

AI4U: Metadata-Driven Modular Design for Cyber-Physical Autonomous Systems

KAMIL WOŁOSZYN¹, KRZYSZTOF TURCHAN¹, KRZYSZTOF PIOTROWSKI¹ (Member, IEEE),
FATINA LILIANA BASMADJI², MAREK WĘGRZYN², AND MAREK SAŁAMAJ²

IHP – Leibniz-Institut für innovative Mikroelektronik, 15236 Frankfurt (Oder), Germany
CBK PAN – Centrum Badań Kosmicznych Polskiej Akademii Nauk, 66-002 Zielona Góra, Poland

CORRESPONDING AUTHOR: KAMIL WOŁOSZYN (e-mail: woloszyn@ihp-microelectronics.com).

This work was supported by the European Regional Development Fund within the INTERREG VI A BB-PL 2021–2027 Programme, project MobiRobAI, under Grant BBPL0100046, and in part by the German Federal Ministry of Research, Technology and Space (BMFTR), project AI-DISCO, under Grant 16ME1126K.

ABSTRACT Cyber-physical autonomous systems combine perception, decision-making, communication, and control to perform tasks with limited human involvement. Their design is still often manual and project-specific. This paper presents the current state of AI4U, an emerging metadata-driven methodology for the modular design of such systems. AI4U is being developed to describe hardware and software modules, support their composition, and check selected interface and deployment constraints. The methodology is discussed using MobiRobAI as the first full-scale application-driven development and validation case. MobiRobAI includes a ground mobile platform prepared for integration with a 7-DoF Kinova manipulator for greenhouse harvesting and a custom-built hexacopter UAV for photovoltaic panel inspection and servicing. Both demonstrators are used to refine the AI4U descriptor model and to test how perception, task planning, control, communication, and safety-related functions can be represented in one modular architecture. The paper clarifies the current scope of AI4U and introduces a conceptual perception-chain reconfiguration example to show how future module replacement can be represented as a localized and checkable operation.

INDEX TERMS Autonomous systems, autonomy, cyber-physical systems, modular design, design methodology, metadata, edge AI, agricultural robotics.

I. INTRODUCTION

AUTONOMOUS systems are designed to perform tasks with limited human intervention. They perceive the environment, make decisions, communicate with other system parts, and control physical actions. For this reason, an autonomous robot or UAV can be treated as a cyber-physical autonomous system (CPAS). Such a system combines computational elements, such as AI models, planners, and control algorithms, with physical components, such as sensors, actuators, mobile platforms, and manipulators. It also requires communication links for data exchange, supervision, and coordination. CPAS solutions are increasingly used in agriculture, energy infrastructure, transportation, and environmental monitoring.

Although these systems reduce human involvement during operation, their design is still often manual and expert-driven. Engineers have to translate user goals into technical requirements, select sensors and actuators, choose AI models

and computing platforms, connect software and hardware interfaces, and verify the complete configuration before deployment. In many projects, these steps are repeated from the beginning for each new application. Even a small change, such as replacing a camera, moving an AI model to another computing platform, or adding a new task to the same robot, can require changes in several parts of the implementation. This is especially important for autonomous systems, because integration errors may appear later during field operation, where human intervention is limited.

Existing tools support only selected parts of this process. Some tools help train and deploy AI models. Others standardize communication between running software components or allow users to build data flows graphically. However, these tools usually do not cover the complete design path from a user-level task to a validated autonomous system. What is needed is a design methodology that supports component

selection, interface checking, system composition, and reconfiguration before deployment.

This paper presents the current state of AI4U, an emerging metadata-driven design methodology for modular cyber-physical autonomous systems. The main idea is to describe hardware and software modules by machine-readable metadata and to use these descriptions during system design. At the current stage, AI4U should not be understood as a fully automatic architecture synthesis tool. The integrator still translates user goals into technical requirements, selects or approves the system structure, and makes final engineering decisions. The AI4U environment is being developed to support this work by storing module descriptors, checking selected input/output relations, identifying missing mandatory connections, and verifying selected platform and timing constraints.

