Software Engineering for Self-Adaptive Robotics: A Research Agenda

HASSAN SARTAJ, Simula Research Laboratory, Norway
SHAUKAT ALI, Simula Research Laboratory and Oslo Metropolitan University, Norway
ANA CAVALCANTI, University of York, UK
LUKAS ESTERLE, Aarhus University, Denmark
CLÁUDIO GOMES, Aarhus University, Denmark
PETER GORM LARSEN, Aarhus University, Denmark
ANASTASIOS TEFAS, Aristotle University of Thessaloniki, Greece
JIM WOODCOCK, Southwest University, China, Aarhus University, Denmark, and University of York, UK
HOUXIANG ZHANG, Norwegian University of Science and Technology, Norway

Self-adaptive robotic systems operate autonomously in dynamic and uncertain environments, requiring robust real-time monitoring and adaptive behaviour. Unlike traditional robotic software with predefined logic, self-adaptive robots leverage artificial intelligence (AI), machine learning, and model-driven engineering to adapt to changing conditions continuously, ensuring reliability, safety, and optimal performance. This paper presents a research agenda for software engineering in self-adaptive robotics, structured along two dimensions. The first concerns the software engineering lifecycle—requirements, design, development, testing, and operations—tailored to the challenges of self-adaptive robotics. The second focuses on enabling technologies, i.e., digital twins and AI-driven adaptation, which support runtime monitoring, fault detection, and automated decision-making. We identify open challenges, including verifying adaptive behaviours under uncertainty, balancing trade-offs among adaptability, performance, and safety, and integrating self-adaptation frameworks such as MAPE-K/MAPLE-K. By consolidating these challenges into a roadmap toward 2030, this work contributes to the foundations of trustworthy and efficient self-adaptive robotic systems capable of meeting the complexities of real-world deployment.

CCS Concepts: • **Software and its engineering** → *Software creation and management*; • **Computer systems organization** → **Embedded and cyber-physical systems**; **Robotics**; *Robotic autonomy*; • **Computing methodologies** → **Artificial intelligence**;

Additional Key Words and Phrases: Autonomous Systems, Robotics, Self-Adaptative Systems, Roadmap

Authors' Contact Information: Hassan Sartaj, Simula Research Laboratory, Oslo, Norway, hassan@simula.no; Shaukat Ali, Simula Research Laboratory and Oslo Metropolitan University, Oslo, Norway, shaukat@simula.no,shaukata@oslomet.no; Ana Cavalcanti, University of York, York, UK, ana.cavalcanti@york.ac.uk; Lukas Esterle, Aarhus University, Aarhus, Denmark, lukas.esterle@ece.au.dk; Cláudio Gomes, Aarhus University, Aarhus, Denmark, claudio.gomes@ece.au.dk; Peter Gorm Larsen, Aarhus University, Aarhus, Denmark, pgl@ece.au.dk; Anastasios Tefas, Aristotle University of Thessaloniki, Thessaloniki, Greece, tefas@csd.auth.gr; Jim Woodcock, Southwest University, Chongqing, China and Aarhus University, Aarhus, Denmark and University of York, York, UK, jim.woodcock@york.ac.uk; Houxiang Zhang, Norwegian University of Science and Technology, Aalesund, Norway, hozh@ntnu.no.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM XXXX-XXXX/2025/11-ART
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Hassan Sartaj, Shaukat Ali, Ana Cavalcanti, Lukas Esterle, Cláudio Gomes, Peter Gorm Larsen, Anastasios Tefas, Jim Woodcock, and Houxiang Zhang. 2025. Software Engineering for Self-Adaptive Robotics: A Research Agenda. 1, 1 (November 2025), 47 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Robotic systems are increasingly expected to operate in dynamic, uncertain, and unstructured environments, making self-adaptation a crucial capability. Unlike traditional robots that follow pre-programmed behaviours, self-adaptive robots need to exploit AI and data-driven techniques to autonomously modify their behaviour in response to environmental changes, operational faults, and evolving objectives. This adaptability is crucial in autonomous driving, industrial automation, search-and-rescue, and assistive robotics. For such a level of autonomy, a fundamental characteristic these robots need to possess is self-management capability, including self-configuration, self-optimisation, self-healing, and self-protection [84].

Self-adaptive robotic systems require sophisticated mechanisms to ensure reliability, safety, and robustness while balancing competing demands, including energy efficiency, computational cost, and ethical considerations. The *Monitor-Analyse-Plan-Execute-Knowledge (MAPE-K) loop* serves as a fundamental framework for self-adaptation, enabling robots to continuously assess their environment, plan adaptive responses, and execute modifications while maintaining a knowledge base. Recent advances, such as *MAPLE-K* [92], extend this framework to verify the legitimacy of adaptations. We adopt the widely used MAPE-K loop and its recent extension MAPLE-K, which adds legitimacy checks to improve trustworthiness.

By framing self-adaptation as a continuous loop throughout the entire lifecycle, this roadmap highlights adaptability challenges across all phases of software engineering. Furthermore, it discusses the transformative role of enabling technologies: AI for intelligent and dynamic decision-making and digital twins for real-time simulation. ~~and quantum computing for computationally intensive tasks.~~ Unlike existing roadmaps for self-adaptive systems (e.g., those by Brugali et al. [31], De Lemos et al. [41], Weyns et al. [153]) or domain-specific robotics (e.g., those by García et al. [60], Goues et al. [67]), we focus specifically on self-adaptive robotics. We identify open challenges and research opportunities across the software engineering lifecycle, covering key enabling technologies, and highlighting the critical quality aspects required for deploying future self-adaptive robots in real-world environments that are reliable, safe, and robust.

This paper provides a research agenda for the software engineering of self-adaptive robotics, addressing challenges from two key perspectives: (1) **software engineering lifecycle**, covering requirements engineering, software design, software development, software testing, and operations tailored for self-adaptive robotics; and (2) **key enabling technologies**, exploring the role of digital twins, and AI ~~and quantum computing~~ in supporting autonomous adaptation and decision-making in robotic software systems. By structuring the discussion around these two dimensions, we present a 2030 research agenda to advance self-adaptive robotic software engineering, ensuring robotic systems can effectively navigate unpredictable environments.

Developing and maintaining self-adaptive robotic software presents several challenges from a software engineering perspective. These include (1) requirements engineering for dynamic and evolving specifications; (2) design of scalable and modular architectures that facilitate real-time adaptation; (3) integration of AI techniques, while ensuring explainability and robustness; (4) verification and validation of adaptive behaviours under uncertainty; and (5) exploitation of digital twins for runtime monitoring, fault prediction, and real-time adaptation validation.

The remainder of the paper is structured as follows. Section 2 provides an overview of self-adaptive robotics. Section 3 presents an overview of our roadmap. Sections 4 to 8 cover all phases

of the software engineering lifecycle. Section 9 discusses key enabling technologies in self-adaptive robotics. Section 10 presents a forward-looking research agenda for 2030. Section 11 discusses related work and finally Section 12 concludes the paper.

2 Background

In this section, we provide an overview of software engineering practices for self-adaptive systems. These are systems capable of autonomously adjusting their behaviour in response to changes in their environment or internal state [41]. Self-adaptive systems dynamically adapt to varying conditions without direct human intervention. Due to their broad applicability, self-adaptive systems have become a critical research focus across diverse domains, including autonomic computing, embedded systems, multi-agent systems, and robotics. At the core, such adaptability is enabled by software, which serves as the key driver for self-adaptive capabilities.

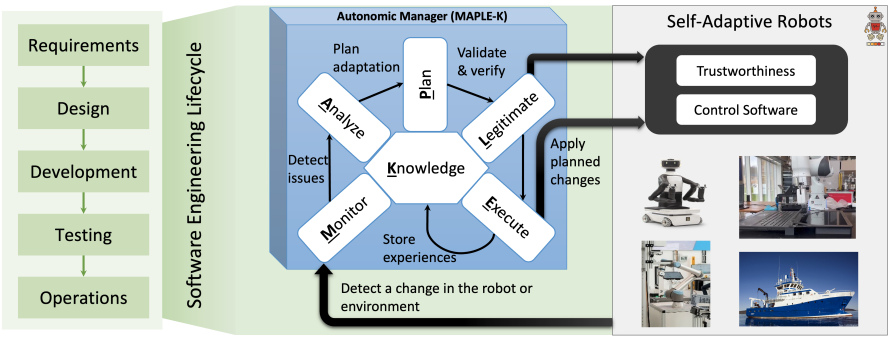


Fig. 1. A conceptual view of the MAPLE-K loop for SAR and the phases of the software engineering lifecycle.

As an example, Figure 1 presents an overview of an approach to developing self-adaptation in robotic systems, highlighting key components of the software architecture, such as the autonomic manager, control software, the trustworthiness analyser, and the associated phases of the software engineering lifecycle. The autonomic manager is central to robotic self-adaptation and operates based on the MAPLE-K loop [92]—an extension of the foundational MAPE-K framework (Monitor, Analyse, Plan, Execute, Knowledge) [84]—with the addition of the *Legitimate* phase to address trustworthiness considerations.

In the *Monitor* phase, the robot continuously collects data from various components, including sensors and the environment, in response to changing conditions. The collected data is then analysed in the *Analyse* phase to understand the current state and predict potential issues and future states. The next phase is *Plan*, which involves determining appropriate actions to adapt to the changes and address potential issues. The subsequent phase is *Legitimate*, which involves verifying and validating the safety of the suggested plan. Finally, in the *Execution* phase, the robot adjusts its behaviour and implements the planned actions. At the core of the MAPLE-K loop is the *Knowledge* phase, which involves maintaining and updating a knowledge base that includes information such as past experiences, environmental conditions, and the outcomes of previous actions.

Software engineering for self-adaptive robotics (SAR) spans the entire software development lifecycle, with each phase addressing distinct adaptability considerations [41]. Requirements engineering involves eliciting and specifying adaptation goals and constraints, as well as runtime evolution and uncertainty. Software design and modelling focus on creating architectures that

support runtime monitoring, decision-making, and reconfiguration, often using modelling languages, formal methods, and simulation models (e.g., Simulink [121]) to capture variability and adaptation strategies. Software development involves implementing control-loop mechanisms, such as the MAPLE-K loop, encompassing monitoring, analysing, planning, legitimising, executing, and managing knowledge to enable adaptive and intelligent system behaviour, as illustrated above. Lastly, quality assurance includes verification and validation activities, such as testing, formal verification, and simulation, to analyse reliability under uncertain and dynamic conditions.

While self-adaptive robots are a special case of cyber-physical systems (CPS), their key differences from other CPS types and software systems arise from robot-specific characteristics such as *embodiment* and *physical interaction*. For example, in traditional CPS and self-adaptive systems, adaptation is largely confined to the computational domain, involving reconfiguring components, reallocating resources, or tuning parameters. In contrast, self-adaptive robots are embodied agents whose physical form may itself vary during operation, requiring the software to remain continuously aware of and responsive to the robot's physical configuration. Below, we list the characteristics of self-adaptive robots that raise challenges discussed in this roadmap.

- **Embodiment:** Individual robots are embodied agents that have to interact with their environment. The shape of their body may vary, potentially even during the operation, requiring the software to adapt accordingly. Overall, a robot needs to be aware of its physical shape and capabilities to operate safely and in accordance with its requirements [46].
- **Physical Interaction:** All robots have a physical interaction with their environment, so there are inherent dynamics with which the robot has to deal. This may include interaction with other robots or even with humans. Any of these interactions, whether active or passive, may introduce uncertainty and unexpected situations that the robot must handle [12]. In addition, accurate system simulations are required to effectively predict and adapt to such dynamic interactions.
- **Localised:** Within the physical environment, the robots need not only to be aware of their own extent but also have an understanding of how and where they are located in the environment. This might be either an absolute, global position or a relative position. This is specifically complex in self-adaptive robotics compared to pure self-adaptive software systems [109].
- **Heterogeneity:** Robots might encounter other robots or systems, different in physical shape, capabilities, resources, goals, or behaviour. This heterogeneity can bring about benefits or additional complexities in identifying potential collaboration or competition among the other robots [47].
- **Distributed Nature:** While a single operator might operate multiple robots, they are usually distributed in the environment. This might require autonomous behaviour and self-adaptation in the absence of a centralised controller or view of the systems as a whole [8, 127].
- **Modularity:** Modular robots are composed of multiple smaller units, allowing for flexible reconfiguration and adaptation to new tasks or environments. This modularity can enhance the robots' capabilities and facilitate collaboration among them. However, this also introduces additional complexity in terms of coordination, communication, and control [165].

In our roadmap, as we discuss challenges and opportunities, we refer to the points above.

3 Overview of the Roadmap

Figure 2 illustrates the structure of the roadmap, organised to cover the software engineering lifecycle, key enabling technologies, and a research agenda. First, we examine the phases of the software engineering lifecycle, which encompass requirements engineering, software design, software development, software testing, and operations. Since software engineering for self-adaptive robotics spans the entire lifecycle, we focus on the phases that have received the most attention in research, while also identifying critical challenges that remain unresolved.

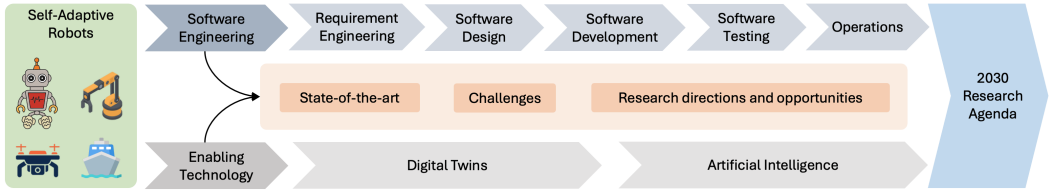


Fig. 2. An overview of the roadmap, illustrating its structure and the aspects covered in the software engineering lifecycle and enabling technologies.

Secondly, we focus on key enabling technologies: digital twins and AI. ~~and quantum computing.~~ Technologies such as digital twins and AI have already demonstrated a significant role in advancing self-adaptive robotics by enabling real-time simulations, decision-making, and automation. ~~Meanwhile, emerging technologies like quantum computing hold significant potential but remain largely unexplored, leaving several promising research directions open for the future. For both aspects,~~. We provide a detailed roadmap by analysing the current state of the art, identifying open challenges, and highlighting potential future research directions.

Finally, we present our comprehensive research agenda for software engineering for self-adaptive robotics towards 2030. This agenda outlines a forward-looking vision and provides a detailed roadmap for researchers in robotics interested in addressing uncertainty.

4 Software Requirements Engineering for SAR

We begin with an overview of related research (Section 4.1), providing context for the state of the art discussion. Following this, we describe key challenges that remain to be addressed (Section 4.2).

4.1 Foundations and Related Research

Software Requirements Engineering (SRE) plays a critical role in ensuring the success of self-adaptive robotic systems. These systems operate in highly dynamic and unpredictable environments, making traditional requirements specification methods insufficient. For robotics, SRE must capture both functional requirements, such as perception, navigation, and interaction with the physical world, and non-functional requirements, such as safety, adaptability, and real-time performance.

Goal-oriented requirements engineering (GORE), formal methods, and model-driven approaches have been used for robotic SRE. Goal-oriented approaches, such as those based on KAOS and i frameworks, enable the capture and refinement of high-level goals into operational requirements [146]. Formal methods, such as Z notation and temporal logic, are gaining traction in robotics for specifying safety-critical requirements and verifying their correctness [94]. Model-driven approaches, such as SysML-based methods, facilitate the integration of requirements with system design and verification [54]. Recent efforts have focused on runtime requirements monitoring for self-adaptive systems, where requirements are treated as dynamic entities that evolve as the robot interacts with its environment [88].

4.2 Challenges and Open Questions

Despite significant advances, several challenges remain in applying SRE to self-adaptive robotics.

4.2.1 Dynamic and Context-Aware Requirements Specification. Self-adaptive robotics requires continuous adaptation to changing environments and tasks, raising the need for context-aware and evolving requirements. Current methods struggle to handle requirements that must be updated at

runtime while maintaining system consistency and safety guarantees. How can requirements be dynamically specified, validated, and adapted in real-time?

4.2.2 Trade-offs between Conflicting Requirements. Self-adaptive robotic systems must balance conflicting requirements, such as energy efficiency, speed, and safety. Resolving these trade-offs at design time is complex. It becomes even more challenging during runtime adaptations [73, 128]. Developing automated techniques to manage and prioritise requirements trade-offs in self-adaptive scenarios dynamically remains an open question.

4.2.3 Requirements Traceability and Verification. It is challenging to ensure traceability from high-level requirements to implementation and maintain it across system adaptations. Verification and validation (V&V) approaches must evolve to provide runtime assurances after adaptation. How can we ensure that traceability and V&V processes are scalable and efficient in self-adaptive robotics?

4.2.4 Stakeholder Involvement and Elicitation for Complex Scenarios. The involvement of diverse stakeholders (roboticists, end-users, and domain experts) makes elicitation challenging. The unpredictable nature of robotics applications, particularly in search and rescue, healthcare, and autonomous driving, complicates requirements gathering. Collaborative approaches and user-centred design methodologies must be extended to better capture evolving requirements.

Normative requirements

Shortage of skilled labour across industries (healthcare, agriculture, logistics, and so on) motivates the deployment of autonomous robots in human environments. To achieve public acceptance, these robots need to satisfy a complex set of normative requirements, encompassing social, legal, ethical, and cultural expectations. There is significant progress [29, 55] in defining normative concepts [78], and a growing number of compliant applications [7]. The UK Trustworthy Autonomous Systems (TAS) Resilience project has created a collaborative method [144] for defining normative rules for a robot captured in SLEEC (Social, Legal, Ethical, Empathetic, and Cultural), a flexible, highly expressive language, incorporating features of defeasible logic to handle exceptions, and defining time constraints [51, 52, 161]. Work by others has established a logic semantics for a simplified version of SLEEC and a connection to Prolog [145]. Verification efforts have recently focused on the transparency, ethical, and legal aspects of high-level autonomous system designs [20, 42]. Separately, studies on multi-robot systems primarily focus on inter-robot trust rather than on the design of individual norm-aware robots [58]. Preliminary work is just beginning to explore the realisability of normative requirements, taking into account factors like the controllability. There are two key challenges: (**NR-Ch-1**) assuring that the robot meets all normative requirements upon initial deployment, and continuously satisfies norms while working in uncertain, evolving scenarios; and (**NR-Ch-2**) providing clear explanations for their norm-sensitive decisions.

4.2.5 Ethical and Societal Requirements. Self-adaptive robotics brings ethical concerns, including privacy, fairness, and accountability. Defining and enforcing ethical requirements in robotic systems is an emerging research area that requires multidisciplinary collaboration. Questions about operationalising these requirements and integrating them into existing SRE processes remain open. We refer to the inset for more details and an example.

4.2.6 Challenges Summary. Addressing the challenges mentioned above requires new methodologies and tools that combine formal, goal-driven, and adaptive approaches. Integrating run-time

monitoring, feedback loops, and automated reasoning into requirements engineering processes will be crucial for advancing the state of the art in this domain. In the following, we present a summary of key SRE challenges for SAR. The graph in Figure 3 shows the dependencies among the challenges, indicating their priorities and timelines. Stakeholder involvement and elicitation (SRE-Ch-4) is foundational, since the system cannot manage adaptation responsibly unless it first captures the goals, contexts, and constraints of the relevant actors. This, in turn, enables the systematic incorporation of ethical and societal requirements (SRE-Ch-5), which should be addressed early because they constrain acceptable adaptation behaviour. Once these foundations are in place, the next priority is managing conflicting requirements (SRE-Ch-2), followed by traceability and scalable verification and validation (SRE-Ch-3), which together provide the assurance infrastructure needed for safe runtime requirement updates (SRE-Ch-1). In parallel, the normative requirements line should progress from ensuring that deployed robots satisfy applicable norms in evolving situations (NR-Ch-1) to providing explanations for norm-sensitive decisions (NR-Ch-2). The graph should therefore be read as a flexible roadmap rather than a rigid sequence.

