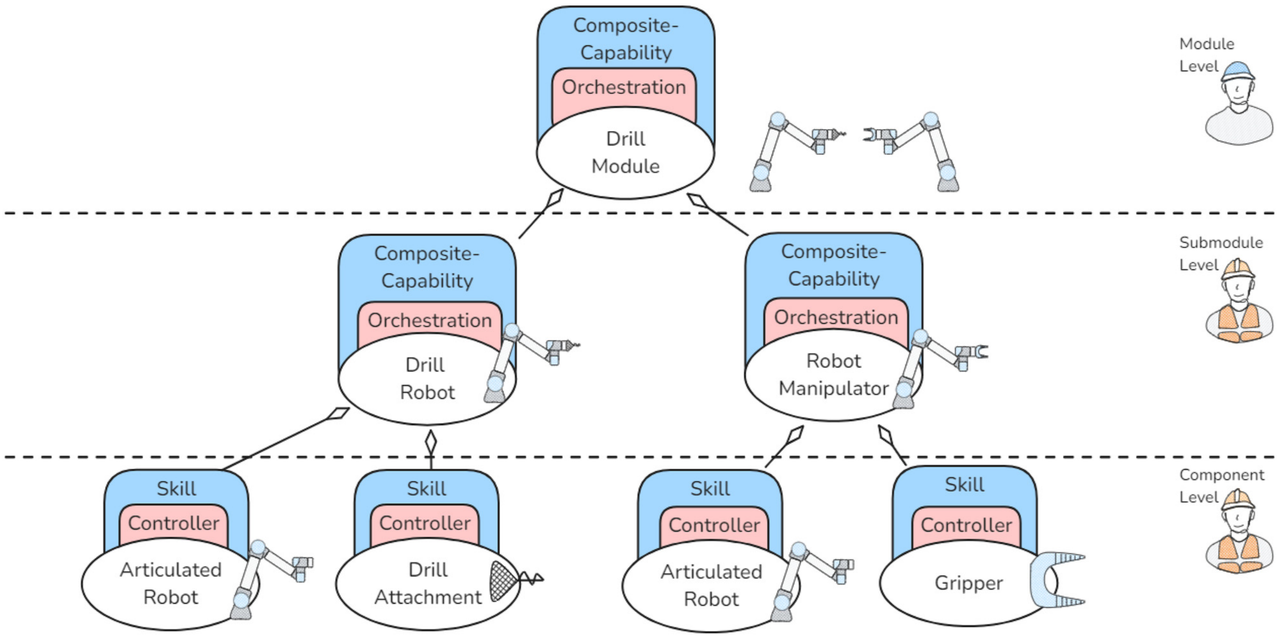

Figure 8: Example decomposition of a robotic drilling module.

the *SMACC2* implemented statecharts. The submodules provide the composite-capabilities *drilling*, *exposeWorkpiece*, *placeWorkpiece*. Within the state charts, each submodule or elementary component is represented as an orthogonal. In the module-layer the composite-capabilities are composed to the exposed capability of *automatedDrilling*, which uses the *exposeWorkpiece*, *placeWorkpiece* and *drilling* capabilities. The functionalities are listed in Table 3 Du to the taken approach, the integration involves little programming. The available elementary skills are provided as ROS2 services and actions. These can be orchestrated as composite capabilities with a SMACC2 state chart. Currently, the state chart integration is done manually, but an assistance based on the functionality description is possible in the future.

An imported point mentioned in Section 3 is the interchangeability of components. Due to the usage of the *ros2_control* implementation, it is possible to exchange the articulated robot with another robot with similar configuration, without any reprogramming. For example, the here used UR5 robot can be exchanged with a UR10 or a similar kuka iizy robot. All three systems are currently supported with open source repositories of the *ros2_control* stack.

Table 3: Functionalities of the drill module.

Name	Layer	Functionality
Articulated robot	Component	*jointMovement*
Drill attachment	Component	*activateDrill*, *deactivateDrill*
Gripper	Component	*openGripper*, *closeGripper*
Drill robot	Submodule	*drilling*
Robot manipulator	Submodule	*exposeWorkpiece*, *placeWorkpiece*
Drill module	Module	*automatedDrilling*

With regard to the security, *SROS2* is used for authentication and integrity within the operational- and submodule-layer. This is done to ensure that the signals and commands are sent by the appropriate system, and were not tampered with. The required device entities, including private key and public key inform of a X.509 certificate, trust anchor, as well as a security configuration for example in the form of a key storage, are provided manually. This is done for all components, participating in the ROS2 communication, this excluded vendor provided hardware controllers, which are currently not regarded within the security aspect.