The contribution of this paper is fourfold. First, we describe the current AI4U design process, including four roles and six stages, and clarify which parts are performed by the integrator and which parts are intended to be supported by the AI4U environment (Section III). Second, we present the current metadata model for describing modules and applications, including common descriptor fields and planned type-specific extensions for sensors, actuators, AI models, logic modules, controllers, and communication links (Section IV). Third, we introduce an illustrative perception-chain example from MobiRobAI to show how metadata can describe module interfaces, how basic compatibility checks are planned to be performed, and how a future reconfiguration test can be represented. Fourth, we discuss the use of AI4U in the MobiRobAI project, which is being used as the first full-scale application-driven case for developing and testing the methodology. MobiRobAI includes two demonstrators: a ground mobile platform with a 7-DoF Kinova manipulator, prepared for greenhouse harvesting tests, and a custom-built hexacopter UAV for photovoltaic (PV) panel inspection and servicing. Both demonstrators are used to refine the AI4U descriptor model and to test how perception, task planning, control, communication, and safety-related functions can be represented within a modular autonomy architecture (Section V). Section VI discusses the current limits of the approach and its possible transfer to distributed measurement systems, while Section VII concludes the paper.

The methodology builds on earlier work on modular application engineering. It started with the Sens4U approach for wireless sensor networks [1] and was later extended through the AI4U framework concept [2], [3], its application to resource-constrained edge devices [4], the automation-oriented AMAD concept [5], an analysis of challenges in modular AI engineering [6], the structured AI4U design process [7], and metadata-based data interfaces [8]. In this paper, these earlier elements are being consolidated and adapted to the design of cyber-physical autonomous systems. MobiRobAI is used as the first physical testbed for this development and validation process.

II. RELATED WORK

Work related to AI4U can be grouped into four areas: robotics frameworks and development tools, model-driven and model-based engineering for cyber-physical systems, interface and semantic information models, and modular or constraint-based composition methods.

Robotics frameworks and development tools: The Robot Operating System 2 (ROS 2) [9] is widely used as an integration layer for research robots. It provides typed topics, services, actions, and interface files that support communication between software components. This makes ROS 2 very useful during implementation. However, ROS 2 does not solve the earlier design problem. It does not decide which camera, AI model, controller, or computing platform should be selected for a given task. It also does not check, before implementation, whether the selected components form a complete system that satisfies timing, resource, and interface constraints.

Flow-based tools, such as Node-RED [10], make it easy to connect data processing blocks, but they usually do not include detailed knowledge about hardware limits, latency, energy use, or AI model performance. Other tools focus on selected stages of development. Edge Impulse [11] supports data collection, model training, and deployment on edge devices. MOV.AI [12] provides tools for managing robotic applications on top of ROS. AutoML services [13] automate parts of model development. These tools are useful, but they do not cover the full path from a user-level autonomy task to a checked cyber-physical system configuration.

Model-driven and model-based engineering for CPS: Model-driven engineering and model-based systems engineering address similar integration problems at a higher design level. A systematic mapping study shows that model-driven engineering has been used in CPS development to model heterogeneous software and hardware parts, support analysis, and manage system complexity [14]. Other work combines model-driven engineering with semantic technologies to support reuse and composition of cyber-physical modules described by hardware and software properties [15]. AI4U follows the same general motivation of moving integration decisions before detailed implementation, but it is intended as a lighter metadata-based methodology focused on AI-based autonomous systems and early checks performed together with an integrator.

Interface and semantic information models: Well-defined interfaces are also central in industrial automation and IoT systems. IEC 61499 [16] describes distributed automation applications as function blocks with typed inputs and outputs. OPC UA [17] provides a standardized information model for industrial systems. Semantic models such as SAREF [18] and SSN/SOSA [19] provide shared descriptions for devices, functions, observations, sensors, actuators, platforms, and procedures. AI4U uses a related idea, but applies it earlier in the design process. In contrast to sensor-oriented semantic models, AI4U also describes AI models, task logic,

controllers, communication modules, deployment platforms, selected constraints, and performance fields.

Modular and constraint-based composition methods: AI4U also relates to modular composition and configuration approaches. General knowledge-based and constraint-based configuration methods use component properties and constraints to compose valid system or product variants [20]. These methods provide useful background for the planned AI4U matching and checking functions, although they are not specific to robotic CPS or AI-based autonomy. Earlier Sens4U work showed that wireless sensor network applications can be built from reusable library components with defined interfaces, supported by the tinyDSM middleware [1], [21]. The AMAD concept extended this idea by adding automated, constraint-based sensor selection from metadata, including measurement type, current draw, and communication interface [5].

