Software architecture-based self-adaptation in robotics

Alberts, Elvin; Gerostathopoulos, Ilias; Malavolta, Ivano; Hernández Corbato, Carlos; Lago, Patricia

DOI
10.1016/j.jss.2024.112258

Publication date
2025

Document Version
Final published version

Published in
Journal of Systems and Software

Citation (APA)
Alberts, E., Gerostathopoulos, I., Malavolta, I., Hernández Corbato, C., & Lago, P. (2025). Software architecture-based self-adaptation in robotics. *Journal of Systems and Software*, *219*, Article 112258. https://doi.org/10.1016/j.jss.2024.112258

The Journal of Systems and Software 219 (2025) 112258

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Software architecture-based self-adaptation in robotics ☆

Elvin Alberts [a,b,*], Ilias Gerostathopoulos [a], Ivano Malavolta [a], Carlos Hernández Corbato [b], Patricia Lago [a]

[a] Vrije Universiteit Amsterdam, The Netherlands
[b] Delft University of Technology, The Netherlands

ARTICLE INFO

Keywords:
Software architecture
Self-adaptation
Robotics
Systematic mapping study

ABSTRACT

Context: Robotics software architecture-based self-adaptive systems (RSASSs) are robotics systems made robust to runtime uncertainty by adapting their software architectures. The research landscape of RSASS approaches is multidisciplinary and fragmented, with many aspects still unexplored or ineffectively shared among communities involved.
Objective: We aim at identifying, classifying, and analyzing the state of the art of existing approaches for RSASSs from the following perspectives: (i) the key characteristics of approaches and (ii) the evaluation strategies applied by researchers.
Method: We apply the systematic mapping research method. We selected 37 primary studies via automatic, manual, and snowballing-based search and selection procedures. We rigorously defined and applied a classification framework composed of 32 parameters and synthesize the obtained data to produce a comprehensive overview of the state of the art.
Results: This work contributes (i) a rigorously defined classification framework for studies on RSASSs, (ii) a systematic map of the research efforts on RSASSs, (iii) a discussion of emerging findings and implications for future research, and (iv) a publicly available replication package.
Conclusion: This study provides a solid evidence-based overview of the state of the art in RSASS approaches. Its results can benefit RSASS researchers at different levels of seniority and involvement in RSASS research.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

1. Introduction

Globally there is a consistent increase in the use and prevalence of robotics (International Federation of Robotics, 2022). As the applications of robotics increase, the breadth of requirements imposed on those systems, both functional and non-functional, tend to increase too. Additionally, the conditions within which robotics missions take place are dynamic and subject to sudden and unexpected changes. To the extent that these increased requirements can be met through software, self-adaptation is a promising solution to handling uncertainty at runtime (de Lemos et al., 2013; Weyns, 2020). Self-adaptive systems are defined as "computing systems that can adapt autonomously to achieve their goals based on high-level objectives" (Weyns, 2020). Self-adaptation solutions provide flexibility for both dealing with uncertainties during operation and evolving a system to incorporate new requirements. It is then of interest to study the application of self-adaptation solutions to the robotics domain. This is evidenced by recent research into the area of self-adaptation in robotics (Edrisi et al., 2023; Silva et al., 2023; Behery et al., 2023; Li et al., 2023)

In this study we focus on *robotics software architecture-based self-adaptive systems* (RSASSs). Architecture-based self-adaptation, where an architectural model of a system is used for representing and reasoning about adaptation decisions at runtime, is an established and effective approach in engineering self-adaptive systems (Oreizy et al., 1998; Garlan et al., 2004; Geihs et al., 2009; Weyns et al., 2012). Focusing on the architectural level is suitable since, first, software architectures are always present, either through explicit modeling or implicitly considered in the development of any software system, including robotics. Second, software architecture offers the right level of abstraction for reasoning about the system's properties since it offers a holistic yet

☆ Editor: Alexander Chatzigeorgiou.