Challenges in Software Requirements Engineering for SAR

SRE-Ch-1: Handling runtime requirement updates while ensuring system consistency and safety.
SRE-Ch-2: Balancing conflicting requirements.
SRE-Ch-3: Ensuring traceability of requirements and scalability of verification and validation activities.
SRE-Ch-4: Eliciting requirements due to diverse stakeholders and the nature of robotics applications.
SRE-Ch-5: Integrating ethical requirements into SRE processes.

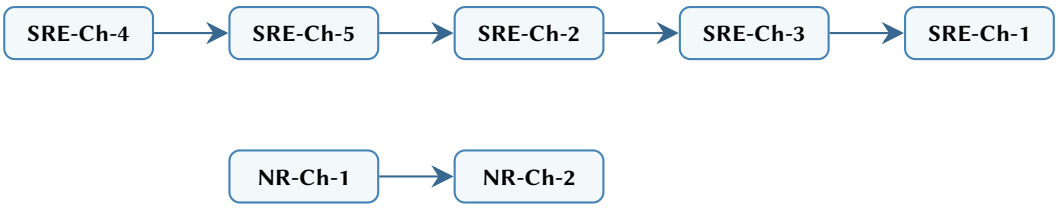


Fig. 3. Dependency graph for challenges in SRE and normative requirements for SAR.

5 Software Design for SAR

In this section, we begin with an overview of the SAR software design and its associated challenges. We then discuss key aspects of SAR software design in the context of model-driven engineering and simulation by reviewing literature and highlighting open research challenges. Finally, we conclude with a summary of the key insights presented.

5.1 Overview and Principles

Software design for SAR is a multidisciplinary endeavour combining principles from software engineering, robotics, AI, and control systems. Its goal is to provide robust, adaptable, and scalable architectures that support continuous changes in tasks, environments, and configurations while ensuring safety and efficiency. Several principles underpin design for SAR.

Modularity and Scalability. Component-based and service-oriented architectures promote modular and scalable development. Middleware such as the Robot Operating System (ROS) and architecture-centric approaches to robotic software provide essential building blocks for integration and evolution across heterogeneous platforms [5, 32, 111].

Abstraction and Automation. Model-driven development (e.g., UML and SysML) facilitates high-level abstraction, analysis, and automated code generation, enabling systematic transitions from design models to implementation [43, 54, 125].

Cross-cutting Concerns. Architectural and aspect-oriented techniques help to manage concerns such as fault tolerance, resource management, and assurance obligations without scattering them across multiple modules [16, 111, 168].

Self-Adaptive Architectures. Feedback control loops such as MAPE-K provide a principled basis for runtime monitoring, analysis, planning, and reconfiguration, and recent work has begun to adapt these ideas explicitly to trustworthy robotic software [41, 84, 92, 116]. AI-based techniques are increasingly integrated to enhance decision-making and adaptation strategies [5, 41].

5.2 Core Challenges in Software Design

Despite advances in methodologies, SAR software faces several unresolved research questions:

Scalable and modular architectures. How can architectures balance flexibility and efficiency while supporting continuous adaptation across diverse robotic platforms?

Trade-offs between adaptability, safety, and performance. How can systems remain reliable without sacrificing adaptability or degrading performance?

Runtime evolution and reconfiguration. How can software be safely updated during operation while maintaining consistency and fault tolerance?

Handling uncertainty in AI-based adaptation. How can design practices mitigate unpredictability when AI is used for adaptation?

Model-driven engineering integration. How can runtime models remain consistent with design-time models to support lifecycle adaptation?

Human-in-the-loop adaptation. What frameworks and interfaces best support human-system collaboration while minimising cognitive load?

Verification and validation. What novel V&V methods can assure the correctness of highly dynamic and safety-critical self-adaptive software?

In the following, we present a concise description of challenges corresponding to the research questions outlined above. These challenges arise directly from the design principles introduced above. SD-Ch-1 follows from the need for modularity and scalability; SD-Ch-5 from abstraction and automation in model-driven engineering workflows; SD-Ch-3 and SD-Ch-6 from self-adaptive architectures that must support safe runtime reconfiguration and meaningful human oversight; SD-Ch-4 from the growing use of AI within adaptation mechanisms; SD-Ch-2 cuts across all of these principles because architectural decisions must balance adaptability, safety, and performance; and SD-Ch-7 is a cross-cutting assurance challenge that spans them all. The graph in Figure 4 shows the dependencies among the challenges, indicating their priorities and timelines. From a timing perspective, scalable and modular architectures (SD-Ch-1) and the management of trade-offs between adaptability, safety, and performance (SD-Ch-2) provide the architectural foundation and should be addressed first, and largely in parallel. Once that foundation is in place, three strands can advance: maintaining consistency between runtime and design-time models (SD-Ch-5), handling uncertainty in AI-based adaptation (SD-Ch-4), and strengthening verification and validation strategies (SD-Ch-7). Progress on runtime-model consistency then enables safer

runtime evolution and reconfiguration (SD-Ch-3). Safe reconfiguration and principled handling of AI uncertainty together support effective human-in-the-loop adaptation (SD-Ch-6). The graph should therefore be read as a flexible roadmap rather than a rigid sequence: early work should focus on architectural structure and design trade-offs, mid-term work on model consistency and AI uncertainty, and later integrated work on safe reconfiguration, human oversight, and system-level assurance.

Challenges in Software Design for SAR

SD-Ch-1: Scalable and modular architectures that balance flexibility and efficiency while supporting continuous adaptation across diverse platforms.
SD-Ch-2: Managing trade-offs between adaptability, safety, and performance so that reliability is preserved without degrading system behaviour.
SD-Ch-3: Safe runtime evolution and reconfiguration with consistency and fault tolerance during software updates in operation.
SD-Ch-4: Handling uncertainty introduced by AI-driven adaptation, including mitigating unpredictability and ensuring trustworthiness.
SD-Ch-5: Keeping runtime models consistent with design-time models to support lifecycle adaptation in MDE workflows.
SD-Ch-6: Human-in-the-loop adaptation with interfaces and processes that minimise cognitive load and enable effective collaboration.
SD-Ch-7: Verification and validation techniques suited to highly dynamic and safety-critical self-adaptive software.

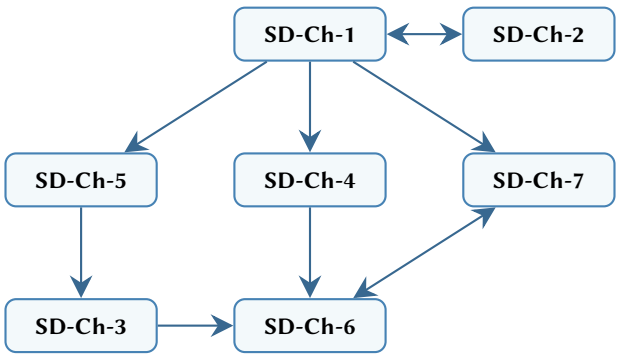


Fig. 4. Dependency graph for challenges in software design for SAR.

5.3 Model-Driven Engineering for SAR

5.3.1 Foundations and Related Research. Model-Driven Engineering (MDE) has been successfully applied in domains such as aerospace and automotive, and offers strong potential for SAR. By treating abstract models as central artefacts, MDE supports requirements clarification, early validation, and automated code generation.

Domain-Specific Languages (DSLs). Numerous DSLs have been proposed for robotics, many with formal semantics enabling reasoning. Examples include UML derivatives such as mUML [61], UML profiles for human–robot collaboration [10], SysML with CSP-like semantics [98], AADL

with rewriting logic [118], RobotML [43], SafeRobots [123], and RoboChart [71]. Frameworks such as MontiArcAutomaton [125] integrate multiple modelling languages and generators for heterogeneous platforms.

Verification Support. Several approaches incorporate formal verification, including contract languages and temporal logic (Mauve [64], Orccad [86], SPECTRA [108]), theorem proving (AutoFocus and Isabelle/HOL transformations [139]), and model checking (GenoM with Petri Nets [57], JavaPathFinder [112]). RoboStar [37] provides domain-specific notations and transformations to support modelling, simulation, testing, and verification.

5.3.2 Challenges and Open Questions. There are, however, many open challenges in the effective use of these notations for MDE. That requires automation based on sound techniques that can ensure the values of the derived artefacts and compensate for the effort required to develop models. Some open problems are as follows.

Formalisation of Architectures for Adaptation. There is no standard or formal understanding of what it means for a self-adaptive system to adopt the MAPE-K architecture. This impacts the traceability between the code and the MAPE-K design of an adaptive system. It also makes it difficult to exploit the features of the MAPE-K architecture at the code level.

Support for Code Generation. The lack of MAPE-K formalisation leads to the lack of customised facilities to model and generate code that adopts that architecture. While general modelling languages and MDE tools can be employed, none provides domain-specific support, i.e., no native support to ensure the communication patterns and control flow of MAPE-K, for example, are enforced at either the model or coding levels.

Hybrid Reasoning. Adaptation is often a response to robotic hardware and environment changes, which are often better described via hybrid models. While MBE is often focused on the software component, we need to employ hybrid models to generate simulations, identify meaningful tests, and reason about adaptation in self-adaptive robotics. There is a rich literature on hybrid model checkers and theorem provers. However, a practical approach capable of handling the scale and complexity of robotic models remains missing.

Integration with AI. MBE techniques are often component-based to a certain level, with at least some structure in the models to reflect commonly used architectures or computational resources. Extensions of the notations and techniques for dealing with AI components are crucial to enabling the system-level approach necessary for adaptation.

Human Factors. Again, with the observation that adaptation is a system-level concern, we have to consider the impact of human factors. There are two perspectives on human behaviour, which lead to a requirement for adaptation and to an assessment of the impact of adaptation on human stakeholders. Much work on ethical factors is being undertaken, but we require full operationalisation to support the deployment of suitable adaptations.

Challenges Summary. In the following, we outline key challenges in MDE for SAR. The graph in Figure 5 shows the dependencies among the challenges, indicating their priorities and timelines. A standard architecture (MDE-Ch-1) is a key enabling result. It is more useful if it can support code generation (MDE-Ch-2), hybrid reasoning (MDE-Ch-3), support for AI (MDE-Ch-4), and consideration of human factors and ethics (MDE-Ch-5). Conversely, the existence of a standard architecture can enable (compositional) reasoning approaches (MDE-Ch-3). In addition, code generation is more (MDE-Ch-2) if it can cater for AI techniques (MDE-Ch-4).

Challenges in MDE for SAR

MDE-Ch-1: Lack of a formal standard for self-adaptive architectures (e.g., MAPE-K), hindering traceability between models and code.
MDE-Ch-2: Limited support for automated code generation that reflects adaptation-specific semantics.
MDE-Ch-3: Need for hybrid reasoning that combines software with models of hardware and environment.
MDE-Ch-4: Insufficient integration of AI components into MDE frameworks.
MDE-Ch-5: Limited treatment of human and ethical factors in adaptation decisions.

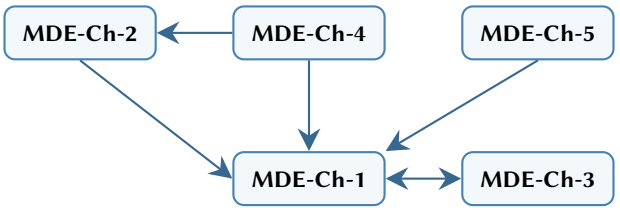


Fig. 5. Dependency graph for challenges in MDE for SAR.

5.4 Simulation Approaches for SAR

5.4.1 Foundations and Related Research. Simulation offers a risk-free and cost-effective means of validating designs and exploring adaptation strategies before deployment.

Foundations. A model is a representation of a system capturing relevant properties. Simulations produce traces of behaviour under varying conditions, with fidelity, substitutability, and verifiability determining their usefulness [91].

Formal Simulation Approaches. ActivFORMS [154] employs timed automata to model MAPE-K loops, supporting runtime verification and execution of adaptation logic. Digital twin frameworks integrate simulations into monitoring and planning [49], enabling predictive what-if analyses and reachability-based verification [155]. Other approaches include agent-based models [99] and adaptive Abstract State Machines (ASMs) [9], which support distributed adaptation and emergent behaviours.

Co-simulation. Heterogeneous subsystem models can be coupled using frameworks such as FMI [83], HLA [115], and DEVS [23]. Orchestration algorithms synchronise simulators, but challenges remain in accurately capturing reconfiguration dynamics and ensuring intellectual property protection.

5.4.2 Challenges and Open Questions. Despite the tremendous work by the community, challenges and open research questions remain in simulating self-adaptive systems.

Adapting Simulations. Simulation of self-adaptive systems requires models that have a broad range of scenarios where the model is relevant. Typically, this is achieved by selecting the most appropriate model for any given scenario [21]. This highlights the challenge of representing the range of valid scenarios a model can be used in, in a compact format, that is *shipped* alongside the model. The representation of this range must be machine-readable and highly expressive, as it

can vary from parameter and initial condition ranges to more dynamic descriptions, such as the maximum frequency of model inputs. So far, no universal description has emerged in the state of the art.

Coupling Simulations. In modern supply chains, a self-adaptive system is formed by subsystems provided by external suppliers [50]. This leads to difficulties in simulating the entire system using a single formalism, such as an ASM. An existing solution to couple heterogeneous system models is co-simulation [66]. In this technique, subsystem simulators are heterogeneous, but they implement the same interface. This allows the coupling of these simulators to form a composite simulation, but it has not been widely used in the context of self-adaptive system simulation.

Orchestration Challenge in Co-simulation. There are multiple interfaces that sub-simulations can implement, such as the Functional Mock-up Interface (FMI [83]), High-Level Architecture (HLA [115]), DEVS [23], and others. When the subsystems implement a standard interface, co-simulation is achieved by intentionally stepping each simulator forward in simulated time while exchanging data between each step with the other simulators [90]. The algorithm that manages this process is called the orchestration algorithm. Orchestration algorithms proposed in the state of the art range from simple and general [15] to complex and highly specialised [74], exhibiting self-adaptive properties [75]. Co-simulation of self-adaptive robotics requires orchestration algorithms that differ from traditional ones to ensure that system reconfigurations are accurately captured in the simulation dynamics, including dynamically changing simulator dependencies [14, 45]. This is an open area of research.

Limitations of Co-simulation Interfaces. Co-simulation interfaces may not be rich enough to enable the orchestrator to capture dynamics correctly. For instance, traditional co-simulation of multi-rate systems is challenging in FMI version 2.0 [25]. Recently, FMI 3.0 has mitigated this issue by introducing the notions of clocks and clocked variables [65]. Conversely, error-free simulation of continuous dynamics in discrete-event simulation interfaces (such as DEVS and HLA) is impossible to achieve due to quantisation [23].

Intellectual Property Protection in Simulation. In modern supply chains, there is a need to protect the intellectual property (IP) of subsystems, as external companies often provide these. To avoid expensive contracts, the co-simulation interface must promote IP protection. The question remains whether traditional IP protection mechanisms in existing co-simulation interfaces are sufficient for simulating self-adaptive systems.

Challenges Summary. A summarised list of key challenges related to simulations in SAR is given below. The graph in Figure 6 highlights a tightly coupled core formed by heterogeneous coupling (Sim-Ch-2), orchestration (Sim-Ch-3), and interface limitations (Sim-Ch-4). Progress in any one of these areas depends on and reinforces progress in the others: richer interfaces enable better orchestration, while the needs of orchestration and coupling expose limitations in current interfaces. Addressing this core also enables more practical support for compact machine-readable scenario ranges (Sim-Ch-1) and stronger IP protection mechanisms (Sim-Ch-5) for supplier-provided subsystem models.

Challenges in Simulations for SAR

Sim-Ch-1: Compact representation of scenario ranges for adaptive simulations.
Sim-Ch-2: Coupling heterogeneous subsystem simulations to form a composite simulation.
Sim-Ch-3: Orchestration algorithms that can dynamically adapt to system reconfiguration.
Sim-Ch-4: Co-simulation interfaces often lack the richness required to capture multi-rate and hybrid dynamics.
Sim-Ch-5: Protection of intellectual property when coupling externally provided subsystem models.

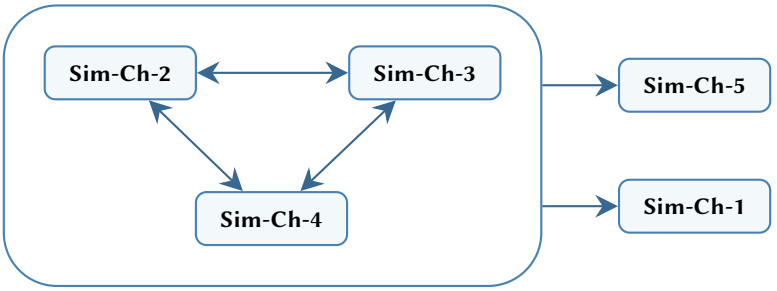


Fig. 6. Dependency graph for challenges in simulations for SAR.

5.5 Summary

Software design for SAR must integrate architectural principles, MDE, and simulation to achieve adaptability without compromising safety or performance. The open challenges (SD-Ch-1 to SD-Ch-7) highlight the need for scalable architectures, robust integration of AI and human input, and advanced V&V methods. Bridging MDE and simulation approaches—while extending them to handle hybrid models, AI, and ethical concerns—remains a critical research direction.

6 Software Development of SAR

In this section, we begin with an overview of research on the development of SAR software. Following this, we outline open challenges that need to be addressed.

6.1 Foundations and Related Research

Software development for SAR focuses on implementing advanced self-adaptation mechanisms, such as MAPE-K/MAPLE-K loops, to enable robots to autonomously adjust their behaviour, configurations, and decision-making processes in response to dynamic and uncertain environments [16]. Given the inherent complexity of SAR systems, modularity, scalability, and maintainability are crucial design principles to ensure seamless integration across subsystems and with evolving requirements. Therefore, the development of SAR software is closely aligned with design models and architectures such as AADL or RoboArch [16], which provide systematic abstractions from high-level designs to implementation. Additional considerations include integrating advanced frameworks and technologies, such as machine learning and data analysis, to support real-time decision-making, interactions in uncertain environments, and collaboration with humans or other robots.

A common method for developing SAR software is to utilise the models and architectures created in the design phase [5]. In this regard, several techniques have been proposed in the literature targeting different aspects of self-adaptation. Edwards et al. [44] employed high-level architectural components to develop self-management capabilities, including monitoring and self-adaptation. To improve runtime reconfigurations, Bao et al. [11] proposed a model-based approach that employs meta-models of a robotic system to generate online configurations. Moreover, Rivera and Chandrasekaran [126] presented an initial effort on developing self-recovering and self-healing capabilities in robotic software.

To support automation in SAR software development, several studies have focused on using design models, such as meta-models and runtime models, for automatic code generation [76, 77]. Similarly, Medvidovic et al. [111] presented RoboPrism, a framework for architecture-driven development of robotic software. Focusing on uncertainty as a key concern, Alberts [4] introduced a reasoning-based approach for implementing and integrating uncertainty-aware adaptation strategies. Silva et al. [137] presented SUAVE, a framework for developing self-adaptation in autonomous underwater vehicles (UUVs) considering uncertainties. In a recent work, Nezhad et al. [116] presented a framework for model-based development of the MAPE-K loop that accounts for the trustworthiness of robotic software.