AI4U is therefore positioned between these areas rather than as a replacement for them. It combines ideas from interface descriptions, semantic metadata, and constraint-based configuration, but applies them to autonomy-oriented chains such as perception, task logic, control, supervision, and safety reactions. At the current stage, the complete AI4U design environment is still under development, and the MobiRobAI case is used to refine the descriptors and planned checks in a physical robotic system.

III. THE AI4U DESIGN METHODOLOGY

AI4U is based on a simple idea: an autonomous application should be built from reusable modules with clear interface descriptions. Each module has declared inputs, outputs, constraints, and performance parameters. A complete application is then created by selecting and connecting modules that satisfy the requirements of a given task. This makes the design process more repeatable and makes later changes easier, for example when a sensor, an AI model, or a computing platform has to be replaced.

The methodology separates two aspects of the system. The first one is what the system should do, for example perceive an object, plan a task, move a robot, or report a fault. The second one is how this function is implemented by concrete hardware and software modules. This separation is important for cyber-physical autonomous systems, because autonomy depends on many connected parts: perception, decision-making, control, communication, and safety mechanisms.

A. SCOPE OF AUTOMATION

AI4U is currently being developed as a methodology and as a supporting design environment. It is not yet a fully automatic system generator. The integrator remains responsible for interpreting user goals, defining technical requirements, approving the architecture, and making final engineering decisions.

The design environment is being implemented to support selected parts of this process. The first planned and partially implemented checks focus on module descriptors, declared inputs and outputs, missing mandatory connections, selected platform constraints, and simple latency-budget estimates.

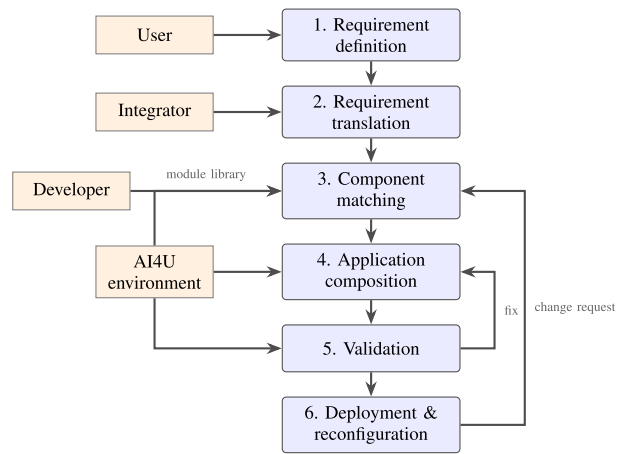


FIGURE 1. AI4U design flow with six stages, main roles, and feedback paths for correction and reconfiguration.

Full automatic architecture synthesis from natural-language requirements, semantic matching of module meanings, automatic insertion of adapter modules, and formal safety verification are not yet implemented. These functions are treated as future work and will be refined during the MobiRobAI project validation.

B. ROLES

Four roles are used in the AI4U process. In small projects, one person may take more than one role, but the separation helps to keep the responsibilities clear.

- **User** – describes, in plain language, what the system should do, where it should operate, and what constraints should be considered. Examples include battery operation, outdoor use, limited space, or safe distance from people.
- **Integrator** – translates the user goals into technical requirements. These requirements may include sensing methods, AI tasks, accuracy, latency, computing resources, energy limits, communication protocols, and safety constraints.
- **Developer** – provides hardware or software modules for the shared module library. Each module is delivered together with the metadata described in Section IV.
- **AI4U environment** – supports the design process. In the current development stage, it is being implemented to store module descriptors and perform selected checks, such as input/output compatibility, missing mandatory inputs, and selected platform constraints. Module ranking, deployment artifact generation, semantic matching, and complete validation are planned as further functions.

C. DESIGN STAGES

The AI4U process consists of six stages, shown in Fig. 1.

- 1) *Requirement definition:* The user describes the task and the operating scenario. For example, the task may be to detect ripe tomatoes in a greenhouse and detach them from the plant. At this stage, the user also defines environmental and physical constraints.