* Corresponding author at: Vrije Universiteit Amsterdam, The Netherlands.
E-mail addresses: e.g.alberts@vu.nl (E. Alberts), i.g.gerostathopoulos@vu.nl (I. Gerostathopoulos), i.malavolta@vu.nl (I. Malavolta), c.h.corbato@tudelft.nl (C. Hernández Corbato), p.lago@vu.nl (P. Lago).

https://doi.org/10.1016/j.jss.2024.112258
Received 15 April 2024; Received in revised form 5 October 2024; Accepted 11 October 2024
Available online 18 October 2024

accessible view of the system through its components and their interactions (Weyns, 2020). Third, self-adaptation solutions formed at an architecture level offer the possibility of abstracting away from a particular application or mission and can be potentially reused within a given domain (in this case the robotics software domain). For example, a software architecture-based self-adaptive robot when in an unsafe position can adapt to lower the maximum velocity parameter used by its autonomous navigation component, or replace that component in favor of a teleoperation component allowing a human to steer the robot into safety. Unfortunately, as of yet, the extent and characteristics of research into RSASSs is fragmented and scattered across multiple publication venues in different fields, mainly robotics, self-adaptive systems, software engineering and software architecture.

As a representative **example of an RSASS**, the authors of Gerostathopoulos et al. (2019) provide an approach where robots switch their operation based on a mode-state machine. This model acts at an architectural level since it prescribes both behavioral and structural runtime changes such as "move to charging station" or "use less power-hungry detection method" in system components in response to uncertain events such as "battery low" and "too many obstacles detected".

The **goal of this study** is to provide a comprehensive overview of research on RSASSs and characterize the approaches to RSASSs. We achieve this goal by applying the **systematic mapping research method** targeting studies which endow robots with architecture-based self-adaptive capabilities. Specifically, we first systematically collect 45 443 studies and through title-based keyword filtering process, we narrow down our search to 3087 potentially relevant studies. From these 3087 studies, a well-defined set of selection criteria is applied, leading to an initial set of 17 primary studies. We use these 17 primary studies in a snowballing procedure, resulting in the identification of 21 additional primary studies, leading to a final set of 37 primary studies. For these 37 studies, we characterize them through a rigorously defined classification framework containing 32 distinct parameters. Overall, in this study we collected a total of 1184 data points (not accounting for potential multiple values per parameter). This process is presented in full detail in Section 2. Afterwards we perform vertical and horizontal synthesis of the extracted data (Sections 3–8) and present the main implications resulting from such a synthesis (Section 9). Additionally, the threats to the validity of our study and their mitigation are presented in Section 10. We compare against related secondary studies to ours in Section 11 and finally conclude by reiterating the main results of the study in Section 12.

Overall, the **main contributions of this study** are: (i) a systematic map of the research efforts on RSASSs, (ii) a rigorously-defined classification framework for categorizing RSASS approaches, (iii) a discussion about the main implications of the map for RSASS researchers, (iv) a publicly available replication package for independent verification and replication of this study.

The **target audience** of this study includes researchers active in the RSASS community. Specifically, (i) young researchers entering the RSASS community can use our results as a starting point for getting a comprehensive and objective overview of the RSASS research area, (ii) researchers already active in the RSASS community can use our classification framework for positioning their own work within the RSASS body of knowledge and precisely assess the novelty of their (present and future) studies, (iii) the RSASS community can also use the research gaps we identified in this study as an evidence-based plan for working on meaningful and relevant research directions in the coming years.

2. Study design

As mentioned in Section 1, this research follows the systematic mapping research method (Petersen et al., 2015). We opted for conducting a systematic mapping study over other research methods since our goal is to provide a comprehensive overview of research on RSASSs and characterize the approaches to RSASSs. Specifically, systematic literature reviews tend to be narrower in scope and are meant to synthesize evidence (*e.g.,* by comparing how different approaches fare against each other), whereas systematic mapping studies are meant to characterize/structure a certain research area (RSASSs in our case) (Petersen et al., 2008, 2015). Also, we have not opted for a classic "survey paper", as defined by Garousi and Mäntylä (2016) since with this research we aim for a study that is empirical, i.e., it is systematic, rigorous, independently verifiable, and replicable. We also have opted not to conduct a multivocal literature review (Garousi et al., 2019) since it generally has a wider scope and includes a wider set of units of analysis (e.g., blog posts, tools documentation, scientific studies, white papers) as compared to mapping studies, which focus exclusively on the scientific literature.