6.2 Challenges and Open Questions

Despite significant advances in SAR software development, several challenges remain to be addressed.

6.2.1 Interfacing with Heterogeneous Components. Self-adaptive robots often rely on a wide range of hardware components (e.g., sensors, actuators, cameras, and LIDAR) and software platforms (e.g., control systems, communication protocols, and AI models). These components are typically developed by different vendors using varying standards, making seamless integration a complex task. For example, a robotic system may have an LIDAR sensor that requires a proprietary driver and actuators that rely on custom firmware. Developing software that effectively interfaces with heterogeneous components and platforms, ensuring seamless communication, compatibility, and adaptability, remains a critical challenge.

6.2.2 Rapidly Changing Hardware and Software. Both the hardware and software of robots continuously evolve at different paces, with robotic software upgrades occurring more frequently than hardware upgrades. To keep up with adaptations to different environments, AI components need to evolve, which requires retraining or fine-tuning of AI models. Similarly, whenever hardware components are upgraded or replaced, the corresponding software must be changed to accommodate hardware advancements. Thus, supporting the continuous integration of new/updated hardware and software into self-adaptive robotic systems without requiring a complete overhaul of the existing software architecture is a major challenge.

6.2.3 Sensor Fusion for Adaptation. The software components for data analysis rely on data from multiple sensors, such as cameras, LIDAR, and GPS, to perceive the robots' operating environment and decide actions. However, these sensors often provide data in different formats, sampling rates, accuracies, and noise levels. Misaligned or inconsistent sensor data can lead to incorrect decisions, such as failing to detect obstacles, which can in turn result in unsafe actions. Developing software capable of real-time sensor fusion to integrate data from various sensors while ensuring reliability and accuracy remains an open challenge. Given the recent rise of large language models with advanced analytical capabilities, one potential direction is to explore their applicability to real-time data fusion and analysis.

6.2.4 Energy-Efficient Software Development. Developing intelligent software components commonly involves developing AI models for various types of decision-making. For example, the perception module requires training AI models on image/video data. As the robot operating environment changes, AI models need to be retrained on the new imagery data. Furthermore, robotic software involves several computationally expensive algorithms, such as path planning and adaptive navigation, to run during the robot's operations in real environments. Developing energy-efficient robotic software that minimises computational overhead while maintaining the robot's ability to adapt in real-time is a critical challenge.

6.2.5 Ethical and Security Concerns. As self-adaptive robots operate autonomously and interact with humans, they raise significant ethical and security concerns. For example, a caregiver robot that assists older people must ensure privacy while handling their personal data, such as health records or behavioural patterns. Such concerns, although specified in software requirements, are critical from an implementation perspective. For example, developing stronger security and privacy mechanisms to make software more resilient to cyberattacks, hijacking, or other malicious attacks. Therefore, developing robotic software capable of handling unknown and dynamic security and privacy concerns at runtime remains an open challenge.

6.2.6 Challenges Summary. In the following sections, we outline the challenges associated with SAR software development. The graph in Figure 7 illustrates the dependencies among the challenges, indicating their relative priorities and timelines. At the foundation, sensor fusion (Dev-Ch-3), sustainable robotic software (Dev-Ch-4), and resilience against cyber attacks (Dev-Ch-5) are interdependent and should be addressed concurrently, with Dev-Ch-5 having the highest priority among the three. Once these foundational challenges are resolved, the next priority is to address hardware/software integration (Dev-Ch-2). Finally, implementing adaptations and evolution (Dev-Ch-1) has comparatively lower criticality and can be addressed in the later stages of development.

Challenges in Software Development of SAR

Dev-Ch-1: Ensuring effective integration and compatibility between various hardware and software components from different vendors.
Dev-Ch-2: Adapting to continuously evolving hardware and software without rebuilding existing architectures.
Dev-Ch-3: Developing real-time sensor fusion to integrate diverse sensor data.
Dev-Ch-4: Developing robotic software in an energy-efficient manner and providing sustainable operations in real environments.
Dev-Ch-5: Ensuring privacy, data security, and robustness against cyberattacks at runtime.

7 Software Testing of SAR

This section begins with an overview of research on software testing in SAR. Then it highlights a set of open challenges that remain to be addressed.

7.1 Foundations and Related Research

Testing plays a key role in ensuring that self-adaptive robots function as intended while meeting all safety and dependability requirements. However, the dynamic and autonomous nature of these systems makes testing significantly challenging. Therefore, SAR system testing is conducted in different setups: software-in-the-loop (SiL), which evaluates robot software components within

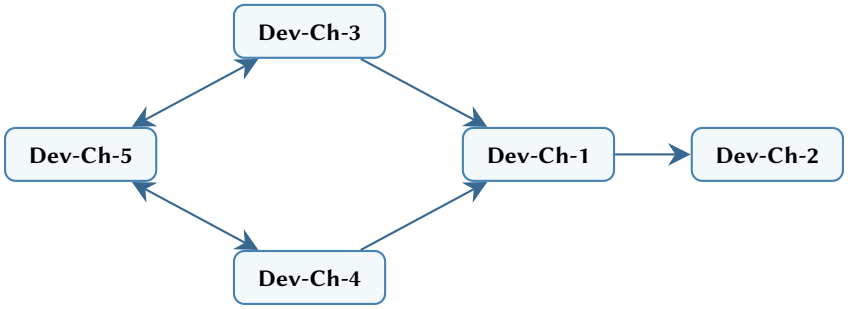


Fig. 7. Dependency graph for challenges in software development of SAR.

a simulated environment; model-in-the-loop (MiL), which verifies robot control logic through simulation models; and hardware-in-the-loop (HiL), which tests interaction between software and hardware by incorporating hardware components, such as sensors and actuators, into the testing process. Moreover, operational testing with physical robots in a controlled environment is often performed, typically supported by advanced technologies such as digital twins, which allow real-time monitoring and validation of the robot's performance.

Many existing works focus on testing various aspects of robots, with model-driven approaches being prevalent. One such framework, RoboStar, provides a model-based solution for automatic test generation for autonomous robots [37]. Another approach utilises mutation testing to automatically generate tests by mutating RoboChart models [71]. For uncrewed aerial vehicles (UAVs), a model-based approach has been proposed to automate the testing of interactive remote control systems [134]. Similarly, a system-level testing framework, AITester, combines model-driven techniques with reinforcement learning to enable automatic and dynamic testing of unmanned aerial systems [130, 135]. In addition to model-driven approaches, search-based test generation has been applied to test autonomous systems using requirements and constraint specifications [1, 133]. Furthermore, field-based testing techniques for robots, such as in-house, online, and offline field testing, are also common [34].

When it comes to testing the self-adaptive behaviour of robots, only a limited number of works are available in the literature. In this context, a SiL testing framework powered by machine-learning techniques has been proposed to test the behaviour of autonomous systems, such as UUVs [113]. To test self-healing behaviour, a model-based framework was introduced to automate testing of autonomous CPS under uncertainty [102]. Subsequently, reinforcement learning algorithms were evaluated to test the self-healing behaviour of various autonomous vehicles, including UAVs and mobile robots [103]. In a similar line of work, a model-based approach was proposed to test the self-adaptive behaviour of UAVs [81]. Building on this, a hybrid approach was introduced that combines model-based and search-based methods to test self-adaptive UAVs [82]. Recently, an approach has been developed for testing autonomous mobile robots that uses robot requirements and vision language models to generate test scenarios [156]. Another direction of research focuses on the use of digital twins for verifying robot behaviour at runtime. For instance, digital twins have been employed to monitor and verify the behaviour of mobile robots under uncertainty [18]. Similarly, a digital twin-based approach has been introduced for real-time out-of-distribution detection in self-adaptive robots, including autonomous vessels and service robots [79, 80]. Furthermore, an initial AI-based digital twin framework has been introduced to enable task verification for medical robots [106].

7.2 Challenges and Open Questions

Although considerable progress has been made in SAR software testing, several challenges remain to be addressed.

7.2.1 Simulation-based Testing in Varied Reality Gap. A key challenge in simulation-based testing (e.g., in a SiL setup) is the reality gap—the discrepancy between simulation and real-world conditions. This makes realistic testing in simulation difficult, as test scenarios may not accurately reflect real-world situations (e.g., weather changes). Highly realistic simulations can be costly, yet full realism may not always be necessary. This raises a key question: What level of realism strikes the balance between effectiveness and cost? In practice, this question is often addressed through a structured progression of simulation and testing environments, such as MiL simulation for rapid development in early stages, SiL simulation for integration testing, and HiL simulation for testing with real hardware components [40]. Although such progressions are effective, transferring behaviours, scenarios, and test artefacts across these stages remains challenging due to differing assumptions, sensor characteristics, and timing behaviour [33, 143]. This presents an opportunity to exploit AI-based techniques to relate and adapt test scenarios, models, and observed behaviours across multiple stages of the testing pipeline. Specifically, AI foundation models with multimodal capabilities (e.g., vision-language models) offer promising ways to enhance simulation realism and improve testing effectiveness.

7.2.2 Testing AI and non-AI Components. AI is increasingly integrated into self-adaptive robotic software (including MAPE-K) for tasks such as object identification and autonomous decision-making. However, this introduces testing challenges for both AI and non-AI components independently and due to their interactions, including uncertainty, lack of explainability, and ethical concerns. Effective testing methodologies must address these issues.

7.2.3 Testing under Uncertainty. Uncertainty is inherent in self-adaptive software. Thus, testing methods must treat uncertainty as a first-class entity to identify faults effectively. Holistic approaches are needed to quantify uncertainty across different sources, including AI components. Furthermore, continuous assessment and management of uncertainty in real-world operations are crucial, along with strategies for handling cases where uncertainty exceeds acceptable limits, to ensure system reliability and safety.

7.2.4 Testing Self-Adaptive MAPE-K Loop. Self-adaptation, often via the MAPE-K loop, introduces testing challenges. New methods are needed to test each component, such as whether Analyse triggers adaptations, Plan generates appropriate strategies, and adaptations execute correctly. Testing must assess both individual components in MAPE-K and the entire loop to ensure overall correctness.

7.2.5 Real-Time Testing through Digital Twins. Digital twins are becoming increasingly prevalent in robotics. Our focus here is on testing challenges; more details on digital twins in the context of self-adaptive robotics are provided in Section 9.1. Whether built using model-driven approaches, AI, or a combination of both, digital twins provide a continuous means of assessing and testing self-adaptive robots, helping prevent failures in real-world operations. However, there are several testing challenges. One key challenge is enabling effective real-time testing while accounting for potential synchronisation delays between the physical system and its digital twin. Such delays are quite common due to communication issues. Within digital twins, there is also a need for validation methods (e.g., AI-based) that can predict or forecast potential failures, enabling the digital twin to prevent them before they occur.

7.2.6 Testing under Continuous Evolution. Self-adaptive robotic software evolves continuously, requiring cost-effective testing of modified parts while ensuring existing functionality remains intact. This requires new regression testing methods for the MAPE-K loop and AI components to support continuous evolution.

7.2.7 Testing Robotic Foundation Models. Recent advancements in foundation models, such as large language models (LLMs), have led to the development of robotic foundation models. These models include LLMs that are specifically fine-tuned for robotic tasks, as well as vision-language-action (VLA) models designed for various types of robots. However, this paradigm shift in robotics also introduces several critical challenges that need to be addressed. A key challenge in robotic foundation models is hallucinations, which pose a significant safety risk in robotics as they can lead to incorrect or unsafe behaviours in real-world environments [158]. Developing testing techniques to assess the robustness of robotic foundation models remains an open challenge. Another challenge is inherent nondeterminism and output variability in robotic foundation models, which lead to inconsistent actions and decision-making [136]. Moreover, these models lack grounding, i.e., a limited understanding of real-world and dynamic contexts, which is crucial for effectively controlling robots in complex environments [68]. Therefore, the development of reliability testing techniques for robotic foundation models, especially when integrated with physical robots, remains a critical and open research problem.

7.2.8 Security Testing of Self-adaptive Robotics. With the increasing integration of AI, especially LLMs in robotics, several new types of vulnerabilities and security threats are emerging [157]. These include prompt-based attacks on LLMs and perception-based attacks on vision-language models (VLMs), intended to manipulate the robot's decision-making. Although recent efforts have focused on addressing these attacks and mitigating associated risks [149, 169], developing effective security testing techniques to ensure robots' resilience against adversarial threats remains an open research challenge.

7.2.9 Challenges Summary. In the following, we present a summary of key challenges in testing SAR software. The graph in Figure 8 illustrates the dependencies among the testing challenges, indicating their relative priorities and timelines. A fundamental challenge is ensuring the correctness of MAPE-K (ST-Ch-4), which must be addressed first. Subsequently, resilience against adversarial threats (ST-Ch-8), testing AI and non-AI component interactions (ST-Ch-2), and testing under uncertainty (ST-Ch-3) can then be tackled in parallel, with ST-Ch-2 and ST-Ch-3 being interdependent and ST-Ch-2 relying on ST-Ch-8. Once ST-Ch-2 and ST-Ch-3 are addressed, they collectively form the basis for addressing robustness and reliability (ST-Ch-7), which is the next priority. Following this, bridging the reality gap (ST-Ch-1) and testing digital twins (ST-Ch-5) can be pursued concurrently, as the two are interdependent. Finally, regression testing (ST-Ch-6) represents the last challenge to be resolved once all preceding priorities have been addressed.

Challenges in Software Testing for SAR

ST-Ch-1: Bridging the reality gap in simulation-based testing to ensure realistic yet cost-effective testing.
ST-Ch-2: Testing the interactions between AI and non-AI components, while considering uncertainty and ethical concerns.
ST-Ch-3: Testing SAR software in the presence of uncertainty.
ST-Ch-4: Ensuring the correctness of individual MAPE-K components and the overall loop during testing.
ST-Ch-5: Real-time testing of digital twins, while accounting for synchronisation delays with physical systems.
ST-Ch-6: Developing cost-effective regression testing methods to support the continuous evolution of SAR.
ST-Ch-7: Testing robustness and reliability of robotic foundation models.
ST-Ch-8: Ensuring resilience against adversarial threats, such as prompt-based and perception-based attacks.

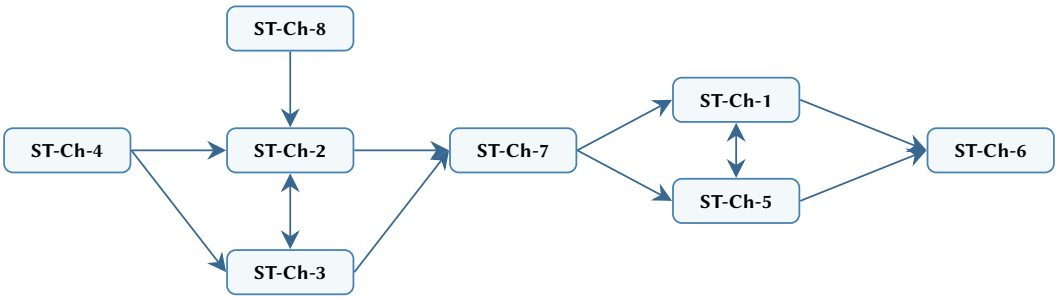


Fig. 8. Dependency graph for challenges in software testing for SAR.

8 Operations for SAR

This section starts by reviewing research contributions in the field of operations for SAR. Next, it presents key open challenges that require further exploration.

8.1 Foundations and Related Research

Many recent works focus on the operation of SARs. These works can be classified by the main artefacts used to trigger and plan adaptations: Models/Data, Non-functional Requirements, and Assurance cases.

Models/Data. Many approaches rely on models and data to monitor the system and its environment and to trigger and plan adaptations. Bencomo et al. [17] provides an extensive survey of 275 papers, providing a taxonomy to classify work along dimensions such as modelled artefacts, runtime model types, purposes (e.g., self-adaptation, assurance), techniques (e.g., model transformation, reflection, reasoning), application domains, and intersecting research areas.

Concrete examples of contributions in this category include Aldrich et al. [6] and Feng et al. [49]. In Aldrich et al. [6], the authors present an approach to making robotic software long-lived and more adaptable by leveraging formal models of software and its environment to enable automated

adaptation. They argue that today's robotic systems require extensive manual effort to handle hardware upgrades, software changes, and environmental variations, which limits their long-term deployment. To address this, they propose intelligent model-based adaptation, where models capture architecture, behaviour, utility, and environment, and are used at runtime to detect when adaptation is needed, explore alternative configurations, and automatically select and enact suitable adaptations. Their work encompasses techniques such as sensitivity analysis for performance/resource trade-offs, state-transition repair using SMT-based solving with human feedback, task-switching policies for dynamic environments, architectural adaptation that integrates ROS-based components, multi-model integration with probabilistic planning, and automatic code repair using tools like Houston.

Similarly, Feng et al. [49] demonstrates how the MAPE-K feedback loop can be integrated into a Digital Twin (DT) to enable self-adaptation in CPS. Using a thermal incubator case study, they designed a DT with services for monitoring, anomaly detection (via a Kalman filter), recalibration, controller optimisation, and reconfiguration. When unexpected conditions occurred (e.g., opening the incubator lid), the DT automatically recalibrated its model and optimised the controller to maintain stable operation, without human intervention.

Non-functional Requirements. Lotz et al. [100] and Brugali et al. [30] represent example works in this category. Lotz et al. [100] proposes a modelling approach to handle dynamic variability in service robots by separating concerns between operational robustness and QoS (quality of service) optimisation. At design-time, they introduce two domain-specific languages: SmartTCL, which models operational variability by defining flexible task coordination strategies that allow robots to react to contingencies at runtime, and VML (Variability Modelling Language), which models variability in QoS by specifying how non-functional properties such as safety, energy consumption, and performance should be optimised under changing conditions. At runtime, these two mechanisms are integrated, enabling SmartTCL to handle functional variability. At the same time, VML resolves trade-offs in non-functional requirements, with different strategies proposed for their safe and consistent coordination.

Brugali et al. [30] proposes a model-driven methodology for designing robotic systems that can dynamically reconfigure themselves at runtime while ensuring QoS guarantees. Their approach builds on three pillars: (1) explicitly modelling runtime variabilities in environment, system resources, and tasks, (2) annotating system variants with QoS-related properties (e.g., execution times, safety constraints), and (3) leveraging UML MARTE and Queuing Network models to evaluate performance and dependability trade-offs.

Assurance cases. These works make use of assurance cases in the SAR adaptation. Cheng et al. [39] introduces AC-ROS, a framework that integrates assurance cases into the MAPE-K feedback loop to manage runtime adaptations in ROS-based autonomous systems. Assurance cases, expressed in Goal Structuring Notation (GSN), provide structured arguments that system requirements are met and are supported by evidence such as tests or monitored data. AC-ROS uses these models at runtime to guide adaptation decisions, ensuring that self-adaptive behaviours continue to satisfy safety and functional requirements.

8.2 Challenges and Open Questions

Due to their self-adaptive nature, SAR systems require continuous monitoring. As argued in [92], legitimation of new plans and system configurations is crucial for continued safe and performant operation.

8.2.1 State Estimation and Model Synchronisation. To achieve this monitoring, several models need to be kept up to date: the environment model, the system model, and models of its subsystems. State

estimation is the preferred technique for maintaining the model states and parameters up to date as the system operates. However, due to the overlap between models (e.g., system models share parameters and states with their subsystem models), it is challenging to ensure consistency across them. This is also an opportunity to develop coupled state estimation techniques that operate on an overlapping ensemble of models.