- 2) *Requirement translation*: The integrator converts the user description into measurable technical requirements. This may include required sensors, AI functions, control functions, latency limits, computing platforms, energy limits, and communication interfaces. Scenario templates, such as a greenhouse robot or a PV-farm inspection UAV, can be used as a starting point.
- 3) *Component matching*: The AI4U environment is intended to search the module library using metadata fields such as module type, interface, constraint, performance parameter, and domain tag. In the current implementation stage, this step is only partially supported and still requires integrator review and approval of selected modules.
- 4) *Application composition*: The selected modules are connected by linking compatible outputs and inputs. The application can be described on three levels. The first level is a dataflow graph, which shows how information moves between modules. The second level is a behavior model, for example a finite state machine (FSM), which describes operating states and reactions to events such as missing data, low battery, or execution faults. The third level is a YAML or JSON configuration file that lists modules, parameters, and links.
- 5) *Validation*: Before deployment, the AI4U environment is intended to perform selected consistency checks on the composed application. In the current implementation, the checks are limited to basic interface compatibility, mandatory input connections, selected platform constraints, and simple latency-budget estimates. Stronger checks, such as semantic compatibility, timing guarantees, and safety properties, remain future work.
- 6) *Deployment and reconfiguration*: After validation, the design should provide the basis for configuration files, installation scripts, model parameters, and wiring or calibration instructions. Later changes, such as replacing a camera or updating an AI model, are expected to be handled by updating the affected metadata descriptors and running the available checks again. In this paper, this is treated as a design goal and is illustrated by a planned perception-chain reconfiguration test. Quantitative confirmation will be part of the MobiRobAI project validation.

The expected advantage of this process is that the application logic can be separated from the concrete implementation. A module should be replaceable when its metadata contract remains compatible with the rest of the system. At the current stage, this is a methodological assumption supported by descriptor examples and planned validation tests, not yet a quantitatively proven result.

IV. METADATA-BASED SPECIFICATION AND COMPOSITION

In the intended AI4U workflow, modules are not connected only by ad-hoc code changes. Each connection should be

TABLE 1. Planned AI4U Metadata Fields and Type-Specific Extensions

Module type	Common fields	Type-specific fields
Sensor	id, type, inputs, outputs, constraints, tags	measured quantity, unit, frame, sampling rate, resolution, interface, calibration
AI model	id, type, inputs, outputs, constraints, tags	task, model class, metric, dataset, input size, quantization, latency condition, target platform
Logic / FSM	id, type, inputs, outputs, constraints, tags	states, events, transition conditions, timeout, safety reactions
Controller	id, type, inputs, outputs, constraints, tags	control rate, controlled axes, real-time requirement, platform, safety limits
Actuator	id, type, inputs, outputs, constraints, tags	command type, range, unit, feedback, update rate, physical limits
Communication	id, type, inputs, outputs, constraints, tags	protocol, bandwidth, latency, reliability, range, security option

based on a machine-readable description of the modules involved. This description defines what a module needs, what it provides, and under which conditions it can be used.

The AI4U descriptor model is still being refined. It uses a common top-level structure, but it does not assume that all module types have identical metadata fields. All modules share fields such as *id*, *type*, *inputs*, *outputs*, *constraints*, and *tags*. Type-specific extensions are then added for different classes of modules. For example, a camera descriptor may include resolution, frame rate, image format, calibration, interface, and coordinate frame. An AI model descriptor may include task type, model class, input size, metric, dataset, quantization, target platform, and latency measurement condition. These extensions are being implemented and tested in MobiRobAI to determine which fields are sufficient and which have to be refined. The planned common metadata fields and type-specific extensions are summarized in Table I.

A. MODULE DESCRIPTION

Each module is described in YAML or JSON. The description contains a unique *id*, a *type*, typed *inputs* and *outputs*, *constraints*, performance parameters, and optional domain *tags*. The module type can describe, for example, a sensor, actuator, AI model, logic module, communication module, or storage module. Constraints may include the hardware platform, resource use, timing limits, energy use, or required communication interface.

Listing 1 is an illustrative descriptor example, not a report of experimental results. The fields under *performance* show where metric values and their measurement conditions can be stored. The final set of performance and latency fields is still being refined and will be evaluated during the MobiRobAI project validation.

Two modules can be connected when the output of one module matches the required input of another module and when their constraints can be satisfied on the target platform. These rules can detect common integration problems, such as missing mandatory inputs, type mismatches, unsupported platforms, or violated latency and resource limits.