All actuated submodules can be described within URDF or SDF information models, which can be used to visualize the state of the system. In this instance, the visualization depicted in Figure 7 is provided with a custom simulation environment implemented in Unity. The described HMI of Subsubsection 3.3.3 is not implemented as of this time.

This example illustrates how the URIP aspects can be used to combine elementary submodules to complex robot modules, with little integration effort.

5 Summary and outlook

In this article, the authors introduce the novel concept of the URIP. In this context, three main aspects: robot control, descriptive aspect, and HMI were introduced, and implementation recommendations were given. The control and orchestration of robot functionalities with the ROS2 control stack enables early validation with mock components and hardware abstracted implementation, as well as reuse of control algorithms. This is followed by the description of skills, capabilities and composite capabilities, enabling the future integration of URIP described robot modules into a ROS2 based RosOL. With the proposed HMI, only relevant functionalities of each module can be accessed during operation, but lower-level functionalities can be accessed for maintenance purposes. Lastly, an exemplary system was introduced to validate and explain the URIP concept.

In following publications, each aspect of the URIP will be elaborated in depth. Starting with the implementation of the control aspect in [33] and the detailed description of an information model for the implemented functionalities as proposed in Section 3.3.2. Afterward, the implementation of the HMI and the integration of URIP described robot modules into a RosOL will be explored.

Additionally, a seamless cross-phase usage of URIP from concept creation to design, implementation, testing, and operation will be transferred into a life-cycle model.

Research ethics: Not applicable.
Informed consent: Not applicable.
Author contributions: All authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Use of Large Language Models, AI and Machine Learning Tools: LanguageTool to detect spelling mistakes.
Conflict of interest: The author states no conflict of interest.
Research funding: None declared.
Data availability: Not applicable.

References

[1] J. Iqbal, M. Ul Islam, S. Abbas, A. Attayyab Khan, and S. Ajwad, "Automating industrial tasks through mechatronic systems — a review of robotics in industrial perspective," *Teh. Vjesn.*, vol. 23, no. 3, 2016. https://doi.org/10.17559/TV-20140724220401.
[2] C. Diedrich, *et al*., *Information Model for Capabilities, Skills & Services*, Berlin, Federal Ministry for Economic Affairs and Climate Action, 2022.
[3] Y. Hao, *et al*., "Universal soft pneumatic robotic gripper with variable effective length," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 6109—6114.
[4] E. Schäffer, L. N. Penczek, M. Bartelt, M. Brossog, B. Kuhlenkötter, and J. Franke, "A microservice- and AutomationML-based reference architecture for an engineering configurator web Platform," in *Procedia CIRP, 9th CIRP Global Web Conference — Sustainable, Resilient, and Agile Manufacturing and Service Operations: Lessons from COVID-19*, vol. 103, 2021, pp. 274—279.
[5] G. G. Rogers and L. Bottaci, "Modular production systems: a new manufacturing paradigm," *J. Intell. Manuf.*, vol. 8, no. 2, pp. 147—156, 1997. https://doi.org/10.1023/A:1018560922013.
[6] VDI/VDE/NAMUR 2658, *Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie*, Düsseldorf, Engl. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, 2022.
[7] L. Bittorf, J. Oeing, T. Kock, R. Garreis, and N. Kockmann, "Design of module type package services for modular downstream units and process analytic technology," *Chem. Eng. Technol.*, vol. 46, no. 7, pp. 1502—1510, 2023.
[8] G. Hildebrandt, P. Habiger, T. Greiner, and R. Drath, "Integrating robots in modular production environments via the module type package," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2024, pp. 1—7.
[9] V. Khaydarov, L. Neuendorf, T. Kock, N. Kockmann, and L. Urbas, "MTPPy: open-source AI-friendly modular automation | IEEE conference publication | IEEE xplore," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation ETFA*, 2022, pp. 1—7.
[10] M. Blumenstein, *et al*., "Designprinzipien für den Modul- und Serviceentwurf in modularen Logistikanlagen," in *Automation 2021*, VDI Verlag, 2021, pp. 101—116.
[11] M. Blumenstein, *et al*., "Logistics Orchestration Layer — Anforderungen an die Orchestrierung modularer Logistiksysteme," in *Automation 2023, ser. VDI-Berichte*, vol. 2419, VDI Verlag, 2023, pp. 55—72.
[12] M. Blumenstein, V. Henkel, A. Fay, A. Stutz, S. Scheuren, and N. Austermann, "Integration of flexible transport systems into modular production-related logistics areas," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1—8.
[13] OPC 10000-1: UA, *Part 1: Overview and Concepts2024-11-29*, USA, OPC Foundation, 2008.
[14] Z. Liu and P. Bellot, "OPC UA PubSub implementation and configuration," in *2019 6th International Conference on Systems and Informatics (ICSAI)*, 2019, pp. 1063—1068.
[15] M. Quigley, *et al*, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*. 3, 2009.
[16] A. Tripathy, J. Van Deventer, C. Paniagua, and J. Delsing, "Interoperability between ROS and OPC UA: a local cloud-based approach," in *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, Coventry, United Kingdom, IEEE, 2022, pp. 1—5.
[17] A. Cutting-Decelle, R. Young, J.-J. Michel, R. Grangel, J. Cardinal, and J.-P. Bourey, "ISO 15531 mandate: a product-process-resource based approach for managing modularity in production management," *Concurrent Eng.: R&A*, vol. 15, no. 2, pp. 217—235, 2007.
[18] J. Pfrommer, M. Schleipen, and J. Beyerer, "PPRS: production skills and their relation to product, process, and resource," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2013, pp. 1—4.
[19] A. Köcher, *et al*., "A reference model for common understanding of capabilities and skills in manufacturing," *at — Automatisierungstechnik*, vol. 71, no. 2, pp. 94—104, 2023.
[20] J. Pfrommer, D. Stogl, K. Aleksandrov, V. Schubert, and B. Hein, "Modelling and orchestration of service-based manufacturing