Our study is designed according to the widely-used guidelines by Kitchenham and Charters (2007). The guidelines define a systematic study of the literature as being composed of three main macro-phases: planning, conducting, and reporting. Fig. 1 presents the design of our study, where Phase 1 corresponds to the *planning* macro-phase, Phases 2–4 correspond to the *conducting* macro-phase, and Phase 5 corresponds to the *reporting* macro-phase. Throughout all phases, we follow well-established guidelines for conducting systematic studies of the literature in general (Kitchenham and Charters, 2007; Kitchenham and Brereton, 2013; Wohlin et al., 2012) and systematic mapping studies in particular (Petersen et al., 2015).

In **phase 1 (Planning)** we produced a *research protocol* describing the context and the needs motivating the study, its goal and research questions, and a plan for carrying out all the activities of this study. We summarize the main points of the context and needs of this study in Section 1 (and also in the description of related work in Section 11), whereas the goal and the research questions of the study are defined in Section 2.1. While planning for this study, we assessed its quality according to the checklist proposed by Petersen et al. (2015). The checklist prescribes all the main actions to carry out in an ideal systematic mapping study and defines a scoring system based on the ratio of the number of actions performed in a given study and the total number of actions within the checklist. Our study achieves a score of 50%; this score is excellent since it is higher than all systematic studies in the literature in 2015 (when the checklist was produced), which had a median score of 33% and an absolute maximum value of 48%. To mitigate potential threats to validity, the research protocol was defined *a priori*, before conducting the study, and it was reviewed by the last author of this work, who has extensive experience in empirical research (including secondary studies) in the role of an evaluator. The evaluator was not involved in the definition of the protocol and was asked to provide open feedback about it, particularly on possible unidentified threats to validity and potential issues in the overall design of the study. All concerns raised during the evaluation have been addressed before proceeding to the next phase.

The *Conducting* macro-phase is about the actual execution of the research protocol. Specifically, **phase 2 (Search and selection)** deals with the identification of (as many as possible) studies that are representative of architecture-based self-adaptation in robotic systems. In this phase, we perform ed a combination of (i) a keyword-based search across the proceedings of relevant publication venues, (ii) a two-step manual selection process, and (iii) snowballing (Wohlin, 2014). The snowballing procedure allows us to expand the set of potentially relevant studies by considering each previously selected primary study and considering those articles either citing or cited by it (Wohlin, 2014). Section 2.2 describes in detail the search and selection phase.

The goal of **phase 3 (Data extraction)** is to collect relevant information about each approach described in each primary study. We collaboratively (i) defined the classification framework, *i.e.,* the set of parameters to be used for comparing primary studies based on our research questions (Kitchenham and Charters, 2007), (ii) analyzed the full text of each primary study, and (iii) populated a data extraction