8.2.2 Managing Uncertainty in SAR Operations. A key challenge in SAR operations is to characterise and manage the uncertainty in the system. This goes beyond maintaining uncertainty measures on model parameters and states (which state estimators already do) to quantify uncertainty at every phase of the MAPLE-K loop. For instance, a planner's optimal solution needs to be accompanied by a measure of confidence in the solution, indicating how robust its optimality is to uncertainties in the system and the models used to quantify it.

8.2.3 Experimentation in Real Environment. During operations, there might not be enough information to accurately identify the relationships between the system and its environment for planning purposes. For instance, a robot that never changes speed will not have a sense of its own inertia. This means that SAR planners may need to actively instruct the managed robot to conduct experiments to gather crucial data about itself or its environment, enabling them to plan effectively. This presents a challenge, as the experiments must be designed to avoid compromising the safety of the system or its environment.

8.2.4 Adaptive Software Updates. Since both the MAPLE-K and SAR require software updates throughout their operations, it is essential to plan and execute these updates without disrupting system operation. Ideally, techniques from dynamic software updating [3] should be applied here, except these need to be adapted from cloud systems to robotic systems. Here, the challenge is that software components may not be replicable or their state may not be so easily transferable to new versions, as these techniques commonly require.

8.2.5 Dev(Sec)Ops Pipelines for SAR Fleets. Current operational practice for SAR often adopts selected DevOps activities (e.g., automated builds, containerised deployment, and remote updates), but typically in a fragmented manner and without a unified Dev(Sec)Ops methodology tailored to heterogeneous robot fleets. Compared to cloud-native systems, SAR deployment pipelines must account for embodiment, intermittent connectivity, heterogeneous hardware/software stacks, and mission/safety constraints during rollout. An open challenge is therefore to define end-to-end pipelines that continuously build, qualify, and secure updates, and then roll them out incrementally (e.g., canary/staged deployment) with robust rollback strategies when runtime evidence indicates elevated risk. This challenge directly intersects with software testing: qualification must combine staged MiL/SiL/HiL evidence, regression testing under continuous evolution, and runtime validation to decide whether a change can be promoted from digital and controlled environments to fleet-wide operation.

8.2.6 Challenges Summary. In the following, we present key challenges related to runtime operations in SAR. The graph in Figure 9 suggests that uncertainty-aware operation (Op-Ch-2) is foundational, since confidence estimates are needed to decide when active uncertainty reduction (Op-Ch-3) or software changes should be triggered. Dynamic software updating (Op-Ch-4) is another prerequisite for continuous Dev(Sec)Ops pipelines (Op-Ch-5), which must safely qualify, roll out, and roll back changes across heterogeneous fleets. In turn, both active uncertainty reduction and pipeline evidence support more dependable overlapping model state estimation (Op-Ch-1) during operation.

Challenges in Operations for SAR

Op-Ch-1: Overlapping model state estimation.
Op-Ch-2: Uncertainty as first-class citizen in MAPLE-K Loops.
Op-Ch-3: Active uncertainty reduction.
Op-Ch-4: Dynamic software updating for SARs.
Op-Ch-5: Continuous Dev(Sec)Ops pipelines for heterogeneous SAR fleets.

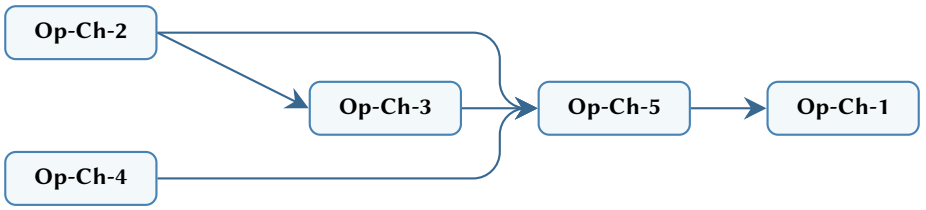


Fig. 9. Dependency graph for challenges in operations for SAR.

9 Key Enabling Technologies

In this section, we provide an overview of two key enabling technologies shaping the future of self-adaptive robotics, namely DTs (Section 9.1) and AI (Section 9.2), along with their associated challenges and opportunities. Several other technologies can play an important role in specific SAR contexts, such as decentralised network and communication systems in multi-robot and swarm robotics; however, these fall outside the scope of this paper. The focus on DTs and AI is motivated by their prominence, growing importance, and broad applicability across robotics in the literature [22, 110]. Although neither is a mandatory prerequisite for SAR, both serve as powerful enablers that substantially enhance how SAR systems are engineered and operated. Their increasing influence on state-of-the-art practices in modelling, validation, optimisation, and runtime adaptation makes them a timely focus for this roadmap.

9.1 Digital Twins

This section starts with a literature overview and then outlines key open challenges.

9.1.1 Foundations and Related Research. Many definitions of Digital Twins (DTs) exist [13]. However, we use the following from [53]. (1) A Digital Twin (DT) is a digital representation of a real-world entity called the Physical Twin (PT). (2) The DT and PT are connected by a communications infrastructure that allows the DT to maintain a known level of fidelity to the PT it represents. (3) A DT offers its stakeholders a range of services that add value to the PT without unduly compromising the PT's operation. This definition on purpose does not distinguish between DTs and Digital Shadows (DSs) (without a direct bi-directional communication) [89] because in reality DTs initially start as DSs where a human is involved. When a satisfactory level of trust in the DT's bidirectional communication is established, autonomy can be enabled. We imagine that this autonomy will be introduced gradually, as is the case with autonomous cars.

DTs are a generic technology that can be applied to any CPS, and naturally, for robots as well [62, 105]. DTs can be used in various domains, offering distinct services. At present, DT is utilised across multiple industries, including manufacturing [142], automotive [19], aviation [160],

maritime [166], construction [159], and healthcare [131, 132, 141], serving diverse functions like simulation, integration, testing, monitoring, and maintenance.

There is growing interest in using DTs in robotics, for example, to improve the design, performance, and maintenance. DTs are applied in various subfields, such as robotics design [87], motion planning and control [162], human-robot interaction [97, 105], autonomous robots [72], smart manufacturing [95], and prognostics and health management (PHM) [69, 138]. Mazumder et al. [110] explored trends in DT-integrated robotics to identify gaps, potential scope, challenges, and future perspectives. Yang et al. [162] proposed a DT-based autonomous navigation and control method for omnidirectional mobile robots (OMRs), achieving a physical-virtual synchronisation tracking error of 0.061 m and effectively handling various tasks. Song et al. [138] introduced a DT-assisted fault diagnosis system for robot joints, using a CycleGAN-based model to map virtual entity data to physical data. Their method increased fault diagnosis accuracy from 32.5% to 98.86% using only 400 measured data sets. The purpose of a DT is essential, as it determines the kind of predictive power its internal models need. One DT can have multiple models (aiming to constantly monitor the state of the PT and make future predictions) inside, and there can be different ways to determine which of these to use in other circumstances.

9.1.2 Challenges and Open Questions. Various publications address the most significant research challenges in DTs [26, 59, 93]. Challenges for DTs and future research directions can be described in different dimensions. However, from a robotics perspective, we think that the most important ones are the following, which deserve attention from the research community.

Data Quality from PT. Data from a physical robot is mainly derived from sensors. Still, it is delivered with different kinds of uncertainties (noise affecting the values and time delay in delivery). Depending on the purpose of a DT, such uncertainties may make it impossible to provide the predictions in a trustworthy manner.

Providing Predictions Promptly. When data from a PT is fed into one of the models inside a DT, it is essential that it is possible to provide predictions to indicate that some action is required promptly. A human user can make an actual decision autonomously if one trusts the predictions. The challenge here is ensuring that the underlying computing infrastructure is enough for the model to make predictions.

State Estimation for the PT. When debugging an application on a computer, one has complete insight into what state the system is in at all times. However, when dealing with physical processes, it is much more challenging to determine the exact moment when a state transition occurs.

Composition of DTs. If we have established DTs for different collaborating robots, the composition of the DTs is far from trivial. One root cause is that the IP embedded in models with predictive power can be essential to the organisations that produce them. Thus, it is paramount to ensure that such IP is protected.

Flexibility and Multi-Purpose Demands. DTs have significant potential to improve lifecycle engineering. For DTs to be effective, their models must adapt across all lifecycle stages with fidelity adjusted to each phase—lower fidelity for design and higher fidelity for control, verification, and operations. However, most current research focuses on specific lifecycle stages, rather than examining the broader value chain [167]. Enhancing the adaptability and flexibility of DT models remains a key challenge.

Limited Complexity and Extensibility. Current DT study in virtual prototyping (VP) [104] often focuses on small-scale modules or isolated systems. However, a typical robot involves a

complex interplay of mechanical, thermal, hydraulic, and control subsystems. The existing literature offers limited insights into the development of large-scale, fully integrated DT systems. The key challenge lies in the comprehensive integration of these subsystems, including sensors, control mechanisms, and mechanical systems.

Dynamic Interaction between DTs and PTs. DTs differ from traditional models through their real-time, dynamic interaction with PTs. Unlike conventional models used in isolated software for tasks like design and control, DTs evolve alongside the systems they represent, integrating live sensor data, simulation updates, and software upgrades throughout the system's life cycle [107]. The complexity of modern robotic systems, with their multidisciplinary nature (e.g., mechanical, electrical, and software), makes it challenging to model all aspects on a single platform. As a result, achieving a fully integrated and responsive representation of real-world systems remains a significant challenge for DTs.

Challenges Summary. In the following, we present a summarised list of challenges in the use of digital twins for SAR. The graph in Figure 10 shows the dependencies among the challenges, indicating their priorities and timelines. From a timing perspective, determining state transitions of PTs is the most important challenge (DT-Ch-3). This is necessary before trustworthy predictions (DT-Ch1), timely predictions (DT-Ch-2), and adaptation (DT-Ch-5) can depend on it. Afterwards, composition (DT-Ch-4) and complex subsystems (DT-Ch-6) will enable more complex DT combinations. Finally, a fully integrated, dynamic and responsive DT (DT-Ch-7) is the ultimate target from a DT research perspective.

Challenges in Digital Twins for SAR

DT-Ch-1: Ensuring trustworthy predictions due to data quality and uncertainties from physical systems.
DT-Ch-2: Requires sufficient computing infrastructure to support timely predictions for decision-making.
DT-Ch-3: Determining the exact moment of state transitions in physical systems.
DT-Ch-4: Composing DTs for collaborating robots.
DT-Ch-5: Adapting DT models across all software engineering lifecycle phases with sufficient fidelity.
DT-Ch-6: Integrating complex subsystems in large-scale robotic DTs.
DT-Ch-7: Achieving a fully integrated, dynamic, and responsive DT representation of multidisciplinary robotic systems.

9.2 Artificial Intelligence

This section first presents an overview of the research and then highlights open challenges.

9.2.1 Foundations and Related Research. AI has become a cornerstone in the evolution of robotics, enabling robots to perform complex tasks with increasing autonomy and adaptability. AI techniques, including machine learning (ML), deep learning (DL), and reinforcement learning (RL), have facilitated advancements in perception, planning, control, and human-robot interaction. AI in this context is characterised by advances in several areas, including computer vision, natural language processing, and reinforcement learning (RL). These technologies have enabled robots to perceive, reason, and act in complex environments with minimal human intervention.

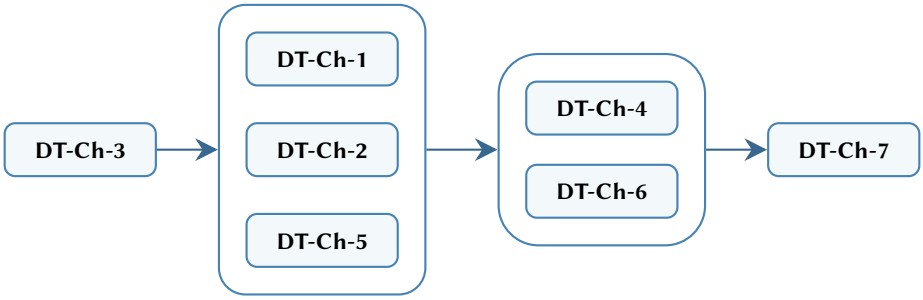


Fig. 10. Dependency graph for challenges in digital twins for SAR.

Perception and Sensing. Modern robotic systems use deep learning-based computer vision techniques to interpret visual data. Convolutional Neural Networks (CNNs) and transformer-based architectures have revolutionised object detection, segmentation, and scene understanding. For example, models such as YOLO [124] and Mask R-CNN [70] are widely used for real-time object detection and instance segmentation in robotics applications. Additionally, multimodal sensing, which combines visual, LiDAR, and radar data, has improved the robustness of perception systems in autonomous vehicles and drones. Furthermore, modern autonomous vehicles employ occupancy networks to reconstruct 3D environments [129] and incorporate flow information to predict object motion [2], further improving situational awareness and navigation.

Decision-Making and Planning. AI has enhanced robotic decision-making through techniques such as RL and LLMs as autonomous agents. RL algorithms, such as Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO), have been widely applied in robotic manipulation [117], autonomous navigation [147], and multi-agent coordination. Moreover, hierarchical RL structures [119] allow robots to decompose complex tasks into manageable subtasks, improving long-term planning and adaptability. LLMs are increasingly integrated into robotics to enable high-level reasoning and task planning. For instance, LLMs can be used as the brain for generating a robot behaviour tree [101].

Control and Actuation. AI has become crucial in robotic control and actuation, enabling precise, adaptive, and efficient movement. One line of research focuses on using AI to learn unmodelled dynamics, including complex aerodynamic effects, vehicle friction models [140], and hydrodynamics [148], resulting in more accurate dynamic models and improved control performance. Another line of research exploits RL to learn controllers directly through simulation, followed by sim-to-real transfer techniques to deploy them on physical robots. This approach has demonstrated the ability to develop robust controllers that enable robots to adapt to varying terrains and dynamic environments.

Human-Robot Interaction. NLP and affective computing have enabled more natural and intuitive interactions between humans and robots. Chatbots and voice assistants powered by transformer models, such as GPT and BERT, are being integrated into service robots for healthcare, education, and customer service. Additionally, emotion recognition systems allow robots to adapt their behaviour based on human emotional states.

9.2.2 Challenges and Open Questions. Despite considerable efforts, AI for software engineering of self-adaptive robots still faces several challenges.

Inherent Challenges Related to Data. Self-adaptation poses specific challenges in AI-powered robotics systems. The most popular AI systems are trained on vast amounts of data. Thus, self-adaptation can usually be considered as gathering more data that represents the new state of the world and the task to which the systems should adapt. This is very expensive or infeasible in many situations.

Overcoming Catastrophic Forgetting in Continual Learning. Another challenge is retaining previously acquired knowledge when adapting to new data, thereby avoiding catastrophic forgetting and promoting smooth, continual learning. Model fine-tuning, transfer learning, knowledge distillation, and RL-based continual training are methods to consider for systems that adapt to new data.

Overfitting. AI often overfits training data, leading to poor generalisation and causing operational anomalies. These should be detected and analysed to provide adaptation plans. More research is needed to address AI overfitting by implementing self-adaptive robots.

AI-specific Challenges. AI in self-adaptive robots presents new challenges, including hallucinations that can lead to unsafe decisions and inherent uncertainty that requires quantification. Real-time handling of uncertainty remains underexplored and needs further study. Regardless of the adaptation method, trustworthiness is key. AI-powered robots must adapt safely, ensuring legitimacy through explainability, fairness, transparency, and robustness. Assuring trustworthiness in self-adaptive robots is an understudied area and requires novel solutions. There are several additional challenges related to human-robot interaction that focus more on human aspects, e.g., developer/user experience, operator trust, organisational policy, transparency, accountability, etc., which are indeed important but are not in the main scope of this paper and will not be discussed in detail.

Challenges Summary. In the following, we present a summary of key challenges in applying AI to SAR. The graph in Figure 11 shows the dependencies among the challenges, indicating their priorities and timelines. From a timing perspective, gathering sufficient data (AI-Ch-1) relevant to the self-adaptation to be performed is the first challenge to be addressed. Retraining or fine-tuning the AI algorithms for adaptation, ensuring that previous knowledge is not severely altered (AI-Ch-2) and at the same time avoiding overfitting (AI-Ch-3), is challenging but necessary for reliable self-adaptation. Finally, the adapted AI models should be trustworthy (AI-Ch-4) without hallucinations or raised ethical concerns.

Challenges in AI for SAR

AI-Ch-1: Gathering sufficient data for self-adaptation in AI-powered robotics is often expensive or infeasible.
AI-Ch-2: Retaining previously acquired knowledge while adapting to new data and avoiding catastrophic forgetting.
AI-Ch-3: Addressing AI overfitting to improve generalisation and prevent operational anomalies in self-adaptive robots.
AI-Ch-4: Ensuring trustworthiness in self-adaptive robots, including handling hallucinations, real-time uncertainty, and ethical concerns.

9.3 ~~Quantum Computing~~

~~This section provides a review of the literature, followed by a discussion of open challenges.~~

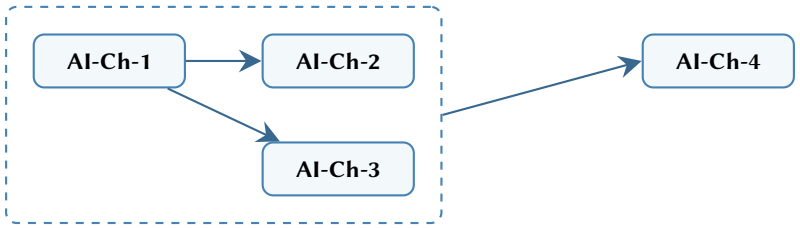


Fig. 11. Dependency graph for challenges in AI for SAR.

9.3.1 Foundations and Related Research. Quantum Computing (QC) uses quantum mechanical principles to perform computations and has the potential to solve many complex problems efficiently. To this end, specialised quantum computers based on various technologies are being built, e.g., by IBM, Google, and IQM. As quantum computers become increasingly powerful, their applications in various domains, including self-adaptive robotics, are also gaining popularity. Concerning software engineering for self-adaptive robotics, there is currently limited work. Recently, Wang et al. [152] developed an approach for quantum-classical machine learning-based testing of self-adaptive robots, demonstrating better performance than a purely classical machine learning approach. Although robots that utilise quantum processors remain a distant prospect, the application of QC to support the software engineering of SARs offers opportunities in the near future.

9.3.2 Challenges and Open Questions. As the use of QC in SAR continues to grow, several open challenges remain to be addressed, as outlined below.

Quantum Optimisation. Quantum optimisation is an active field of research, where algorithms such as the Quantum Approximate Optimisation Algorithm (QAOA) [24] and quantum annealing [163] offer more efficient solutions to certain optimisation problems than their classical counterparts. To this end, numerous research opportunities exist to explore the application of these algorithms in various software engineering activities for SARs, such as test optimisation. Such areas remain largely unexplored; however, with increasing access to commercial quantum computers, we foresee a growing use of quantum optimisation algorithms for SARs by 2030, in areas such as test optimisation to trigger self-adaptation, trajectory planning, test-oracle generation, and optimal coordination among the fleet of SARs.

However, to apply quantum optimisation, many challenges need to be addressed. For instance, it is necessary to determine how to map an optimisation problem in the software engineering of SARs into a format that can be solved by quantum optimisation algorithms (e.g., Quadratic Unconstrained Binary Optimisation or Ising for QAOA). Moreover, current quantum computers have a limited number of qubits and are affected by noise. Therefore, there is a need to efficiently map optimisation problems to maximise the use of small-scale hardware while also mitigating hardware noise. For a more detailed discussion on these challenges, which also apply to SARs, the reader is referred to relevant literature [150].