systems via skills," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–4.

[21] A. Bayha, J. Bock, B. Boss, C. Diedrich, and M. Somayeh, *Describing Capabilities of Industrie 4.0 Components: Joint White Paper between Plattform Industrie 4.0*, Berlin, Engl. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, 2020.

[22] A. Kocher, L. Beers, and A. Fay, "A mapping approach to convert MTPs into a capability and skill ontology," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Stuttgart, Germany, IEEE, 2022, pp. 1–8.

[23] V. 5100, *Systemarchitektur für die Intralogistik (SAIL)*, Düsseldorf, VDI-Gesellschaft-Produktion und Logistik, 2016.

[24] S. Yoo and A. Jerraya, "Introduction to hardware abstraction layers for SoC," in *Automation and Test in Europe Conference and Exhibition 2003 Design*, 2003, pp. 336–337.

[25] ISO 10218-1, *2011 Robots and Robotic Devices — Safety Requirements for Industrial Robots — Part 1: Robots*, Geneva, International Organization for Standardization, 2011.

[26] ISO 13482, *2014 Robots and Robotic Devices — Safety Requirements for Personal Care Robots*, Geneva, International Organization for Standardization, 2014.

[27] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: a decade's overview," *Inf. Sci.*, vol. 280, pp. 218–238, 2014.

[28] A. Stutz, A. Fay, M. Barth, and M. Maurmaier, "Orchestration vs. Choreography functional association for future automation systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8268–8275, 2020.

[29] D. Harel, "Statecharts: a visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, no. 3, pp. 231–274, 1987.

[30] S. Chitta, *et al*., "Ros_control: a generic and simple control framework for ROS," *J. Open Source Softw.*, vol. 2, p. 456, 2017.

[31] GitHub — Robosoft-ai/SMACC2, "An event-driven, asynchronous, behavioral state machine library for ROS2 (robotic operating system) applications written in C++." https://github.com/robosoft-ai/SMACC2 [accessed: Jan. 08, 2025].

[32] IEC 61512-1, *Batch control - Part 1: Models and terminology*, Geneva, Switzerland, International Electrotechnical Commission, 1997.

[33] L. Witucki and M. Barth, "Konzeptionelle Erweiterung von ROS2-basierten Robotersystemen für modulare Automatisierungsarchitekturen," in *Konferenzband zum VDI-Kongress AUTOMATION 2025, 26. VDI-Kongress AUTOMATION — Leitkongress der Mess- und Automatisierungstechnik (2025). Baden-Baden, Deutschland, 01.07.2025–02.07.2025*, 2025.

[34] *Specification of the Asset Administration Shell Part 1: Metamodel - IDTA, Number: 01001-3-0-1, Version: 3.12023. Industrial Digital Twin Association e.V. Lyoner Straße 1860528 Frankfurt am Main*, vol. 15, p. 2025, Jan. Available at: https://industrialdigitaltwin.org/en/content-hub/aasspecifications/specification-of-the-asset-administration-shell-part-1-metamodel-idta-number-01001-3-0-1 [accessed: Jan. 15, 2025].

[35] "Submodels — IDTA." https://industrialdigitaltwin.org/en/content-hub/submodels [accessed: Jan. 10, 2025].

[36] GitHub — gazebosim/sdformat, "Simulation Description Format (SDFormat) parser and description files." https://github.com/gazebosim/sdformat [accessed: Jan. 08, 2025].

[37] GitHub — ros2/rviz, "ROS 3D robot visualizer," 2025. https://github.com/ros2/rviz [accessed: Jan. 08, 2025].

[38] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154.