Listing 1: Illustrative descriptor of an object-detection component.

```

1 id: tomato_detector_v1
2 type: ai_model
3 task: object_detection
4 inputs: [image_rgb]
5 outputs: [bbox, confidence, ripeness]
6 constraints:
7   platform: example_edge_platform
8 performance:
9   metric: example_metric
10  value: example_value
11  condition: example_condition
12 tags: [greenhouse, fruit, detection]

```

B. APPLICATION DESCRIPTOR

In the intended workflow, a complete application is described as a set of connected modules. The application descriptor lists the selected modules, their parameters, and their links. For example, an autonomy chain can be described as *camera* → *detector* → *task FSM* → *actuator driver*. In the current implementation stage, this descriptor is used mainly to prepare selected checks, such as input/output compatibility, missing mandatory inputs, selected platform constraints, and simple latency-budget estimation.

The matching step is planned to become more automated following the AMAD concept [5]. If the requirements are measurable, the design environment should be able to rank candidate modules according to how well they satisfy the constraints. For example, for a requirement such as “measure temperature with a sensor drawing at most 10 mA”, the environment should be able to search the library, reject incompatible sensors, and propose suitable candidates. In the current AI4U implementation, this ranking is not yet fully available and still requires integrator review.

Ontology support is planned as a further extension. For example, domain tags could be aligned with SAREF concepts to make matching semantic rather than only syntactic [5], [8]. This could help the system interpret different names or descriptions that refer to the same type of device, function, or observation.

C. CONCEPTUAL PERCEPTION-CHAIN RECONFIGURATION EXAMPLE

To make the planned reconfiguration mechanism more concrete, we define a conceptual example from the MobiRobAI ground demonstrator. The example considers the perception chain *RGB camera* → *tomato detector* → *harvesting FSM*. In the intended descriptor set, the camera module provides *image_rgb*, the detector consumes *image_rgb* and provides *bbox*, *confidence*, and *ripeness*, and the harvesting FSM consumes these outputs to select a target fruit. At the current stage, this example is not reported as an executed experiment. It is used to specify the descriptor structure and the expected validation behavior that are being implemented in the AI4U environment.



FIGURE 2. MobiRobAI ground demonstrator prepared for integration with autonomy modules.

The planned reconfiguration scenario replaces the detector descriptor with a version that also provides distance, but requires an additional *image_depth* input. With the original RGB camera descriptor, the expected validation result is a missing-input report, because no selected module provides *image_depth*. The inconsistency could be resolved by selecting an RGB-D camera descriptor or by adding a separate distance-estimation module. The purpose of this example is not to compare detector performance. It is to define a concrete module-replacement scenario that will be used to test whether AI4U can make replacement a localized and checkable operation.

V. MOBIROBAI AS A CYBER-PHYSICAL AUTONOMOUS SYSTEM

MobiRobAI is used as a case study for applying AI4U to a real cyber-physical autonomous system. The project includes two demonstrators based on one modular design approach: a ground mobile platform is shown in Fig. 2 with a 7-DoF Kinova manipulator for greenhouse harvesting, and a custom-built hexacopter UAV for photovoltaic (PV) panel inspection and servicing. Both demonstrators combine computation, physical components, and communication links. They also share the same design goal: the operator should give only a short task command, while the system should handle perception, task planning, control, monitoring, and fault reaction as far as possible on its own.

In this paper, the MobiRobAI project focuses not only on building a robotic platform, but also on developing a cyber-physical autonomous system (CPAS). Autonomy is not implemented in a single module. Instead, it is achieved through the cooperation of many modules, including sensors, AI perception models, task logic, control algorithms, actuators, communication links, and safety mechanisms. AI4U is currently being introduced to describe selected modules using metadata and to prepare consistency checks for their integration and future replacement. The complete validation of this process is planned as part of the MobiRobAI project.

A. AUTONOMY-ORIENTED MODULAR ARCHITECTURE

Both demonstrators follow a hierarchical architecture with three main layers: supervision, control, and perception/hardware.

- **Supervision layer** – this layer represents the main autonomy logic. It receives short operator commands, such as a harvesting task in a greenhouse row or an inspection task over PV panels. It divides the task into smaller steps, supervises execution, receives status information from lower layers, and reacts to events such as low battery, missing data, a blocked path, or loss of communication. In AI4U, this layer is described mainly by logic modules, task planners, and finite state machines (FSMs).
- **Control layer** – this layer is responsible for trajectory generation and motion control. In the ground demonstrator, the control part includes FPGA-accelerated real-time algorithms for the mobile platform and manipulator. In the aerial demonstrator, the flight control is based on an ArduPilot-compatible autopilot module. In AI4U, these elements are described as control and logic modules with timing and platform constraints.
- **Perception and hardware layer** – this layer contains sensors, edge AI models, actuators, and platform-specific hardware. For the ground robot, this includes cameras, encoders, the mobile base, the Kinova arm, and the end effector. For the UAV, this includes RTK GNSS, optical flow, HD and thermal imaging, the hexacopter drive system, and the payload interface. In AI4U, these elements are described as sensor, AI model, actuator, and communication modules.