Quantum Machine Learning. Quantum Machine Learning (QML) offers many research opportunities for the software engineering of SARs. It has the potential to discover patterns in data that may be missed by classical machine learning [38]. To this end, there is a need for further exploration of these algorithms in SARs, as they can help robots perform tasks with greater precision and efficiency, while also supporting software engineering activities such as testing, as recently demonstrated for self-adaptive robots by Wang et al. [152]. The tasks where QML can assist robots include identifying patterns more precisely, even with fewer features and less data. Recent studies [114,

151] have shown that QML can achieve similar or better performance compared to classical methods or QML using larger feature sets. Moreover, QML has the potential to improve optimal decision-making for SARs by processing data efficiently in high-dimensional quantum feature spaces.

Several challenges must be addressed to apply QML to software engineering for SARs successfully. Examples include how to develop efficient quantum features for training, how to encode classical data into the quantum domain, and how to maximise the use of limited hardware while mitigating noise, as in quantum optimisation. Several of these challenges are described in the literature [150] and are also applicable to software engineering for SARs.

Challenges Summary. In the following, we outline the challenges related to quantum computing for SAR.

Challenges in Quantum Computing for SAR

QC-Ch-1: Efficiently mapping optimisation problems in SARs to formulations that quantum optimisation algorithms can solve.
QC-Ch-2: Efficient use of small-scale quantum computers with a limited number of qubits and inherent noise.
QC-Ch-3: Effective methods for quantum feature design as well as encoding data in the quantum domain to be used by quantum machine learning algorithms.
QC-Ch-4: Intuitive frameworks to enable the use of quantum computing solutions with software engineering workflows for self-adaptive robots.

10 2030 Research Agenda

In this section, we begin by analysing the challenges outlined in the previous sections with respect to quality aspects (Section 10.1), aiming to identify key areas that require future attention. We then examine how SAR characteristics are relevant to the challenges outlined in this roadmap (Section 10.2). Next, we analyse the relationships between each enabling technology and the five phases of the software engineering lifecycle (Section 10.3). Finally, we explore potential research opportunities (Section 10.4) to drive advancements in the field of SAR.

10.1 Challenges and Quality Aspects

For all challenges identified in the software engineering lifecycle and enabling technologies, we analysed the quality aspects that they highlight. Specifically, we analysed: (1) the relationships between challenges and quality aspects, (2) the relative criticality of challenges based on their impact across multiple quality aspects (i.e., challenges influencing several aspects are considered more critical), and (3) the overall importance of each quality aspect (i.e., quality aspects requiring further attention). The results of this analysis are presented in Table 1, and a detailed interactive version of the table is available at https://simula-complex.github.io/robomap/.

The analysis was conducted as follows. First, we identified the set of quality aspects from the SAR literature [5] and the ISO/IEC 25010:2023 standard [56]. Next, we examined the description of each challenge to identify which quality aspects were highlighted. For example, the description of *SRE-Ch-1* highlights safety, reliability, and adaptability; these three aspects are therefore marked for that challenge in the table. We then computed the criticality of each challenge by counting the number of quality aspects associated with it (i.e., the number of rows in the table). In the case of *SRE-Ch-1*, this yields a criticality score of 3. A higher criticality count indicates that the corresponding challenge requires greater attention. Finally, to assess the overall importance of each quality aspect, we counted how many challenges reference it (i.e., the number of vertical lines in

the table). A higher value indicates that the quality aspect is widely emphasised across multiple challenges, reflecting its overall significance in the context of SAR.

When examining the criticality of challenges across various quality aspects, it is evident that challenges in MDE and software testing span most of them. This indicates that these challenges require significant attention in future research. Similarly, challenges related to SAR simulations and operations appear to be the second most critical, underscoring their importance for advancing SAR research. Among enabling technologies, the challenges in AI and QC are closely related to multiple quality aspects, indicating that applying these technologies to SAR will require more attention in the future.

In terms of the importance of quality aspects highlighted across the challenges, reliability is the most frequently cited concern, underscoring the need for further research to ensure the dependability of SAR systems throughout the software engineering lifecycle. Adaptability follows as the second most emphasised aspect, suggesting a growing demand for SAR systems that can seamlessly adapt to evolving environments and requirements. Other most highlighted quality aspects are safety and performance, especially in challenges involving AI-driven components, indicating the need for further research. However, privacy and security are less highlighted across challenges, appearing mainly in challenges related to simulations, software development, software testing, AI, and digital twins. Given the growing integration of AI in robotics, the increasing importance of software testing to ensure reliability, and the key role of digital twins, both privacy and security require future attention, especially as future robots are expected to operate in real-world, human-centric environments.

Table 1. Mapping of challenges in self-adaptive robotics to cross-cutting quality aspects. Rows with more "+" symbols indicate higher criticality of the corresponding challenge, while columns with more "+" symbols show higher importance of the associated quality aspect.

Topic	Challenge	Safety	Performance	Reliability	Explainability	Privacy	Adaptability	Security	Ch. Criticality
Requirements Engineering	SRE-Ch-1	+		+			+		**3**
	SRE-Ch-2			+			+		**2**
	SRE-Ch-3	+		+			+		**3**
	SRE-Ch-4	+			+		+		**3**
	SRE-Ch-5				+	+			**2**
	NR-Ch-1	+		+			+		**3**
	NR-Ch-2				+				**1**
Software Design	SD-Ch-1		+				+		**2**
	SD-Ch-2	+	+	+			+		**4**
	SD-Ch-3	+		+			+		**3**
	SD-Ch-4			+			+		**2**
	SD-Ch-5			+			+		**2**
	SD-Ch-6	+			+		+		**3**
	SD-Ch-7	+		+			+		**3**
	MDE-Ch-1			+	+		+		**3**
	MDE-Ch-2		+	+			+		**3**

Topic	Challenge	Safety	Performance	Reliability	Explainability	Privacy	Adaptability	Security	Ch. Criticality
	MDE-Ch-3	+	+	+			+		**4**
	MDE-Ch-4			+	+		+		**3**
	MDE-Ch-5	+			+	+	+		**4**
	Sim-Ch-1			+			+		**2**
	Sim-Ch-2			+			+		**2**
	Sim-Ch-3		+	+			+		**3**
	Sim-Ch-4		+	+					**2**
	Sim-Ch-5					+		+	**2**
Software Development	Dev-Ch-1		+	+			+		**3**
	Dev-Ch-2		+	+			+		**3**
	Dev-Ch-3	+	+	+			+		**4**
	Dev-Ch-4		+	+			+		**3**
	Dev-Ch-5	+				+		+	**3**
Software Testing	ST-Ch-1	+	+	+			+		**4**
	ST-Ch-2	+		+	+				**3**
	ST-Ch-3	+		+			+		**3**
	ST-Ch-4	+	+	+			+		**4**
	ST-Ch-5		+	+			+		**3**
	ST-Ch-6	+		+	+		+		**4**
	ST-Ch-7	+		+	+				**3**
	ST-Ch-8	+		+	+			+	**4**
Operations	Op-Ch-1			+			+		**2**
	Op-Ch-2	+		+	+		+		**4**
	Op-Ch-3	+		+			+		**3**
	Op-Ch-4	+		+			+		**3**
	Op-Ch-5	+		+			+	+	**4**
Digital Twins	DT-Ch-1	+		+	+				**3**
	DT-Ch-2	+	+	+					**3**
	DT-Ch-3			+	+		+		**3**
	DT-Ch-4					+		+	**2**
	DT-Ch-5		+	+			+		**3**
	DT-Ch-6		+	+			+		**3**
	DT-Ch-7		+	+			+		**3**
Artificial Intelligence	AI-Ch-1		+	+			+		**3**
	AI-Ch-2	+		+			+		**3**
	AI-Ch-3	+		+			+		**3**
	AI-Ch-4	+		+	+	+		+	**5**
Quality Aspect Focus		**28**	**18**	**44**	**15**	**6**	**41**	**6**	-

The various types of robotic systems, such as soft robotics, modular robotics, and swarm robotics, may lead to greater complexity. Adaptive and autonomous robots can adapt to changing goals, requirements, and environmental conditions. This can range from simple parameter adjustments to

more complex behavioural changes, such as switching between modes of operation, adapting to new tasks, or modifying their current goals to operate safely and in accordance with their requirements.

Soft robotics encompasses a field of robotics where robots are built using highly flexible materials, allowing them to adapt to their environment and perform tasks that require a high degree of flexibility [164]. This allows them to interact with their environment in a more varied way, whether this regards their movement, their way of grasping or interacting with objects, or their response to changing environmental conditions. This high flexibility, however, leads to a state-space explosion in the robot's potential behaviour. Programming and controlling such robots can therefore become challenging [122].

Modular robots can interact with each other and adapt to new complexities and tasks through self-composition. Specifically, robots are designed to interlock and create new, larger physical structures [165]. While the individual robots are usually very simple shapes, e.g., cubes, they can, through their ability to compose with other robots, take on more complex shapes. This, in turn, may allow them to overcome additional changes in their environment, e.g., by forming a bridge to cross a gap. The ability to self-compose, however, also leads to a combinatorial explosion in the number of possible configurations and shapes that robots can take on, which in turn leads to higher complexity in the programming and control of such robots [63]. This poses significant challenges for the distributed planning and verification of both individual and composed robotic systems [48].

Swarm robotics emphasises the individuality of robots and their ability to operate autonomously while also interacting with their peers and the environment [28]. The core idea in swarm robotics is that collective behaviour emerges from local interactions among robots. As they are tasked with achieving their common goals, they work towards their individual objectives. For this, they coordinate locally and adapt individually to changes in their local environment. Through this emergent behaviour, the swarm can achieve complex tasks that would be difficult or impossible for individual robots to accomplish on their own. While swarm robotics offers higher robustness compared to individual robots due to their redundancy, designing or identifying local rules, guaranteeing their effectiveness and efficiency, and ensuring the overall system's performance remain challenging tasks [35, 85, 120].

10.2 Challenges and SAR Characteristics

Table 2 highlights the relevance of the challenges with the SAR characteristics outlined in Section 2. A detailed interactive version of this table is available at https://simula-complex.github.io/robomap/. Each "+" indicates that the respective characteristic plays a significant role as part of this challenge. In the following, we provide illustrative examples of how SAR characteristics are relevant to the challenges highlighted in this paper.

- **Embodiment:** Robots handling uncertainty during their adaptation are required to have an understanding of their physical shape to adapt in a predictable and trustworthy manner (SD-Ch-4). Another example is real-time sensor fusion, which requires the robot to understand where the diverse sensors are located and how their fusion needs to be adapted based on physical changes of the robot (Dev-Ch-3).
- **Physical Interaction:** As robots interact with their environment, there is a need for combining software with models of hardware and the immediate environment (MDE-Ch-3). When interacting physically with the environment, and specifically with other robots, robots may adapt using AI and online learning to improve their performance. With this, we need to address AI overfitting to improve generalisation and prevent operational anomalies (AI-Ch-3).
- **Localised:** To provide energy-efficient robotic software and sustainable operations, robots need to have an understanding of their local environment and their own embedding within to optimise their

activities and adaptations (Dev-Ch-4). As ethical requirements may vary by location, robots need to understand their environments and localise themselves within the relevant ethical environment (SRE-Ch-5). Similarly, this holds for norm-sensitive decisions (NR-Ch-2).

• **Heterogeneity:** In dynamic, open environments, robotic systems may encounter other robots; coupling these systems in a composite simulation requires them to deal with their respective heterogeneity (Sim-Ch-2). The heterogeneity of SAR fleets is also relevant when defining the underlying Dev(Sec)Ops pipelines (Op-Ch-5).

• **Distributed Nature:** As self-adaptive multi-robot systems and fleets are inherently distributed in the physical environment, this distributed nature needs to be explicitly considered in the algorithms that dynamically adapt to system reconfiguration (Sim-Ch-3). Similarly, when operating fleets of DTs for collaborative robots, the distributed nature needs to be taken into account (DT-Ch-4).

• **Modularity:** As robots are composed of various hardware that might change during runtime (e.g., changing the gripper on a robotic arm), this modularity should be a fundamental aspect to avoid rebuilding existing architectures (Dev-Ch-2). With modular SAR, MAPE-K components might be available in different modules. With this, we have to ensure the correctness of individual MAPE-K components and the overall loop during testing (ST-Ch-4).

Table 2. Mapping of challenges in self-adaptive robotics to SAR characteristics. Rows with more "+" symbols indicate the relevance of the characteristic to the challenge.

Topic	Challenge	Embodiment	Physical Interaction	Localised	Heterogeneity	Distributed Nature	Modularity
Requirements Engineering	SRE-Ch-1		+		+	+	+
	SRE-Ch-2	+			+	+	
	SRE-Ch-3					+	
	SRE-Ch-4	+	+				
	SRE-Ch-5	+		+			
	NR-Ch-1	+	+		+		+
	NR-Ch-2			+	+	+	+
Software Design	SD-Ch-1					+	+
	SD-Ch-2	+	+		+		
	SD-Ch-3	+	+		+		
	SD-Ch-4	+	+	+			
	SD-Ch-5				+		+
	SD-Ch-6	+	+	+			
	SD-Ch-7		+	+	+		+
	MDE-Ch-1	+		+	+	+	+
	MDE-Ch-2	+		+	+		+
	MDE-Ch-3		+	+	+		
	MDE-Ch-4	+			+		+
	MDE-Ch-5	+	+	+			
	Sim-Ch-1	+	+	+			

Topic	Challenge	Embodiment	Physical Interaction	Localised	Heterogeneity	Distributed Nature	Modularity
	Sim-Ch-2		+		+	+	+
	Sim-Ch-3			+	+	+	
	Sim-Ch-4				+		+
	Sim-Ch-5	+					
Software Development	Dev-Ch-1	+					+
	Dev-Ch-2	+					+
	Dev-Ch-3	+	+	+			
	Dev-Ch-4	+		+	+	+	
	Dev-Ch-5	+	+	+			
Software Testing	ST-Ch-1	+	+				
	ST-Ch-2				+		+
	ST-Ch-3	+	+				
	ST-Ch-4	+		+	+		+
	ST-Ch-5		+		+	+	
	ST-Ch-6	+			+		
	ST-Ch-7	+	+		+		+
	ST-Ch-8	+					
Operations	Op-Ch-1	+			+		+
	Op-Ch-2	+	+	+			
	Op-Ch-3	+	+	+	+	+	+
	Op-Ch-4	+			+		+
	Op-Ch-5	+			+	+	
Digital Twins	DT-Ch-1	+	+	+	+		
	DT-Ch-2		+	+		+	
	DT-Ch-3		+				
	DT-Ch-4				+	+	
	DT-Ch-5	+			+		
	DT-Ch-6				+	+	+
	DT-Ch-7				+	+	+
Artificial Intelligence	AI-Ch-1	+	+	+			
	AI-Ch-2	+	+				
	AI-Ch-3	+	+				
	AI-Ch-4	+	+				

10.3 Enabling Technologies across SAR Software Engineering Lifecycle

Table 3 presents the relationships between each enabling technology and the five phases of the software engineering lifecycle for SAR. As shown in the table, DTs and AI play key roles across

Table 3. Relation between enabling technologies and the software engineering lifecycle phases in the context of SAR

	Digital Twins (DTs)	Artificial Intelligence (AI)
Requirements Engineering	Validate requirements in virtual environments before physical deployment.	Support automated requirements elicitation, conflict detection, and adaptive requirement prioritisation.
Software Design	Enable virtual prototyping and architecture validation through high-fidelity twins.	Support design space exploration, trade-off analysis, and synthesis of adaptive control and decision policies.
Software Development	Provide real-time feedback to validate new features in the twin before deployment.	Automate code generation, self-adaptive logic, and static-analysis support.
Software Testing	Enable safe and cost-effective hardware-/software-in-the-loop testing in realistic twin scenarios.	Support automated test generation and prioritisation, fault localisation, adaptive test execution, and test oracle inference.
Operations	Enable continuous runtime monitoring, predictive maintenance, and what-if analysis of adaptation strategies in the twin.	Facilitate runtime adaptation, predictive analytics, online anomaly and fault detection.

these phases. DTs, in particular, provide high-fidelity, executable models of the robotic system and its environment that can be used at multiple phases of the lifecycle. In *Requirements Engineering*, DTs enable engineers to validate requirements across various operational contexts (e.g., different layouts, environmental conditions, or human–robot interaction patterns) and verify that the requirements remain coherent under these conditions. During *Software Design*, they support virtual prototyping and architecture validation through high-fidelity twin simulations, allowing engineers to explore and evaluate different design configurations and refine system architectures. As development progresses into Software Development, Testing, and Operations, DTs continue to provide substantial value. In *Software Development*, they provide real-time feedback that guides implementation decisions, validates new features against the intended system behaviour, and helps developers iteratively refine adaptive logic. For *Software Testing*, DTs enable safe and cost-effective testing in hardware- and software-in-the-loop setups, allowing engineers to expose the system to a wide range of scenarios, including edge cases and hazardous conditions, that would be difficult or dangerous to replicate in the physical world. Finally, in *Operations*, DTs support runtime monitoring, predictive maintenance, and what-if analysis of adaptation strategies by maintaining an up-to-date virtual model of the system. This allows operators to anticipate failures and trigger adaptive responses proactively, a capability recently demonstrated in [80].

Artificial intelligence contributes more pervasively across all phases of the software engineering lifecycle for SAR. In *Requirements Engineering*, AI techniques (e.g., natural language processing and LLMs) can support automated requirements elicitation, the detection of conflicts and inconsistencies in requirements, and the prioritisation of requirements based on risk or stakeholder impact. In *Software Design*, AI enables intelligent design space exploration, trade-off analysis, and the synthesis and evaluation of adaptive control and decision policies, helping engineers navigate the complexity of designing systems that must balance quality aspects such as safety, reliability, and performance. During *Software Development*, AI techniques assist with automated code generation and completion,

the implementation of self-adaptive logic, and support for static analysis (e.g., detecting potential defects or security weaknesses). In *Software Testing*, AI is increasingly used to generate and prioritise test cases, to perform intelligent fault detection and localisation, to develop adaptive test execution strategies, and to infer test oracles from specifications or historical traces. During *Operations*, AI serves as the cognitive backbone of self-adaptive robotic systems, enabling autonomous decision-making, real-time adaptation, predictive analytics, and anomaly detection in dynamic, unpredictable environments. This enables robotic systems to respond intelligently to unforeseen changes without human intervention.

Looking ahead to 2030, we expect both DTs and AI to become even more deeply integrated into the software engineering lifecycle for SAR. DTs are likely to evolve into more intelligent, semantically rich, and real-time synchronised models, capable not only of reflecting physical systems but also of predicting future states and autonomously triggering adaptive responses. AI, and in particular LLMs and foundation models specialised for robotics and software engineering, will increasingly support end-to-end lifecycle activities, from interactive requirements elicitation and automated design-space exploration, through AI-assisted implementation and test generation, to autonomous runtime adaptation and fleet-level optimisation in operations. Together, these technologies are likely to give rise to a new generation of self-learning robotic systems that can continuously refine their behaviour, anticipate environmental changes, and self-optimise across the entire lifecycle, ultimately bringing SAR closer to full autonomy in safety-critical real-world applications.

10.4 Research Opportunities

Given the numerous challenges currently being addressed across various fields of self-adaptive robotics, we aim to identify and outline future research directions and long-term challenges. An overview of these opportunities is given in the green box below.