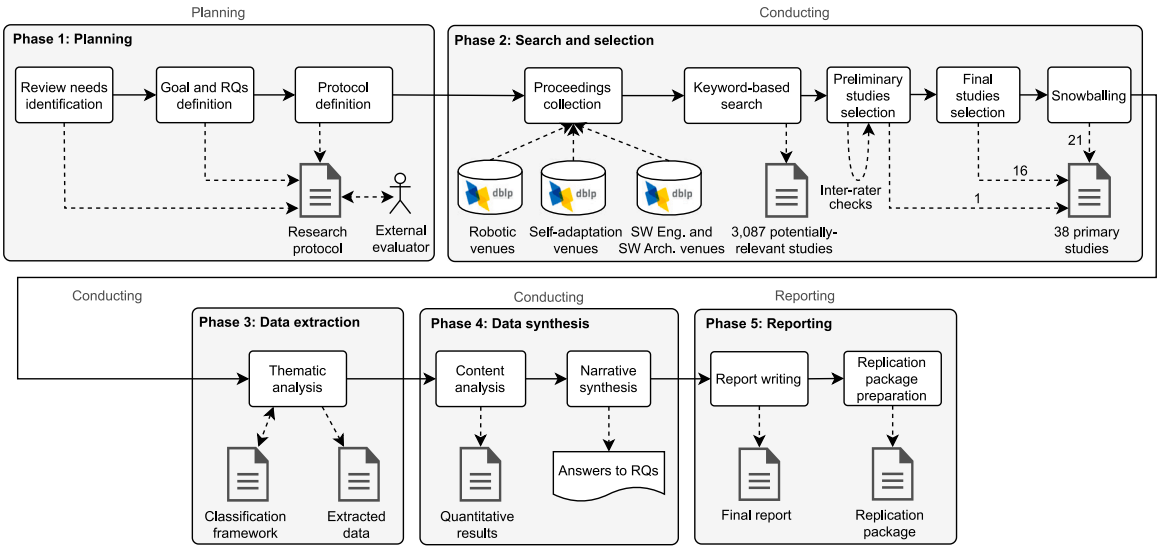


Fig. 1. Overview of the study design.

form with the collected data according to the classification framework. Section 2.3 describes in detail the data extraction phase.

The aim of **phase 4 (Data synthesis)** is to elicit the answers to the research questions of this study. We analyzed the populated data extraction form in accordance with the research questions. This activity involves both quantitative and qualitative analyses and it has been carried out by all the authors. Section 2.4 describes in detail the data synthesis phase.

Finally, in **phase 5 (Reporting)** we produced a *final report* containing a thorough description of the methodological aspects of the study, the main results emerging from the data synthesis phase, and a thorough discussion of their implications. The final report is evaluated by external reviewers and forms the basis of this article. For the sake of independent verification and replication of this study, we have made a complete *replication package* (Replication, 2024) publicly available containing (i) the research protocol we followed, (ii) the raw data we collected and produced across each individual step of the study (across the search and selection, data extraction, and synthesis phases), and (iii) the source code of the scripts we developed for the search, data analysis, and visualization steps.

2.1. Goal and research questions

We formulate the goal of this study according to the Goal-Question-Metric (GQM) framework defined by Caldiera and Rombach (1994). Specifically, the goal of this study is to:

- *Purpose*: identify, classify, and analyze
- *Issue*: the characteristics and evaluation strategies of
- *Object*: existing approaches for architecture-based self-adaptation in robotics software
- *Viewpoint*: from a researcher's point of view

The *issue* of our goal definition is multifaceted and focuses on two main aspects of existing approaches for architecture-based self-adaptation in robotics software: (1) characteristics, decomposed into the sub-aspects of (i) characteristics of the self-adaptive system as a whole, (ii) the feedback loop which manages it, and (iii) the robotic system it adapts, and (2) strategies applied by researchers for evaluating them. We define two main research questions for this study, one for each main aspect.

RQ1 — What are the *key characteristics* of approaches for architecture-based self-adaptation in robotics software? We use the term characteristics to refer to the parameters in the classification framework. *Key* characteristics serve a role similar to themes in the sense of thematic synthesis (Cruzes and Dyba, 2011) in being patterns we observe in the data extracted from the primary studies. Particularly, we are trying to determine the distribution of extracted values across primary studies, as with this knowledge we can draw conclusions about the state of the art. For example, if many primary studies are found to use the same type of adaptation mechanism it may indicate it is effective at tackling problems in this domain.

The main outcome of RQ1 as a whole is a map that classifies existing approaches based on the parameters we devise ourselves as well as those adapted from the modeling dimensions for self-adaptive software systems proposed by Andersson et al. (2009). Within the research sub-questions of RQ1 we make a distinction between the managing system and managed system which are the two of the three main conceptual parts of every self-adaptive system (Weyns, 2019) (the third being the environment), through the research sub-questions RQ1.2 and RQ1.3 respectively. In RQ1.1 those characteristics which describe the system as a whole are covered and the relationship between the two sub-systems.