B. GROUND DEMONSTRATOR: MOBILE ROBOT WITH A 7-DOF MANIPULATOR

The ground demonstrator is designed for production greenhouses, where the robot must operate in narrow aisles, carry harvested products, and move both on a concrete floor and on a pipe-rail track. The mobile platform is an in-house mechanical design with a welded steel frame, two independently controlled DC gear motors, wheel encoders, a guiding front axle, batteries, electronics, crate space, and a modular height-adjustable base for the manipulator.

Manipulation is planned with a commercial 7-DoF Kinova arm and an exchangeable end effector. For harvesting, the end effector can combine gripping and cutting functions. The target perception setup uses an RGB-D camera close to the end effector and an AI module intended to detect tomatoes, classify ripeness, support tracking, and estimate target distance. In the AI4U description, these elements are represented as sensor, AI model, logic, and control modules.

C. AERIAL DEMONSTRATOR: CUSTOM-BUILT INSPECTION UAV

The aerial demonstrator is a custom-built hexacopter for PV panel inspection and servicing. The hexacopter layout was

TABLE 2. Mapping of MobiRobAI Components to CPAS Functions and AI4U Metadata Types

CPAS function	Ground robot	UAV	AI4U type
Perception	RGB camera, tomato detector, visual markers	HD/thermal camera, PV defect detection	sensor, ai_model
Localization	Encoders, visual pose estimation	RTK GNSS, optical flow	sensor
Decision-making	Harvesting task FSM, fruit selection	Inspection and servicing FSM	logic
Control	Platform and manipulator control, FPGA acceleration	ArduPilot flight control	logic, actuator
Physical action	Mobile base, Kinova arm, gripper/cutter	Hexacopter drive, cleaning payload	actuator
Communication	Operator commands, robot status, alerts	HD link, telemetry, return-to-home trigger	comm.
Safety	Obstacle reaction, low-battery handling	Loss-of-link handling, motor-out redundancy	logic, constraint

selected for redundancy and safety, because it provides better fault tolerance than a quadcopter. The platform uses a carbon-fiber Hexa X frame, six BLDC drive units with 17-inch propellers, and a payload interface for a cleaning module of up to 4 kg. The full take-off mass is about 11.5 kg, and the selected drive setup gives a thrust-to-weight ratio of about 2.35.

The avionics are based on an ArduPilot-compatible Cube Orange+ flight controller. The UAV includes RTK GNSS, optical flow for landing support, an HD/thermal camera, an HD video and control link, and an independent 868 MHz telemetry channel. The planned autonomous mission is represented as a task FSM with waypoint-based inspection, image and thermal data acquisition, return-to-home behavior, and later servicing with the cleaning payload.

D. MAPPING MOBIROBAI TO CPAS AND AI4U

Table 2 shows how the main parts of MobiRobAI map to the CPAS view and to AI4U metadata types. This mapping is important because it connects the practical robotic system with the design methodology. The same metadata model is used for commercial components, in-house mechanical subsystems, AI models, control logic, and communication links.

E. METADATA-GOVERNED INTERFACES AND RECONFIGURATION

In the AI4U view, connections between subsystems are represented as metadata contracts. For example, a perception module may provide outputs such as `bbox`, `confidence`, `ripeness`, or `distance`. These outputs can be consumed by a task FSM or a control module if the required types and selected constraints match. The supervision layer can also exchange typed task and status messages with the control layer, while the operator link can be described by a limited set of commands, reports, and alerts.

For MobiRobAI, this idea is currently being tested on selected subsystem chains rather than on the fully integrated autonomous system. The greenhouse harvesting task and the PV inspection task use different sensors, actuators, payloads,

and AI models, but both can be described using the same metadata principles. The expected benefit is that replacing a selected module, such as a camera, detector, payload, or end effector, should require changes mainly in the affected descriptors and links. This benefit still has to be evaluated experimentally.