Within **Requirements Engineering** (O1–O2), there is a need for incorporating insights from social sciences and humanities to improve the interaction of humans and robots. Specifically, we need to include social and ethical norms and enable robots to identify, understand, and reason on them. For example, robots might be trusted more easily in some societies than in others. In response, SARs need to adjust their behaviour according to their social environment. With this, not only will the safety of interactions with robots improve, but it can also lead to better performance of the robot in its required tasks.

In **Software Design** (O3–O4), we need to develop scalable, modular software architectures that enable safe runtime reconfiguration. This includes the ability to dynamically compose and recompose software modules based on the current context and requirements. This, in itself, creates several challenges, such as ensuring safe reconfiguration, consistency, and minimal disruption, while also providing semantic information for each software module to enable greater autonomy in the reconfiguration process. Furthermore, we need to advance hybrid MDE and simulation frameworks. These frameworks need to provide simple approaches to test various types of AI, software components, and hardware models. They further need to provide adaptive co-simulation to enable more accurate and efficient design and testing of SARs.

Research Opportunities

Requirements Engineering

O1: Interdisciplinary research combining safety and human behaviour.
O2: Incorporating normative requirements in SARs.

Software Design

O3: Develop scalable and modular software architectures that ensure safe runtime reconfiguration.
O4: Advance hybrid MDE and simulation frameworks incorporating AI, hardware models, and adaptive co-simulation.

Software Development

O5: Efficient integration of heterogeneous and evolving hardware and software components.
O6: Large language models for real-time sensor fusion to advance self-adaptation.
O7: Energy-efficient AI-driven robotic software development.

Software Testing

O8: Address reality gap in simulation-based testing of SARs.
O9: Advance testing techniques to address uncertainty, nondeterminism, and safety risks in AI components embodied in SARs.
O10: Advance security testing to strengthen SARs' resilience against emerging threats.

Operations

O11: Uncertainty management across all phases of the MAPLE-K loop.
O12: Dynamic deployment and maintenance through continuous Dev(Sec)Ops pipelines.

Digital Twins

O13: Enabling IP to be protected in DTs supporting multiple legal entities.
O14: Develop scalable and adaptable DT architectures to support integration of complex robotic subsystems.

~~Quantum Computing~~

O17: ~~Explore quantum optimisation algorithms for software engineering of SARs to enhance self-adaptation capabilities.~~
O18: ~~Quantum machine learning to improve perception, decision-making, and task precision in SARs.~~

In **Software Development** (O5–O7), we need to efficiently integrate heterogeneous hardware and software components and enable the evolution of these components. This includes the ability to seamlessly integrate new and replace existing sensors, actuators, and software modules into

existing robotic systems. This requires standardised interfaces, protocols, and middleware capable of handling the complexity and diversity of robotic systems. Moreover, foundation models are required for robotic applications. This needs to capture all aspects of robotics, including real-time sensor fusion, which enables self-adaptation. These models should be capable of processing and integrating data from multiple sensors in real time, thereby reducing development effort. This, in turn, will provide the robots with a better understanding of the context and environment, advancing their self-adaptive capabilities. Additionally, we need to focus on energy-efficient AI-driven robotic software development. This includes developing algorithms and techniques that run efficiently on resource-constrained robotic platforms while maintaining high performance and adaptability.

In **Software Testing** (O8–O10), future research over the coming years needs to address the reality gap in SAR simulation. This includes developing techniques to bridge the gap between simulated environments and real-world scenarios, ensuring that tests conducted in simulation accurately reflect robot behaviour under real-world conditions. This can be achieved through techniques such as domain randomisation, transfer learning, and sim-to-real adaptation. Furthermore, we need to advance testing techniques to address uncertainty, nondeterminism, and safety risks in AI components embodied in SARs. This includes developing methods to systematically and rigorously test and validate AI algorithms under various conditions. While exhaustive testing may not be possible, providing guarantees and verification techniques can help ensure the reliability and safety of SARs in dynamic and uncertain environments. Additionally, we need to advance security testing to strengthen SARs' resilience against emerging threats. This includes developing techniques to identify and mitigate vulnerabilities in robotic systems, ensuring their integrity and availability in the face of cyber-attacks.

When it comes to **Operations** (O11–O12) of SARs, we will need to manage uncertainty within all stages of the MAPE-K loop. This includes developing techniques to quantify, model, and mitigate uncertainty in the perception, decision-making, and adaptation processes of SARs. This will enable robots to make more informed decisions and adapt more effectively to changing environments and requirements. Additionally, we need to focus on dynamic deployment and maintenance through continuous Dev(Sec)Ops pipelines for heterogeneous fleets. This includes developing techniques to continuously build, qualify, secure, and deploy updates in real-time while preserving safety and availability. Such pipelines should support staged and canary rollouts, runtime monitoring, and dependable rollback strategies, and they should explicitly integrate evidence from simulation/testing stages (e.g., MiL/SiL/HiL and regression testing under continuous evolution) before fleet-wide promotion.

In the area of **Digital Twins** (O13–O14), we need to enable IP protection in DTs, particularly when collaborating across multiple legal entities. This includes developing techniques to ensure the confidentiality, integrity, and availability of data and models used in DTs, while also allowing collaboration and sharing among different stakeholders. Additionally, we need to develop scalable and adaptable DT architectures to support the integration of complex robotic subsystems. This includes developing modular, flexible architectures that accommodate the diverse and evolving needs of SARs, enabling seamless integration and interaction among components and systems. Additionally, we must consider the different fidelities of DTs for various use cases and situations, e.g., high-fidelity DTs for highly critical situations and low-fidelity DTs for safe states of the SAR operation. This will require work to generate these DTs, potentially on the fly, as well as using and switching between them during SAR operation.

Finally, in the field of **Artificial Intelligence** (O15–O16), we need to develop foundational models that capture the underlying physical principles and dynamics of robotic systems, enabling more accurate and efficient perception, decision-making, and control. These models should be capable of generalising across different tasks and environments, reducing the need for task-specific

training and adaptation. Furthermore, we need to improve autonomous dynamic training and online learning. This includes developing techniques that enable robots to autonomously explore their environment and acquire new skills and knowledge through interaction with the same. Generating the scenario and executing it safely, without damaging or endangering the SAR and others, is crucial.

Finally, we need to develop techniques to exploit the unique properties of **Quantum Computing** (O17–O18), such as superposition and entanglement, to solve complex optimisation problems in robotics more efficiently than classical methods. Along the same lines, we need to develop quantum machine learning algorithms that solve complex tasks and process large amounts of data more efficiently than classical methods. This will enable robots to learn from their experiences and adapt their behaviour more effectively, improving their performance and adaptability in dynamic and uncertain environments.

11 Related Works

Several efforts have been made to outline future roadmaps for robotics software engineering. García et al. [60] presented research challenges and future directions based on an empirical evaluation with industry practitioners from the service robotics domain. The main differences are: (i) our work is more comprehensive without targeting any specific robotics domain, and (ii) our focus is on self-adaptive robotics. Goues et al. [67] presented a roadmap for robotics software engineering based on a discussion with experts from industry and academia. Our work differs in that it focuses specifically on self-adaptive robotics and presents a roadmap based on a detailed review of state-of-the-art research across key phases of the software engineering lifecycle. while also covering enabling technologies such as quantum computing that require further attention.

For self-adaptive systems, De Lemos et al. [41] outlined a roadmap for their software development lifecycle. Weyns et al. [153] presented challenges and future directions for uncertainty in self-adaptive systems. Recently, Li et al. [96] surveyed the research potential and challenges of large language models for self-adaptive systems. Although these works generally cover self-adaptive systems, we mainly focus on self-adaptive robots. Brugali et al. [31] outlined open challenges and future directions across four aspects of robotic software engineering: domain expertise and architecture, verification and validation, prototype-to-product transition, and trustworthiness. Our work differs in several key aspects: (i) it focuses on current challenges and future opportunities specific to self-adaptive robotics, (ii) structures the roadmap across software engineering phases, spanning from requirements to operations, (iii) provides a detailed analysis of enabling technologies like DTs, and (iv) examines critical quality aspects essential for self-adaptive robotics.

Some related systematic reviews also exist. Bozhinoski et al. [27] presented a review of available software engineering solutions for mobile robot safety. Similarly, Casalaro et al. [36] presented a systematic review of MDE for mobile robots. These works primarily focus on literature reviews on mobile robots or on specific aspects, such as MDE. In comparison, we present a forward-looking roadmap for self-adaptive robotics, identifying open challenges and future research opportunities through a comprehensive review of the existing literature, and covering key phases of the software engineering lifecycle and enabling technologies such as AI and quantum computing.

12 Conclusion

Self-adaptive robotics poses fundamental software engineering challenges across the lifecycle, from evolving requirements to runtime operations. Addressing these challenges requires rigorous methods that integrate runtime verification with continuous learning, and development frameworks that combine model-driven engineering, AI-based decision-making, and formal verification. Promising directions include co-simulation and digital twins, which can close the reality gap by

enabling safe and efficient testing and validation of adaptive behaviours before deployment. A central open problem remains how to ensure trustworthiness by capturing safety, reliability, explainability, and ethical accountability when data and uncertainty drive adaptation. Looking toward 2030, progress will depend on combining advances in adaptive architectures, assurance techniques, and enabling technologies such as AI and digital twins. By consolidating these perspectives into a unified roadmap, this paper establishes a structured foundation for the next generation of self-adaptive robotic systems, enabling them to achieve trustworthy autonomy in critical domains such as healthcare, manufacturing, and autonomous transportation.

Acknowledgments

This research is supported by the RoboSAPIENS project, funded through the European Commission Horizon Europe program under Grant Agreement No. 101133807.

References

[1] Jubril Gbolahan Adigun, Tom Philip Huck, Matteo Camilli, and Michael Felderer. 2023. Risk-driven Online Testing and Test Case Diversity Analysis for ML-enabled Critical Systems. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, New York, NY, USA, 344–354. doi:10.1109/ISSRE59848.2023.00017

[2] Ben Agro, Quinlan Sykora, Sergio Casas, and Raquel Urtasun. 2023. Implicit occupancy flow fields for perception and prediction in self-driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York, NY, USA, 1379–1388. doi:10.1109/CVPR52729.2023.00139

[3] Babiker Hussien Ahmed, Sai Peck Lee, Moon Ting Su, and Abubakar Zakari. 2020. Dynamic software updating: a systematic mapping study. *IET Software* 14, 5 (2020), 468–481. doi:10.1049/iet-sen.2019.0201

[4] Elvin Alberts. 2023. Development and integration of self-adaptation strategies for robotics software. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, New York, NY, USA, 131–136. doi:10.1109/ICSA-C57050.2023.00038

[5] Elvin Alberts, Ilias Gerostathopoulos, Ivano Malavolta, Carlos Hernández Corbato, and Patricia Lago. 2025. Software architecture-based self-adaptation in robotics. *Journal of Systems and Software* 219 (2025), 112258. doi:10.1016/j.jss.2024.112258

[6] Jonathan Aldrich, David Garlan, Christian Kästner, Claire Le Goues, Anahita Mohseni-Kabir, Ivan Ruchkin, Selva Samuel, Bradley Schmerl, Christopher Steven Timperley, Manuela Veloso, et al. 2019. Model-based adaptation for robotics software. *IEEE software* 36, 2 (2019), 83–90. doi:10.1109/MS.2018.2885058

[7] Constanza Alfieri, Paola Inverardi, Patrizio Migliarini, and Massimiliano Palmiero. 2022. Exosoul: Ethical Profiling in the Digital World. In *HHAI 2022: Augmenting Human Intellect (Frontiers in Artificial Intelligence and Applications, Vol. 354)*, Stefan Schlobach, María Pérez-Ortiz, and Mel Tielman (Eds.). IOS Press, Amsterdam, Netherlands, 128–142. doi:10.3233/FAIA220194

[8] Xing An, Celimuge Wu, Yangfei Lin, Min Lin, Tsutomu Yoshinaga, and Yusheng Ji. 2023. Multi-Robot Systems and Cooperative Object Transport: Communications, Platforms, and Challenges. *IEEE Open Journal of the Computer Society* 4 (2023), 23–36. doi:10.1109/OJCS.2023.3238324

[9] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. 2017. Formal Design and Verification of Self-Adaptive Systems with Decentralized Control. *ACM Transactions on Autonomous and Adaptive Systems* 11, 4 (Jan. 2017), 1–35. doi:10.1145/3019598

[10] M. Askarpour, L. Lestingi, S. Longoni, N. Iannacci, M. Rossi, and F. Vicentini. 2021. Formally-based Model-Driven Development of Collaborative Robotic Applications. *Journal of Intelligent & Robotic Systems* 102, 3 (2021), 59.

[11] Xiaoan Bao, Xiance Sun, Ning Gui, Na Zhang, Hui Lin, and Shuhan Liu. 2015. Architecture Model to Improve the Development of Robotics Online Reconfiguration. *International Journal of Control and Automation* 8, 1 (2015), 69–82. doi:10.14257/ijca.2015.8.1.06

[12] Chloe M. Barnes, Lukas Esterle, and John N. A. Brown. 2019. "When you Believe in Things that you don't Understand": the Effect of Cross-Generational Habits on Self-Improving System Integration. In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. IEEE, New York, NY, USA, 28–31. doi:10.1109/FAS-W.2019.00020

[13] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. 2019. A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. *IEEE Access* 7 (2019), 167653–167671. doi:10.1109/ACCESS.2019.2953499

[14] Fernando Barros. 2024. π HyFlow: formalism, semantics, and applications. *Discrete Event Dynamic Systems* 34, 1 (2024), 95–124. doi:10.1007/s10626-023-00390-y

[15] J. Bastian, C. Clauß, S. Wolf, and P. Schneider. 2011. Master for Co-Simulation using FMI. In *Modelica Conf.* Fraunhofer IIS / EAS, Dresden, Germany, 1–6.

[16] James Baxter, Bert Van Acker, Morten Kristensen, Thomas Wright, Ana Cavalcanti, and Cláudio Gomes. 2025. Formal Architectural Patterns for Adaptive Robotic Software. In *International Conference on Fundamental Approaches to Software Engineering*. Springer Nature, Switzerland Cham, 145–165. doi:10.1007/978-3-031-90900-9_8

[17] Nelly Bencomo, Sebastian Götz, and Hui Song. 2019. Models@run.time: a guided tour of the state of the art and research challenges. *Software & Systems Modeling* 18, 5 (2019), 3049–3082. doi:10.1007/s10270-018-00712-x

[18] Joakim Schack Betzer, Jalil Boudjadar, Mirgita Frasheri, and Prasad Talasila. 2024. Digital Twin Enabled Runtime Verification for Autonomous Mobile Robots under Uncertainty. In *2024 28th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, New York, NY, USA, 10–17. doi:10.1109/DS-RT62209.2024.00012

[19] G. Bhatti, H. Mohan, and R. R. Singh. 2021. Towards the future of smart electric vehicles: Digital twin technology. *Renewable and Sustainable Energy Reviews* 141 (2021), 110801.

[20] Hridoy Bhuiyan, Franco Olivieri, Guido Governatori, Md. Belal Islam, Alan Bond, and Andry Rakotonirainy. 2019. A methodology for encoding regulatory rules. In *Proceedings of the 4th International Workshop on Mining and Reasoning with Legal texts: co-located with the 32nd International Conference on Legal Knowledge and Information Systems (JURIX 2019)*. Sun SITE Central Europe (CEUR), Aachen, 1–13. https://eprints.qut.edu.au/236378/

[21] Raheleh Biglari and Joachim Denil. 2022. Model validity and tolerance quantification for real-time adaptive approximation. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (MODELS '22)*. ACM, New York, NY, USA, 668–676. doi:10.1145/3550356.3561604

[22] Aude Billard, Alin Albu-Schaeffer, Michael Beetz, Wolfram Burgard, Peter Corke, Matei Ciocarlie, Ravinder Dahiya, Danica Kragic, Ken Goldberg, Yukie Nagai, et al. 2025. A roadmap for AI in robotics. *Nature Machine Intelligence* 7, 6 (2025), 818–824. doi:10.1038/s42256-025-01050-6

[23] María Julia Blas, Horacio Leone, and Silvio Gonnet. 2022. DEVS-based formalism for the modeling of routing processes. *Software and Systems Modeling* 21, 3 (2022), 1179–1208. doi:10.1007/s10270-021-00928-4

[24] Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. 2024. A review on Quantum Approximate Optimization Algorithm and its variants. *Physics Reports* 1068 (2024), 1–66. doi:10.1016/j.physrep.2024.03.002

[25] Torsten Blockwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. 2012. Functional Mockup Interface 2.0: The Standard for Tool-independent Exchange of Simulation Models. In *Proceedings of the 9th International MODELICA Conference, September 3-5, 2012, Munich, Germany (MODELLICA)*. Linköping University Electronic Press, Linköping, Sweden, 173–184. doi:10.3384/ecp12076173

[26] Diego M. Botín-Sanabria, Adriana-Simona Mihaita, Rodrigo E. Peimbert-García, Mauricio A. Ramírez-Moreno, Ricardo A. Ramírez-Mendoza, and Jorge de J. Lozoya-Santos. 2022. Digital Twin Technology Challenges and Applications: A Comprehensive Review. *Remote Sensing* 14, 6 (2022), 1335. doi:10.3390/rs14061335

[27] Darko Bozhinoski, Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, and Ivica Crnkovic. 2019. Safety for mobile robotic systems: A systematic mapping study from a software engineering perspective. *Journal of Systems and Software* 151 (2019), 150–179. doi:10.1016/j.jss.2019.02.021

[28] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. 2013. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence* 7, 1 (2013), 1–41. doi:10.1007/s11721-012-0075-2

[29] Paul A. Bremner, Louise A. Dennis, Michael Fisher, and Alan F. T. Winfield. 2019. On Proactive, Transparent, and Verifiable Ethical Reasoning for Robots. *Proc. IEEE* 107, 3 (2019), 541–561. doi:10.1109/JPROC.2019.2898267

[30] Davide Brugali, Rafael Capilla, Raffaela Mirandola, and Catia Trubiani. 2018. Model-based development of QoS-aware reconfigurable autonomous robotic systems. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*. IEEE, New York, NY, USA, 129–136. doi:10.1109/IRC.2018.00027

[31] Davide Brugali, Ana Cavalcanti, Nico Hochgeschwender, Patrizio Pelliccione, and Luciana Rebelo. 2024. Future Directions in Software Engineering for Autonomous Robots: An Agenda for Trustworthiness [Opinion]. *IEEE Robotics & Automation Magazine* 31, 3 (2024), 186–204. doi:10.1109/MRA.2024.3417089

[32] Davide Brugali and Azamat Shakhimardanov. 2010. Component-based robotics. *IEEE Robotics & Automation Magazine* 17, 1 (2010), 99–111. doi:10.1109/MRA.2010.936952

[33] Mingyu Cai, Mohammadhosein Hasanbeig, Shaoping Xiao, Alessandro Abate, and Zhen Kan. 2021. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE robotics and automation letters* 6, 4 (2021), 7973–7980. doi:10.1109/LRA.2021.3101544

[34] Ricardo Caldas, Juan Antonio Piñera García, Matei Schiopu, Patrizio Pelliccione, Genaína Rodrigues, and Thorsten Berger. 2024. Runtime Verification and Field-Based Testing for ROS-Based Robotic Systems. *IEEE Transactions on Software Engineering* 50, 10 (2024), 2544–2567. doi:10.1109/TSE.2024.3444697

[35] Roberto Casadei, Gianluca Aguzzi, Giorgio Audrito, Ferruccio Damiani, Danilo Pianini, Giordano Scarso, Gianluca Torta, and Mirko Viroli. 2025. Software Engineering for Collective Cyber-Physical Ecosystems. *ACM Trans. Softw. Eng. Methodol.* 34, 5, Article 153 (May 2025), 40 pages. doi:10.1145/3712004

[36] Giuseppina Lucia Casalaro, Giulio Cattivera, Federico Ciccozzi, Ivano Malavolta, Andreas Wortmann, and Patrizio Pelliccione. 2022. Model-driven engineering for mobile robotic systems: a systematic mapping study. *Software and Systems Modeling* 21, 1 (2022), 19–49. doi:10.1007/s10270-021-00908-8

[37] Ana Cavalcanti, Will Barnett, James Baxter, Gustavo Carvalho, Madiel Conserva Filho, Alvaro Miyazawa, Pedro Ribeiro, and Augusto Sampaio. 2021. RoboStar technology: a roboticist's toolbox for combined proof, simulation, and testing. *Software Engineering for Robotics* (2021), 249–293. doi:10.1007/978-3-030-66494-7_9

[38] M. Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J. Coles. 2022. Challenges and opportunities in quantum machine learning. *Nature Computational Science* 2, 9 (01 Sep 2022), 567–576. doi:10.1038/s43588-022-00311-3

[39] Betty HC Cheng, Robert Jared Clark, Jonathon Emil Fleck, Michael Austin Langford, and Philip K McKinley. 2020. AC-ROS: Assurance case driven adaptation for the robot operating system. In *Proceedings of the 23rd ACM/IEEE International Conference on Model-Driven Engineering Languages and Systems*. Association for Computing Machinery, New York, NY, USA, 102–113. doi:10.1145/3365438.3410952

[40] Jingjun Cheng, Zhen Wang, Xiangmo Zhao, Zhigang Xu, Ming Ding, and Kazuya Takeda. 2024. A Survey on Testbench-Based Vehicle-in-the-Loop Simulation Testing for Autonomous Vehicles: Architecture, Principle, and Equipment. *Advanced Intelligent Systems* 6, 6 (2024), 2300778. doi:10.1002/aisy.202300778

[41] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, Vol. 7475. Springer, Berlin, Heidelberg, 1–32. doi:10.1007/978-3-642-35813-5_1

[42] Louise Dennis, Michael Fisher, Marija Slavkovik, and Matt Webster. 2016. Formal verification of ethical choices in autonomous systems. *Robotics and Autonomous Systems* 77 (2016), 1–14. doi:10.1016/j.robot.2015.11.012

[43] S. Dhouib, S. Kchir, S. Stinckwich, T. Ziadi, and M. Ziane. 2012. RobotML, a Domain-Specific Language to Design, Simulate and Deploy Robotic Applications. In *SIMPAR 2012*. Springer, Berlin, Heidelberg, 149–160.