RQ1.1 — What are the *key characteristics* of the entire self-adaptive system in approaches for architecture-based self-adaptation in robotics software? In this research sub-question, we consider the adaptation goal of the system *i.e.,* the reason it adapts, the quality attributes that are targeted by the system (*e.g.,* reliability, safety, energy efficiency) and the extent to which the managing system and managed system are independent of one another. We provide each of these parameters ourselves. The possible values for the adaptation goal are an open set we fill using the primary studies. For the quality attributes (QAs) we draw from the fixed set of QAs defined by ISO25010:2023 (ISO/IEC, 2023), and we describe the independence of the managing system from the managed system through a fixed set of our own making. Definitions for each of these possible values can be found in Table 4.

RQ1.2 — What are the *key characteristics* of the *managing* systems of architecture-based self-adaptation approaches in robotics software? The main outcome of this research sub-question is a map classifying approaches based on their characteristics with respect to two aspects. The first aspect is the *mechanism* — "what is the reaction of the system towards change" from the modeling dimensions of Andersson et al. (2009) This 'reaction towards change' is the primary goal of the managing system as it adapts the robot in response to changes that affect it. The second aspect is the well-established MAPE-K reference model (Brun et al., 2009; Kephart and Chess, 2003) — a

feedback loop that represents the internal working of the managing system. The 'mechanism' modeling dimension offers 6 parameters: type, organization, scope, duration, timeliness, and trigger. We also append the consideration of an extra parameter 'method' which considers the algorithm or logic backing the managing system. Definitions of each of these parameters are provided in the classification framework (Table 3). As for MAPE-K, its feedback loop consists of four main phases: Monitor, Analyze, Plan, Execute and a shared Knowledge. We collect information about how each approach operates in every phase and the particular format used to represent the managed system within the Knowledge. This format is predetermined to be of an architectural variety by virtue of our selection criteria (in particular E1 in Section 2.2.3). Such information allows us to shed light on aspects related to how the managed system is being monitored, how the monitored information is being analyzed by the managing system, how the managing system plans for necessary adaptations, how the managing system enacts/executes the necessary adaptations, and how it internally represents important information about the managed system.

RQ1.3 — What are the *key characteristics* of the *managed* systems of architecture-based self-adaptation approaches in robotics software? As RQ1.3 concerns the managed systems, and in this study these are all robotics systems, it in essence provides the key characteristics of the robotic platforms involved in the given approaches. The results we obtain when answering this research sub-question can support researchers in making informed decisions and mitigating risk when selecting the software and hardware platforms to use in the context of future research efforts into RSASSs. The main outcome of this research question is a map of the types of software platforms and robots targeted by researchers when realizing architecture-based self-adaptation approaches for robotic systems and their characterization. Specifically, we elicit three aspects. Firstly, the details about the missions the robots are tasked with. Secondly, the modeling dimensions of change, *i.e.,* "the cause for adaptation" and effect, *i.e.,* "the impact of adaptation upon the system" (Andersson et al., 2009). Thirdly, the software platform backing the robotic system (*e.g.,* ROS2 — a component-based library which has become the de facto standard for robotic software Macenski et al., 2022) with a special emphasis on the software components that enable self-adaptation for the robotic system, and the models of robots (its hardware) (*e.g.,* Turtlebot 3,[1] Clearpath Boxer[2]). For the first aspect we gather what kind of mission the robot is tasked with, as well as how this evolves at runtime. For the second aspect we consider the source, type, anticipation, and frequency of 'change', as well as the criticality, predictability, overhead, and resilience of 'effect'. Definitions for each of these parameters are provided in Table 3. Lastly, for the third aspect we consider two parameters – the software platform, and the type of robot model – as an open set which we fill with the data extracted from the primary studies.