[39] ISO 12100:2010, *Safety of Machinery — General Principles for Design — Risk Assessment and Risk Reduction*, Geneva, International Organization for Standardization, 2010.

[40] IEC 62061:2021, *Safety of Machinery — Functional Safety of Safety-Related Control Systems*, Geneva, Switzerland, International Electrotechnical Commission, 2021.

[41] IEC 61508:2010, *Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-Related Systems*, Geneva, Switzerland, International Electrotechnical Commission, 2010.

[42] M. Stolze and M. Barth, "Semantische Verknüpfung von Informationsmodellen für Maschinen und Anlagen," in *Konferenzband zum VDI-Kongress AUTOMATION 2024, 25. VDI-Kongress AUTOMATION — Leitkongress der Mess- und Automatisierungstechnik (2024). Baden-Baden, Deutschland, 02.07.2024–03.07.2024*, 2024, p. 207.

[43] IEC 62443, *Industrial Communication Networks — Network and System Security Series*, Geneva, Switzerland, International Electrotechnical Commission, 2009/2020.

[44] M. Madsen, A. Palmin, A. Stutz, and M. Barth, "Security analysis of the module type package concept," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, IEEE, 2023, pp. 1–8.

[45] M. Ehrlich, A. Bröring, H. Trsek, J. Jasperneite, and C. Diedrich, "Evaluation concept for prototypical implementation towards automated security risk assessments," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–4.

[46] Dds-security, *Object Management Group*, Milford, Standard, 2018.

[47] M. Madsen, B. Geib, and M. Barth, "Enabling industrial security via certificate management concepts in the life cycle of a production gray-box," in *IECON 2024 — 50th Annual Conference of the IEEE Industrial Electronics Society: 03-06 November 2024, Chicago, (Chicago, IL, USA, Nov. 3–6, 2024)*, Institute of Electrical and Electronics Engineers (IEEE), 2024.

Bionotes

Linus Witucki
Institute of Control Systems, Karlsruher Institute of Technology, Kaiserstraße 12, D-76131 Karlsruhe, Germany
linus.witucki@kit.edu
https://orcid.org/0009-0003-4973-9471

Linus Witucki graduated from the Karlsruhe Institute of Technology (KIT) with a Master's degree in Electrical Engineering, where he established a strong expertise in control theory and robotics. He is currently working as a PhD student at the Institute of Control Systems (IRS) at KIT. The focus of his work is on the modular description of robotic systems and robotics in factory automation. In addition to his research, he teaches the practical application of control engineering in robotics at the master's level.

Marwin Madsen
Karlsruher Institut für Technologie (KIT) Institut für Regelungs- und Steuerungssysteme (IRS), Karlsruhe,Germany
https://orcid.org/0009-0006-9953-2382

After his graduation with a Master's degree in Computer Science from Karlsruhe Institute of Technology (KIT), Marwin Madsen specializes in the development of approaches to certificate management in modern, heterogeneous and flexible automation architectures. With a background in computer science as well as industrial automation, he has authored multiple international papers on automation security and teaches master-level courses targeting industrial security at the KIT. Currently he is pursuing his PhD at the Institute of Control Systems (IRS) at KIT.

Eric L. Wagemann
Karlsruher Institut für Technologie (KIT) Institut für Regelungs- und Steuerungssysteme (IRS), Karlsruhe,Germany
https://orcid.org/0009-0008-8420-644X

Eric L. Wagemann completed his Master's degree in Mechanical Engineering at Hamburg University of Technology (TUHH). During his graduate studies, he worked on the control design for electric drives in industrial robots. He is currently working as a PhD student at Karlsruhe Institute of Technology (KIT) under the guidance of Prof. Mike Barth. His focus is on information models and the simulation of electric drives. Additionally he is teaching practical robotics application at KIT.

Mike Barth
Karlsruhe Institute of Technology: Karlsruher Institut für Technologie, Institute for Control Systems, Fritz-Haber-Weg 1, 76131 Karlsruhe, Germany
https://orcid.org/0000-0003-2337-063X

Prof. Dr.-Ing. Mike Barth is Professor at the KIT - Institute of Control Systems. Holding the Chair for interconnected and secure automation technology, his research groups focus on the seamless use of information models and digital twins in the lifecycle of automation systems, new methods and architectures addressing cyber-physical robotic systems, as well as enhanced algorithms and processes in the field of IT/OT-Security. Besides various collaborations and projects with national and international research institutions and industry partners, he is chair of the IFAC Technical Committee 3.1 on "Computers for Control", advisory board member of the VDI/VDE association for Measurement and Automation Technology. In addition, he is working in the standardization committees of the Open Industry Alliance, PROFIBUS and PROFINET International (PI), IDTA and NAMUR.