[44] George Edwards, Joshua Garcia, Hossein Tajalli, Daniel Popescu, Nenad Medvidovic, Gaurav Sukhatme, and Brad Petrus. 2009. Architecture-driven self-adaptation and self-management in robotics systems. In *2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, New York, NY, USA, 142–151. doi:10.1109/SEAMS.2009.5069083

[45] Henrik Ejersbo, Kenneth Lausdahl, Mirgita Frasheri, and Lukas Esterle. 2023. Dynamic runtime integration of new models in digital twins. In *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, New York, NY, USA, 44–55. doi:10.1109/SEAMS59076.2023.00016

[46] Lukas Esterle and Radu Grosu. 2016. Cyber-physical systems: challenge of the 21st century. *e & i Elektrotechnik und Informationstechnik* 133, 7 (2016), 299–303. doi:10.1007/s00502-016-0426-6

[47] Lukas Esterle and David W. King. 2022. Loosening Control—A Hybrid Approach to Controlling Heterogeneous Swarms. *ACM Transactions on Autonomous and Adaptive Systems* 16, 2, Article 5 (March 2022), 26 pages. doi:10.1145/3502725

[48] Lukas Esterle, Barry Porter, and Jim Woodcock. 2021. Verification and Uncertainties in Self-integrating System. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. IEEE, New York, NY, USA, 220–225. doi:10.1109/ACSOS-C52956.2021.00050

[49] Hao Feng, Claudio Gomes, Santiago Gil, Peter H. Mikkelsen, Daniella Tola, Peter Gorm Larsen, and Michael Sandberg. 2022. Integration Of The Mape-K Loop In Digital Twins. In *2022 Annual Modeling and Simulation Conference (ANNSIM)*. IEEE, New York, NY, USA, 102–113. doi:10.23919/annsim55834.2022.9859489

[50] Hao Feng, Cláudio Gomes, Michael Sandberg, Casper Thule, Kenneth Lausdahl, and Peter Gorm Larsen. 2021. Developing a physical and digital twin: an example process model. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, New York, NY, USA, 286–295. doi:10.1109/MODELS-C53483.2021.00050

[51] N. Feng, L. Marsso, S. G. Yaman, Y. Baatartogtokh, R. Ayad, V. O. D. Mello, B. Townsend, I. Standen, I. Stefanakos, C. Imrie, G. N. Rodrigues, A. L. C. Cavalcanti, R. Calinescu, and M. Chechik. 2024. Analyzing and Debugging Normative Requirements via Satisfiability Checking. In *IEEE/ACM 46th International Conference on Software Engineering*. Association for Computing Machinery, New York, NY, USA, Article 214, 12 pages. doi:10.1145/3597503.3639093

[52] N. Feng, L. Marsso, S. G. Yaman, B. Townsend, A. L. C. Cavalcanti, R. Calinescu, and M. Chechik. 2023. Towards a Formal Framework for Normative Requirements Elicitation. In *Automated Software Engineering (Conference Publishing Services)*. IEEE, New York, NY, USA, 1776–1780.

[53] John Fitzgerald, Cláudio Gomes, and Peter Gorm Larsen (Eds.). 2024. *The Engineering of Digital Twins.* Springer, Switzerland.

[54] John Fitzgerald, Peter G. Larsen, and Marcel Verhoef. 2014. *Collaborative Design for Embedded Systems: Co-modelling and Co-simulation.* Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-54118-6

[55] Jessica Fjeld, Nele Achten, Hannah Hilligoss, Adam Nagy, and Madhulika Srikumar. 2020. *Principled Artificial Intelligence: Mapping Consensus in Ethical and Rights-based Approaches to Principles for AI.* Technical Report 2020-1. Berkman Klein Center for Internet & Society, Cambridge, MA. https://dash.harvard.edu/handle/1/42160420

[56] International Organization for Standardization, Software Technical Committee ISO/IEC JTC 1, Information technology. Subcommittee SC 7, and systems engineering. 2011. *Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): System and Software Quality Models.* ISO, Switzerland.

[57] M. Foughali, B. Berthomieu, S. Dal Zilio, F. Ingrand, and A. Mallet. 2016. Model Checking Real-Time Properties on the Functional Layer of Autonomous Robots. In *Formal Methods and Soft. Eng.* Springer, Cham, 383–399.

[58] Mirgita Frasheri, Vaclav Struhar, Alessandro Vittorio Papadopoulos, and Aida Causevic. 2022. Ethics of Autonomous Collective Decision-Making: The CAESAR Framework. *Science and Engineering Ethics* 28, 6 (2022), 61. doi:10.1007/s11948-022-00414-0

[59] A. Fuller, Z. Fan, C. Day, and C. Barlow. 2020. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access* 8 (2020), 108952–108971. doi:10.1109/ACCESS.2020.2998358

[60] Sergio García, Daniel Strüber, Davide Brugali, Thorsten Berger, and Patrizio Pelliccione. 2020. Robotics software engineering: A perspective from the service robotics domain. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* Association for Computing Machinery, New York, NY, USA, 593–604. doi:10.1145/3368089.3409743

[61] H. Giese and W. Schäfer. 2013. Model-Driven Development of Safe Self-optimizing Mechatronic Systems with MechatronicUML. In *Assurances for Self-Adaptive Systems - Principles, Models, and Techniques*, J. Cámara, R. Lemos, C. Ghezzi, and A. Lopes (Eds.). Lecture Notes in Computer Science, Vol. 7740. Springer, Berlin Heidelberg, 152–186.

[62] Santiago Gil, Bentley J Oakes, Cláudio Gomes, Mirgita Frasheri, and Peter G Larsen. 2025. Toward a systematic reporting framework for Digital Twins: a cooperative robotics case study. *SIMULATION* 101, 3 (2025), 313–339. doi:10.1177/00375497241261406

[63] Kyle Gilpin and Daniela Rus. 2010. Modular Robot Systems. *IEEE Robotics and Automation Magazine* 17, 3 (2010), 38–55. doi:10.1109/MRA.2010.937859

[64] N. Gobillot, C. Lesire, and D. Doose. 2014. A Modeling Framework for Software Architecture Specification and Validation. In *Simulation, Modeling, and Programming for Autonomous Robots*, D. Brugali, J. F. Broenink, T. Kroeger, and B. A. MacDonald (Eds.). Springer International Publishing, Cham, 303–314.

[65] Cláudio Gomes, Torsten Blochwitz, Christian Bertsch, Karl Wernersson, Klaus Schuch, Pierre R., Oliver Kotte, Irina Zacharias, Matthias Blesken, Torsten Sommer, Masoud Najafi, and Andreas Junghanns. 2021. The FMI 3.0 Standard Interface for Clocked and Scheduled Simulations. In *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021 (Modelica 2021, Vol. 181).* Linköping University Electronic Press, Linköping, Sweden, 27–36. doi:10.3384/ecp2118127

[66] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. 2018. Co-Simulation: A Survey. *Comput. Surveys* 51, 3 (May 2018), 1–33. doi:10.1145/3179993

[67] Claire Le Goues, Sebastian Elbaum, David Anthony, Z. Berkay Celik, Mauricio Castillo-Effen, Nikolaus Correll, Pooyan Jamshidi, Morgan Quigley, Trenton Tabor, and Qi Zhu. 2024. Software Engineering for Robotics: Future Research Directions; Report from the 2023 Workshop on Software Engineering for Robotics. arXiv:2401.12317 [cs.RO] https://arxiv.org/abs/2401.12317

[68] Samuel Greengard. 2025. Can LLMs Make Robots Smarter? *Commun. ACM* 68, 2 (2025), 11–13. doi:10.1145/3701227

[69] P. Han, A. L. Ellefsen, G. Li, V. Æsøy, and H. Zhang. 2021. Fault prognostics using LSTM networks: application to marine diesel engine. *IEEE Sensors Journal* 21, 22 (2021), 25986–25994.

[70] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2020. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 2 (2020), 386–397. doi:10.1109/TPAMI.2018.2844175

[71] Robert M Hierons, Maciej Gazda, Pablo Gómez-Abajo, Raluca Lefticaru, and Mercedes G Merayo. 2021. Mutation Testing for RoboChart. *Software Engineering for Robotics* (2021), 345–375. doi:10.1007/978-3-030-66494-7_11

[72] T. Hoebert, W. Lepuschitz, E. List, and M. Merdan. 2019. Cloud-based digital twin for industrial robotics. In *Industrial Applications of Holonic and Multi-Agent Systems: 9th International Conference, HoloMAS 2019, Linz, Austria, August 26–29, 2019, Proceedings 9.* Springer International Publishing, Switzerland, 105–116. doi:10.1007/978-3-030-25894-3_9

[73] Ali M Hosseini, Clara Fischer, Mukund Bhole, Wolfgang Kastner, Thilo Sauter, and Sebastian Schlund. 2023. A safety and security requirements management methodology in reconfigurable collaborative human-robot application. In *2023 IEEE 19th International Conference on Factory Communication Systems (WFCS).* IEEE, New York, NY, USA, 1–8. doi:10.1109/WFCS57264.2023.10144233

[74] Emin Oguz Inci, Jan Croes, Wim Desmet, Claudio Gomes, Casper Thule, Kenneth Lausdahl, and Peter Gorm Larsen. 2021. The Effect and Selection of Solution Sequence in Co-Simulation. In *2021 Annual Modeling and Simulation Conference (ANNSIM)*. IEEE, New York, NY, USA, 1–12. doi:10.23919/annsim52504.2021.9552130

[75] Emin Oguz Inci, Wim Desmet, Cláudio Gomes, and Jan Croes. 2023. Error Estimators for Adaptive Scheduling Algorithm for Serial Co-Simulation. In *2023 Annual Modeling and Simulation Conference (ANNSIM)*. IEEE, New York, NY, USA, 73–83.

[76] Juan Francisco Inglés Romero, Cristina Vicente Chicote, Brice Morin, and Olivier Barais. 2010. Using Models@ Runtime for designing adaptive robotics software: an experience report. In *International Workshop on Model Based Engineering for Robotics*. Laurent Riox, Oslo, Norway, 1–11.

[77] Juan F Inglés-Romero, Cristina Vicente-Chicote, Brice Morin, and Olivier Barais. 2011. Towards the automatic generation of self-adaptive robotics software: An experience report. In *2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE, New York, NY, USA, 79–86. doi:10.1109/WETICE.2011.54

[78] Paola Inverardi. 2022. *The Challenge of Human Dignity in the Era of Autonomous Systems*. Springer International Publishing, Cham, Chapter 4, 25–29. doi:10.1007/978-3-030-86144-5_4

[79] Erblin Isaku, Hassan Sartaj, and Shaukat Ali. 2025. Digital Twin-based Out-of-Distribution Detection in Autonomous Vessels. arXiv:2504.19816 [eess.SY] https://arxiv.org/abs/2504.19816

[80] Erblin Isaku, Hassan Sartaj, Shaukat Ali, Beatriz Sanguino, Tongtong Wang, Guoyuan Li, Houxiang Zhang, and Thomas Peyrucain. 2025. Out of Distribution Detection in Self-adaptive Robots with AI-powered Digital Twins. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, New York, NY, USA, 3403–3414. doi:10.1109/ASE63991.2025.00281

[81] Zainab Javed, Muhammad Zohaib Iqbal, Muhammad Uzair Khan, Muhammad Usman, and Atif Aftab Ahmed Jilani. 2024. An automated model-based testing approach for the self-adaptive behavior of the unmanned aircraft system application software. *Software: Practice and Experience* 54, 12 (2024), 2375–2427. doi:10.1002/spe.3358

[82] Zainab Javed, Muhammad Zohaib Iqbal, Muhammad Uzair Khan, Muhammad Usman, and Atif Aftab Ahmed Jilani. 2025. A hybrid search and model-based approach for testing the self-adaptive unmanned aircraft system software. *Computer Standards & Interfaces* 93 (2025), 103959. doi:10.1016/j.csi.2024.103959

[83] Andreas Junghanns, Cláudio Gomes, Christian Schulze, Klaus Schuch, Pierre R., Matthias Blaesken, Irina Zacharias, Andreas Pillekeit, Karl Wernersson, Torsten Sommer, Christian Bertsch, Torsten Blochwitz, and Masoud Najafi. 2021. The Functional Mock-up Interface 3.0 - New Features Enabling New Applications. In *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021 (Modelica 2021, Vol. 181)*. Linköping University Electronic Press, Linköping, Sweden, 17–26. doi:10.3384/ecp2118117

[84] Jeffrey O Kephart and David M Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (2003), 41–50. doi:10.1109/MC.2003.1160055

[85] Jeffrey O. Kephart, Ada Diaconescu, Holger Giese, Anders Robertsson, Tarek Abdelzaher, Peter Lewis, Antonio Filieri, Lukas Esterle, and Sylvain Frey. 2017. *Self-adaptation in Collective Self-aware Computing Systems*. Springer International Publishing, Cham, 401–435. doi:10.1007/978-3-319-47474-8_13

[86] David Kortenkamp, Reid Simmons, and Davide Brugali. 2016. Robotic systems architectures and programming. In *Springer handbook of robotics*. Springer, Cham, 283–306. doi:10.1007/978-3-319-32552-1_12

[87] N. Kousi, C. Gkournelos, S. Aivaliotis, K. Lotsaris, A. C. Bavelos, P. Baris, and S. Makris. 2021. Digital twin for designing and reconfiguring human–robot collaborative assembly lines. *Applied Sciences* 11, 10 (2021), 4620. doi:10.3390/app11104620

[88] Morten Haahr Kristensen, Thomas Wright, Cláudio Gomes, Lukas Esterle, and Peter Gorm Larsen. 2026. DynSRV: Dynamically Updated Properties for Stream Runtime Verification. In *Runtime Verification*, Bettina Könighofer and Hazem Torfah (Eds.). Springer Nature Switzerland, Cham, 101–119. doi:10.1007/978-3-032-05435-7_7

[89] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. 2018. Digital Twin In Manufacturing: A Categorical Literature Review And Classification. *IFAC-PapersOnLine* 51 (01 2018), 1016–1022. doi:10.1016/j.ifacol.2018.08.474

[90] R. Kubler and W. Schiehlen. 2000. Two Methods of Simulator Coupling. *Mathematical and Computer Modelling of Dynamical Systems* 6, 2 (June 2000), 93–113. doi:10.1076/1387-3954(200006)6:2;1-m;ft093

[91] Thomas Kühne. 2005. What is a Model?. In *Language Engineering for Model-Driven Software Development (Dagstuhl Seminar Proceedings (DagSemProc), Vol. 4101)*, Jean Bezivin and Reiko Heckel (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 1–10. doi:10.4230/DagSemProc.04101.15

[92] Peter G Larsen, Shaukat Ali, Roland Behrens, Ana Cavalcanti, Claudio Gomes, Guoyuan Li, Paul De Meulenaere, Mikkel L Olsen, Nikolaos Passalis, Thomas Peyrucain, et al. 2024. Robotic safe adaptation in unprecedented situations: the RoboSAPIENS project. *Research Directions: Cyber-Physical Systems* 2 (2024), e4. doi:10.1017/cbp.2024.4

[93] Peter Gorm Larsen, John Fitzgerald, and Jim Woodcock. 2023. How do we Engineer Trustworthy Digital Twins? *Research Directions: Cyber-Physical Systems* 1 (2023), 1–6. doi:10.1017/cbp.2023.3

[94] Emmanuel Letier and Axel Van Lamsweerde. 2025. Obstacle Analysis in Requirements Engineering: Retrospective and Emerging Challenges. *IEEE Transactions on Software Engineering* 51, 3 (2025), 795–801. doi:10.1109/TSE.2025.3534318

[95] C. Li, P. Zheng, S. Li, Y. Pang, and C. K. Lee. 2022. AR-assisted digital twin-enabled robot collaborative manufacturing system with human-in-the-loop. *Robotics and Computer-Integrated Manufacturing* 76 (2022), 102321. doi:10.1016/j.rcim.2021.102321

[96] Jialong Li, Mingyue Zhang, Nianyu Li, Danny Weyns, Zhi Jin, and Kenji Tei. 2024. Exploring the Potential of Large Language Models in Self-adaptive Systems. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. Association for Computing Machinery, New York, NY, USA, 77–83. doi:10.1145/3643915.3644088

[97] X. Li, B. He, Z. Wang, Y. Zhou, G. Li, and R. Jiang. 2021. Semantic-enhanced digital twin system for robot–environment interaction monitoring. *IEEE Transactions on Instrumentation and Measurement* 70 (2021), 1–13. doi:10.1109/TIM.2021.3077125

[98] L. Lima, A. Miyazawa, A. L. C. Cavalcanti, M. Cornélio, J. Iyoda, A. C. A. Sampaio, R. Hains, A. Larkham, and V. Lewis. 2017. An integrated semantics for reasoning about SysML design models using refinement. *Software & Systems Modeling* 16, 3 (2017), 1–28. doi:10.1007/s10270-015-0492-y

[99] Yim Ling Loo, Azhana Ahmad, Aida Mustapha, and Salama A. Mostafa. 2021. A Self-Adaptive Agent-Based Dynamic Processes Simulation Modelling Framework. In *2021 4th International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR)*. IEEE, New York, NY, USA, 124–129. doi:10.1109/isamsr53229.2021.9567854

[100] Alex Lotz, Juan F Inglés-Romero, Cristina Vicente-Chicote, and Christian Schlegel. 2013. Managing run-time variability in robotics software by modeling functional and non-functional behavior. In *International Workshop on Business Process Modeling, Development and Support*. Springer, Berlin, Heidelberg, 441–455. doi:10.1007/978-3-642-38484-4_31

[101] Artem Lykov and Dzmitry Tsetserukou. 2024. LLM-BRAIn: AI-driven Fast Generation of Robot Behaviour Tree based on Large Language Model. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*. IEEE, New York, NY, USA, 392–397.