The same principle is also reflected in the physical design. The ground platform uses an exchangeable end-effector concept and a modular arm base, while the UAV uses a quick-mount payload interface. These mechanical interfaces support future reconfiguration tests, but their effect on integration effort has not yet been quantified.

F. VALIDATION PLAN

The current validation status differs between the two demonstrators. The ground mobile platform has been built and structurally verified as a basis for further integration with autonomy, perception, and manipulation functions. The UAV prototype has been built and configured, and the drive system and avionics have been prepared for functional flight tests.

The next validation steps will focus on both platform autonomy and the AI4U design approach. For the ground robot, planned tests include operation in laboratory rows of tomato plants and later tests in greenhouse conditions. The evaluated scenarios will include fruit detection, target selection, manipulator motion, and operator supervision limited to high-level commands. For the UAV, planned tests include PV panel inspection, image and thermal data collection, return-to-home behavior, and later servicing with the cleaning payload.

The AI4U methodology will be evaluated in parallel. The first evaluation will check whether the descriptors are sufficient to represent selected module inputs, outputs, constraints, and performance conditions. Further tests will check whether the implemented validation rules can detect typical integration problems, such as missing mandatory inputs, incompatible data types, unsupported target platforms, and violated simple latency budgets. Reconfiguration tests will focus on selected subsystem chains, starting with the perception chain described in Section IV.

Planned quantitative metrics include task completion time, number of operator interventions, module replacement effort, number of descriptor changes required during reconfiguration, and the number of integration errors detected before field testing. These metrics will be used to assess the practical usefulness of AI4U and to identify which parts of the methodology need further refinement.

VI. DISCUSSION AND TRANSFER TO DISTRIBUTED MEASUREMENT SYSTEMS

The AI4U approach is used in this letter for robotic cyber-physical autonomous systems, but the same descriptor idea can also be applied to distributed measurement systems. IoT and wireless sensor network applications are also built from connected modules, such as sensing nodes, communication links, processing blocks, AI models, and storage

services. The MobiRobAI autonomy chain *camera–AI model–task logic–control* can therefore be mapped to a measurement chain such as *sensor node–communication–processing–AI analysis–storage* [8]. In this domain, descriptors would need additional fields for communication quality, sampling rate, latency, energy use, local processing capability, and semantic data description using concepts such as SAREF [18].

The main limitation is that AI4U is still under development. The methodology and descriptor model are defined, and first descriptors are being prepared and tested in MobiRobAI, but the complete design environment is only partially implemented. Current checks focus mainly on syntactic compatibility and selected declared constraints. Semantic compatibility, automatic adapter insertion, formal safety verification, and full timing guarantees remain future work. The MobiRobAI validation campaign will be used to refine the metadata fields and validation rules.

VII. CONCLUSION

This paper presented the current state of AI4U, an emerging metadata-driven methodology for modular cyber-physical autonomous systems. The methodology separates what the system should do from how it is implemented and represents sensors, actuators, AI models, control modules, logic modules, and communication links using machine-readable descriptors.

The paper clarified that AI4U is not yet a fully automatic architecture synthesis tool. At the current stage, the integrator remains responsible for requirement interpretation, architecture approval, and final engineering decisions. The AI4U environment is being implemented to support selected parts of the process, including descriptor-based representation, input/output compatibility checking, missing-input detection, selected platform constraints, and simple latency-budget estimation.

MobiRobAI was discussed as the first full-scale application-driven case for developing and testing the approach. The ground mobile platform and the hexacopter UAV show how different cyber-physical autonomous platforms can be mapped to a common metadata model. A perception-chain example was introduced to show how a future module replacement test can be represented as a localized and checkable operation. Future work will focus on completing the AI4U environment, extending semantic and safety checks, and quantitatively evaluating autonomy-related integration and reconfiguration effort during the MobiRobAI project validation.

ACKNOWLEDGMENT

The funding institutions had no role in the design of the study; the collection, analysis, or interpretation of data; the writing of the manuscript; or the decision to publish the results.

REFERENCES

- [1] K. Piotrowski and S. Peter, “Sens4U: Wireless sensor network applications for environment monitoring made easy,” in *Proc. 4th Int. Workshop Softw. Eng. Sensor Netw. Appl.*, 2013, pp. 37–42.