[102] Tao Ma, Shaukat Ali, and Tao Yue. 2019. Modeling foundations for executable model-based testing of self-healing cyber-physical systems. *Software & Systems Modeling* 18 (2019), 2843–2873. doi:10.1007/s10270-018-00703-y

[103] Tao Ma, Shaukat Ali, and Tao Yue. 2021. Testing self-healing cyber-physical systems under uncertainty with reinforcement learning: an empirical study. *Empirical Software Engineering* 26 (2021), 1–54. doi:10.1007/s10664-021-09941-z

[104] P. Major, G. Li, H. P. Hildre, and H. Zhang. 2021. The use of a data-driven digital twin of a smart city: A case study of Ålesund, Norway. *IEEE Instrumentation & Measurement Magazine* 24, 7 (2021), 39–49. doi:10.1109/MIM.2021.9549127

[105] Ali Ahmad Malik and Alexander Brem. 2021. Digital twins for collaborative robots: A case study in human-robot interaction. *Robotics and Computer-Integrated Manufacturing* 68 (2021), 102092. doi:10.1016/j.rcim.2020.102092

[106] Patrick Mania, Michael Neumann, Franklin Kenghagho Kenfack, and Michael Beetz. 2025. Towards autonomous verification: Integrating cognitive AI and semantic digital twins in medical robotics. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, New York, NY, USA, 1736–1742. doi:10.1109/ICRA55743.2025.11127803

[107] R. Mao, Y. Li, G. Li, H. P. Hildre, and H. Zhang. 2025. A systematic survey of digital twin applications: Transferring knowledge from automotive and aviation to maritime industry. *IEEE Transactions on Intelligent Transportation Systems* 26, 4 (2025), Early Access. doi:10.1109/TITS.2025.3535593

[108] S. Maoz and J. O. Ringert. 2018. *Spectra Language and Spectra Tools User Guide*. smlab.cs.tau.ac.il/syntech/spectra/

[109] Francesco Maurelli, Szymon Krupiński, Xianbo Xiang, and Yvan Petillot. 2022. AUV localisation: a review of passive and active techniques. *International Journal of Intelligent Robotics and Applications* 6, 2 (2022), 246–269. doi:10.1007/s41315-021-00215-x

[110] A. Mazumder, M. F. Sahed, Z. Tasneem, P. Das, F. R. Badal, M. F. Ali, and M. R. Islam. 2023. Towards next generation digital twin in robotics: Trends, scopes, challenges, and future. *Heliyon* 9, 2 (2023), e12982. doi:10.1016/j.heliyon.2023.e12982

[111] Nenad Medvidovic, Hossein Tajalli, Joshua Garcia, Ivo Krka, Yuriy Brun, and George Edwards. 2010. Engineering heterogeneous robotics systems: A software architecture-based approach. *Computer* 44, 5 (2010), 62–71. doi:10.1109/MC.2010.368

[112] Alvaro Miyazawa, Sharar Ahmadi, Ana Cavalcanti, James Baxter, Mark Post, Pedro Ribeiro, Jon Timmis, and Thomas Wright. 2025. Diagrammatic physical robot models: A. Miyazawa et al. *Software and Systems Modeling* 24, 5 (2025), 1549–1593. doi:10.1007/s10270-025-01270-9

[113] Galen E Mullins, Paul G Stankiewicz, R Chad Hawthorne, and Satyandra K Gupta. 2018. Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles. *Journal of Systems and Software* 137 (2018),

197–215. doi:10.1016/j.jss.2017.10.031

[114] Asmar Muqeet, Hassan Sartaj, Aitor Arrieta, Shaukat Ali, Paolo Arcaini, Maite Arratibel, Julie Marie Gjøby, Narasimha Raghavan Veeraragavan, and Jan F. Nygård. 2024. Assessing Quantum Extreme Learning Machines for Software Testing in Practice. arXiv:2410.15494 [cs.SE] https://arxiv.org/abs/2410.15494

[115] Thomas Nagele and Jozef Hooman. 2017. Co-simulation of cyber-physical systems using HLA. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, New York, NY, USA, 1–6. doi:10.1109/ccwc.2017.7868401

[116] Sahar Nasimi Nezhad, Bert Van Acker, and Paul De Meulenaere. 2025. Towards A Standardized Framework For Developing Trustworthy Self-Adaptive Robotic Systems. In *2025 Annual Modeling and Simulation Conference (ANNSIM)*. IEEE, New York, NY, USA, 1–13.

[117] Hai Nguyen and Hung La. 2019. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, New York, NY, USA, 590–595.

[118] P. C. Ölveczky, A. Boronat, and J. Meseguer. 2010. Formal Semantics and Analysis of Behavioral AADL Models in Real-Time Maude. In *Formal Techniques for Distributed Systems*, J. Hatcliff and E. Zucca (Eds.). Springer, Berlin Heidelberg, 47–62.

[119] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–35. doi:10.1145/3453160

[120] Danilo Pianini, Federico Pettinari, Roberto Casadei, and Lukas Esterle. 2022. A Collective Adaptive Approach to Decentralised k-Coverage in Multi-robot Systems. *ACM Trans. Auton. Adapt. Syst.* 17, 1–2, Article 4 (Sept. 2022), 39 pages. doi:10.1145/3547145

[121] Maria Pozzi, Gabriele Maria Achilli, Maria Cristina Valigi, and Monica Malvezzi. 2022. Modeling and Simulation of Robotic Grasping in Simulink Through Simscape Multibody. *Frontiers in Robotics and AI* 9 (2022), 873558. doi:10.3389/frobt.2022.873558

[122] Ahmad Rafsanjani, Katia Bertoldi, and André R. Studart. 2019. Programming soft robots with flexible mechanical metamaterials. *Science Robotics* 4, 29 (2019), eaav7874. doi:10.1126/scirobotics.aav7874

[123] A. Ramaswamy, B. Monsuez, and A. Tapus. 2014. SafeRobots: A model-driven Framework for developing Robotic Systems. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, New York, NY, USA, 1517–1524.

[124] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New York, NY, USA, 779–788. doi:10.1109/CVPR.2016.91

[125] J. O. Ringert, A. Roth, B. Rumpe, and A. Wortmann. 2015. Code Generator Composition for Model-Driven Engineering of Robotics Component & Connector Systems. *Journal of Software Engineering for Robotics* 6, 1 (2015), 33–57.

[126] Luis J Figueroa Rivera and Balasubramaniyan Chandrasekaran. 2021. A Software Based Self-Recovering Robotic System Architecture Using ROS. In *2021 7th International Conference on Mechatronics and Robotics Engineering (ICMRE)*. IEEE, New York, NY, USA, 29–34. doi:10.1109/ICMRE51691.2021.9384816

[127] Yara Rizk, Mariette Awad, and Edward W. Tunstel. 2019. Cooperative Heterogeneous Multi-Robot Systems: A Survey. *Comput. Surveys* 52, 2, Article 29 (April 2019), 31 pages. doi:10.1145/3303848

[128] William N. Robinson, Suzanne D. Pawlowski, and Vladimir Volkov. 2003. Requirements interaction management. *Comput. Surveys* 35, 2 (2003), 132–190. doi:10.1145/857076.857078

[129] Thomas Roddick and Roberto Cipolla. 2020. Predicting semantic map representations from images using pyramid occupancy networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York, NY, USA, 11138–11147. doi:10.1109/CVPR42600.2020.01115

[130] Hassan Sartaj. 2021. Automated approach for system-level testing of unmanned aerial systems. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, New York, NY, USA, 1069–1073. doi:10.1109/ASE51524.2021.9678902

[131] Hassan Sartaj, Shaukat Ali, and Julie Marie Gjøby. 2025. MeDeT: Medical Device Digital Twins Creation with Few-shot Meta-learning. *ACM Transactions on Software Engineering and Methodology* 34, 6 (2025), 1–36. doi:10.1145/3708534

[132] Hassan Sartaj, Shaukat Ali, Tao Yue, and Kjetil Moberg. 2024. Model-based digital twins of medicine dispensers for healthcare IoT applications. *Software: Practice and Experience* 54, 6 (2024), 1172–1192. doi:10.1002/spe.3311

[133] Hassan Sartaj, Muhammad Zohaib Iqbal, Atif Aftab Ahmed Jilani, and Muhammad Uzair Khan. 2025. Search-Based MC/DC Test Data Generation With OCL Constraints. *Software Testing, Verification And Reliability* 35, 1 (2025), e1906. doi:10.1002/stvr.1906

[134] Hassan Sartaj, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan. 2021. Testing cockpit display systems of aircraft using a model-based approach. *Software and Systems Modeling* 20, 6 (2021), 1977–2002. doi:10.1007/s10270-020-00844-z

[135] Hassan Sartaj, Asmar Muqeet, Muhammad Zohaib Iqbal, and Muhammad Uzair Khan. 2024. Automated system-level testing of unmanned aerial systems. *Automated Software Engineering* 31, 64 (2024), 1–48. doi:10.1007/s10515-024-

00462-9

[136] Tim Schreiter, Jens V Rüppel, Rishi Hazra, Andrey Rudenko, Martin Magnusson, and Achim J Lilienthal. 2025. Evaluating efficiency and engagement in scripted and LLM-enhanced human-robot interactions. In *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, New York, NY, USA, 1608–1612. doi:10.1109/HRI61500.2025.10974124

[137] Gustavo Rezende Silva, Juliane Päßler, Jeroen Zwanepol, Elvin Alberts, S Lizeth Tapia Tarifa, Ilias Gerostathopoulos, Einar Broch Johnsen, and Carlos Hernández Corbato. 2023. SUAVE: an exemplar for self-adaptive underwater vehicles. In *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, New York, NY, USA, 181–187. doi:10.1109/SEAMS59076.2023.00031

[138] Z. Song, H. Shi, X. Bai, and G. Li. 2023. Digital twin-assisted fault diagnosis system for robot joints with insufficient data. *Journal of Field Robotics* 40, 2 (2023), 258–271. doi:10.1002/rob.22177

[139] Maria Spichkova, Florian Hölzl, and David Trachtenherz. 2012. Verified System Development with the AutoFocus Tool Chain. In *Proceedings 2nd Workshop on Formal Methods in the Development of Software*. EPTCS, Paris, France, 17–24. doi:10.4204/EPTCS.86.3

[140] Nathan A Spielberg, Matthew Brown, Nitin R Kapania, John C Kegelman, and J Christian Gerdes. 2019. Neural network vehicle models for high-performance automated driving. *Science robotics* 4, 28 (2019), eaaw1975. doi:10.1126/scirobotics.aaw1975

[141] T. Sun, X. He, and Z. Li. 2023. Digital twin in healthcare: Recent updates and challenges. *Digital Health* 9 (2023), 20552076221149651. doi:10.1177/20552076221149651

[142] F. Tao and M. Zhang. 2017. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. *IEEE Access* 5 (2017), 20418–20427.

[143] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, New York, NY, USA, 23–30. doi:10.1109/IROS.2017.8202133

[144] B. Townsend, C. Paterson, T. T. Arvind, G. Nemirovsky, R. Calinescu, A. L. C. Cavalcanti, I. Habli, and A. Thomas. 2022. From Pluralistic Normative Principles to Autonomous-Agent Rules. *Minds and Machines* 32, 4 (2022), 683–715. doi:10.1007/s11023-022-09614-w

[145] Nicolas Troquard, Martina De Sanctis, Paola Inverardi, Patrizio Pelliccione, and Gian Luca Scoccia. 2024. Social, Legal, Ethical, Empathetic, and Cultural Rules: Compilation and Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 38. AAAI Press, Washington, DC, USA, 22385–22392. doi:10.1609/aaai.v38i20.30245

[146] Axel van Lamsweerde. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, NJ, USA.

[147] Chao Wang, Jian Wang, Yuan Shen, and Xudong Zhang. 2019. Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology* 68, 3 (2019), 2124–2136.

[148] Tongtong Wang, Guoyuan Li, Lars Ivar Hatledal, Robert Skulstad, Vilmar Æsøy, and Houxiang Zhang. 2021. Incorporating approximate dynamics into data-driven calibrator: A representative model for ship maneuvering prediction. *IEEE Transactions on Industrial Informatics* 18, 3 (2021), 1781–1789. doi:10.1109/TII.2021.3088404

[149] Wenhao Wang, Yanyan Li, Long Jiao, and Jiawei Yuan. 2025. GSCE: A prompt framework with enhanced reasoning for reliable LLM-driven drone control. In *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, New York, NY, USA, 441–448. doi:10.1109/ICUAS65942.2025.11007864

[150] Xinyi Wang, Shaukat Ali, and Paolo Arcaini. 2025. Quantum Artificial Intelligence for Software Engineering: the Road Ahead. arXiv:2505.04797 [cs.SE] https://arxiv.org/abs/2505.04797

[151] Xinyi Wang, Shaukat Ali, Aitor Arrieta, Paolo Arcaini, and Maite Arratibel. 2024. Application of Quantum Extreme Learning Machines for QoS Prediction of Elevators' Software in an Industrial Context. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering* (Porto de Galinhas, Brazil) *(FSE 2024)*. ACM, New York, NY, USA, 399–410. doi:10.1145/3663529.3663859

[152] Xinyi Wang, Qinghua Xu, Paolo Arcaini, Shaukat Ali, and Thomas Peyrucain. 2025. Quantum Machine Learning-based Test Oracle for Autonomous Mobile Robots. In *Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering (ASE 2025)*. IEEE/ACM, Seoul, Republic of Korea. https://arxiv.org/abs/2508.02407 Accepted at ASE 2025.

[153] Danny Weyns, Radu Calinescu, Raffaela Mirandola, Kenji Tei, Maribel Acosta, Nelly Bencomo, Amel Bennaceur, Nicolas Boltz, Tomas Bures, Javier Camara, et al. 2023. Towards a research agenda for understanding and managing uncertainty in self-adaptive systems. *ACM SIGSOFT Software Engineering Notes* 48, 4 (2023), 20–36. doi:10.1145/3617946.3617951

[154] Danny Weyns and Usman M. Iftikhar. 2023. ActivFORMS: A Formally Founded Model-based Approach to Engineer Self-adaptive Systems. *ACM Transactions on Software Engineering and Methodology* 32, 1 (Jan. 2023), 1–48. doi:10.

1145/3522585
[155] Thomas Wright, Cláudio Gomes, and Jim Woodcock. 2022. *Formally Verified Self-adaptation of an Incubator Digital Twin.* Springer Nature, Switzerland, 89–109. doi:10.1007/978-3-031-19762-8_7
[156] Jiahui Wu, Chengjie Lu, Aitor Arrieta, Shaukat Ali, and Thomas Peyrucain. 2025. Vision Language Model-based Testing of Industrial Autonomous Mobile Robots. arXiv:2508.02338 [cs.SE] https://arxiv.org/abs/2508.02338
[157] Xiyang Wu, Ruiqi Xian, Tianrui Guan, Jing Liang, Souradip Chakraborty, Fuxiao Liu, Brian M Sadler, Dinesh Manocha, and Amrit Bedi. 2024. On the safety concerns of deploying LLMs/VLMs in robotics: Highlighting the risks and vulnerabilities. In *First Vision and Language for Autonomous Driving and Robotics Workshop*. 1–10.
[158] Xuan Xiao, Jiahang Liu, Zhipeng Wang, Yanmin Zhou, Yong Qi, Shuo Jiang, Bin He, and Qian Cheng. 2025. Robot learning in the era of foundation models: A survey. *Neurocomputing* 638, 14 (2025), 129963. doi:10.1016/j.neucom.2025.129963
[159] H. Xie, M. Xin, C. Lu, and J. Xu. 2023. Knowledge map and forecast of digital twin in the construction industry: State-of-the-art review using scientometric analysis. *Journal of Cleaner Production* 383 (2023), 135231.
[160] Minglan Xiong and Huawei Wang. 2022. Digital twin applications in aviation industry: A review. *The International Journal of Advanced Manufacturing Technology* 121, 9-10 (2022), 5677–5692. doi:10.1007/s00170-022-09717-9
[161] S. G. Yaman, P. Ribeiro, A. L. C. Cavalcanti, R. Calinescu, C. Paterson, and B. Townsend. 2025. Specification, validation and verification of social, legal, ethical, empathetic and cultural requirements for autonomous agents. *Journal of Systems and Software* 220 (2025), 112229. doi:10.1016/j.jss.2024.112229
[162] H. Yang, Z. Qin, Y. Xia, and F. Cheng. 2025. Digital Twin-Based Autonomous Navigation and Control of Omnidirectional Mobile Robots. *IEEE Transactions on Vehicular Technology* 74, 4 (2025), 5687–5697. doi:10.1109/TVT.2024.3254501
[163] Sheir Yarkoni, Elena Raponi, Thomas Bäck, and Sebastian Schmitt. 2022. Quantum annealing for industry applications: introduction and review. *Reports on Progress in Physics* 85, 10 (sep 2022), 104001. doi:10.1088/1361-6633/ac8c54
[164] Oncay Yasa, Yasunori Toshimitsu, Mike Y. Michelis, Lewis S. Jones, Miriam Filippi, Thomas Buchner, and Robert K. Katzschmann. 2023. An Overview of Soft Robotics. *Annual Review of Control, Robotics, and Autonomous Systems* 6, Volume 6, 2023 (2023), 1–29. doi:10.1146/annurev-control-062322-100607
[165] M. Yim, Ying Zhang, and D. Duff. 2002. Modular robots. *IEEE Spectrum* 39, 2 (2002), 30–34. doi:10.1109/6.981854
[166] Houxiang Zhang, Guoyuan Li, Lars Ivar Hatledal, Yingguang Chu, André Ellefsen, Peihua Han, Pierre Major, Robert Skulstad, Tongtong Wang, and Hans Petter Hildre. 2023. A Digital Twin of the Research Vessel Gunnerus for Lifecycle Services: Outlining Key Technologies. *IEEE Robotics & Automation Magazine* 30, 3 (2023), 6–19. doi:10.1109/MRA.2022.3217745
[167] H. Zhang, G. Li, and H. P. Hildre. 2024. *Digital Twins for Vessel Life Cycle Service: Innovation in Maritime Industry.* Springer Nature, Singapore.
[168] L. Zhang and He Jifeng. 2011. A Formal Framework for Aspect-Oriented Specification of Cyber Physical Systems. In *Convergence and Hybrid Information Technology (Communications in Computer and Information Science, Vol. 206)*, G. Lee, D. Howard, and D. Slezak (Eds.). Springer, Berlin, Heidelberg, 391–398.
[169] Wenxiao Zhang, Xiangrui Kong, Conan Dewitt, Thomas Bräunl, and Jin B. Hong. 2025. Enhancing Reliability in LLM-Integrated Robotic Systems: A Unified Approach to Security and Safety. arXiv:2509.02163 [cs.RO] https://arxiv.org/abs/2509.02163