- [2] K. Wołoszyn, K. Turchan, M. R/kała, and K. Piotrowski, "AI4U: Modular framework for AI application design," in *Proc. Conf. Multimedia, Interaction, Des. Innov. (MIDI)*, 2022, pp. 56–66.
- [3] K. Wołoszyn, E. Michta, and K. Piotrowski, "Modular AI applications following the Sens4U approach," in *Proc. Sci. Conf. Sensors Meas. Syst. (SP)*, 2022, pp. 1–4.
- [4] K. Wołoszyn and K. Piotrowski, "Concept for building edge devices application using the AI4U approach," in *Proc. 20th GI/ITG KuVS Fachgespräch Sensornetze (FGSN)*, 2023, pp. 1–2.
- [5] K. Wołoszyn and K. Piotrowski, "Automated modular application development," in *Proc. Sci. Conf. Sensors Meas. Syst. (SP)*, 2024, pp. 1–4.
- [6] K. Wołoszyn, K. Piotrowski, and K. Turchan, "Challenges in developing modular AI applications," in *Proc. 21st GI/ITG KuVS Fachgespräch Sensornetze (FGSN)*, 2024, pp. 1–3.
- [7] K. Wołoszyn, K. Piotrowski, and K. Turchan, "AI4U: Structured design process for modular AI applications," in *Proc. 22nd GI/ITG KuVS Fachgespräch Sensornetze (FGSN)*, 2025, pp. 1–4, doi: [10.21268/20250808-1](https://doi.org/10.21268/20250808-1).
- [8] K. Wołoszyn and K. Piotrowski, "AI4U: Data interfaces for modular AI applications," in *Proc. Sci. Conf. Sensors Meas. Syst. (SP)*, 2026, pp. 56–66.
- [9] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, 2022, Art. no. eabm6074.
- [10] Node-RED: Low-code programming for event-driven applications, Accessed: Jul. 15, 2026. [Online]. Available: <https://nodered.org/>
- [11] Edge Impulse, Accessed: Jul. 15, 2026. [Online]. Available: <https://www.edgeimpulse.com/>
- [12] MOV.AI Robotics engine platform, Accessed: Jul. 15, 2026. [Online]. Available: <https://www.mov.ai/>
- [13] Google Cloud Vertex AI AutoML, Accessed: Jul. 15, 2026. [Online]. Available: <https://cloud.google.com/vertex-ai>
- [14] M. A. Mohamed, M. Challenger, and G. Kardas, "Applications of model-driven engineering in cyber-physical systems: A systematic mapping study," *J. Comput. Lang.*, vol. 59, 2020, Art. no. 100972, doi: [10.1016/j.cola.2020.100972](https://doi.org/10.1016/j.cola.2020.100972).
- [15] S. Feldmann, S. Rösch, D. Schütz, and B. Vogel-Heuser, "Model-driven engineering and semantic technologies for the design of cyber-physical systems," *IFAC Proc. Vol.*, vol. 46, no. 7, pp. 210–215, 2013, doi: [10.3182/20130522-3-BR-4036.00050](https://doi.org/10.3182/20130522-3-BR-4036.00050).
- [16] Function Blocks—Part 1: Architecture, International Electrotechnical Commission, Standard IEC 61499-1, 2012.
- [17] OPC Unified Architecture—Part 1: Overview and Concepts, International Electrotechnical Commission, Standard IEC 62541-1, 2020.
- [18] ETSI, "SAREF: The smart applications REference ontology," European Telecommunications Standards Institute, Tech. Specifications 103 264, 2025. Accessed: Jul. 15, 2026. [Online]. Available: <https://saref.etsi.org/>
- [19] A. Haller, K. Janowicz, S. J. D. Cox, D. Le Phuoc, K. Taylor, and M. Lefrançois, *Semantic Sensor Network Ontology*, W3C Recommendation, Oct. 19, 2017. [Online]. Available: <https://www.w3.org/TR/vocab-ssn/>
- [20] A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, Eds., *Knowledge-Based Configuration: From Research to Business Cases*. Waltham, MA, USA: Morgan Kaufmann/Elsevier, 2014.
- [21] K. Piotrowski, P. Langendörfer, and S. Peter, "TinyDSM: A highly reliable cooperative data storage for wireless sensor networks," in *Proc. IEEE Int. Symp. Collaborative Technol. Syst.*, 2009, pp. 225–232.
- [22] C. W. Bac, E. J. van Henten, J. Hemming, and Y. Edan, "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *J. Field Robot.*, vol. 31, no. 6, pp. 888–911, 2014.