RQ2 — What are the *evaluation strategies* of approaches for architecture-based self-adaptation in robotics software? Evaluating self-adaptive software systems is known to be a difficult problem (Gerostathopoulos et al., 2021), primarily due to the intrinsic characteristics of self-adaptive systems, such as the presence of feedback loops to realize adaptation, the uncertainties to be taken care of while designing evaluative experiments, and the often non-trivial distinction between managed and managing systems when running these experiments. The main outcome of this research question is a comprehensive map of the various strategies put in place by researchers when evaluating an approach for architecture-based self-adaptation for robotics software. The first aspect that we consider in this research question is whether studies are reporting on the deployment of real robotic systems in the field or whether the robotic system is simulated. Another aspect that we consider is the realism of the robotic system. Finally, we also consider how the evaluation is carried out in terms of which aspects/properties of the approach are evaluated and whether a baseline is considered.

By answering the two research questions above we provide a detailed and up-to-date overview of the state of the art on architecture-based self-adaptation for robotics software. Such overview can be used by researchers as a solid foundation for (i) classifying existing research efforts, (ii) positioning their own approaches with respect to the state of the art, and (iii) identifying current research gaps to be targeted in future research. The identified research questions drive the whole study, with a special influence on (i) the search and selection, (ii) data extraction, (iii) synthesis, and (iv) reporting and discussion of the main findings.

2.2. Search and selection

The goal of this phase is to identify a representative set of studies presenting approaches for architecture-based self-adaptation in robotics software (Kitchenham and Brereton, 2013). In accordance with community guidelines (Kitchenham and Charters, 2007; Wohlin et al., 2012; Petersen et al., 2015), in the remainder of this article we will refer to the selected studies as *primary studies*.

As shown in Fig. 1, our search and selection phase is composed of five internal stages. For the sake of replicability and for having full control on the overall process, the stages are carried out sequentially and independently from each other; the only inputs for each stage are the outputs produced by the previous one. In this subsection we present each stage of our search and selection process. We do so in chronological order, starting with the collection of potentially relevant studies, the process of filtering through these, selecting the primary studies, and finally snowballing the initially-selected set.

2.2.1. Proceedings collection

As reported in Section 2.1, the object of this study is the body of scientific literature presenting approaches for architecture-based self-adaptation in robotics software. Essentially, we are interested in those scientific studies whose scope falls within the intersection of three distinct research areas: (i) software engineering with a specific focus on its architecture, (ii) self-adaptive systems, and (iii) robotics. In this stage, we aimed to identify publication venues that are specific to each of those research areas and collect all scientific studies belonging to those publication venues.

According to well-accepted guidelines on systematic mapping studies (Petersen et al., 2015), the key to having a successful search process when applying snowballing (which we describe in Section 2.2.5) is to have a starting set of heterogeneous and high-quality primary studies. Accordingly, we constructed a list of top publication venues in software architecture, self-adaptive systems, and robotics. Only studies published at these venues are considered in the next stage of this study. Table 1 presents the selected publication venues. We decided to consider those venues based on our extensive experience in each of the three research communities.

As publications database, we chose the *DBLP computer science bibliography*.[3] More specifically, we make use of the publicly available snapshot of the whole DBLP database dated June 6th 2024. Using this snapshot contributes to the reproducibility of this study, as opposed to the usage of other academic search engines, such as Google Scholar, whose search algorithms tend to be more opaque and less deterministic. Also, previous experience tells us that DBLP is a reliable publication database for studies at the intersection between the software engineering and robotics research domains (Albonico et al., 2023). We collected all studies included in all publication venues in Table 1 within the 2011 June 2024 time frame (inclusive). The time frame is

[1] https://www.robotis.us/turtlebot-3.

[2] https://clearpathrobotics.com/boxer.

[3] https://dblp.uni-trier.de/.

Table 1
Publication venues targeted in the proceedings collection stage.

Acronym	Name	Type	#Studies
Software architecture and software engineering			
ICSA	International Conference on Software Architecture	Conf.	512
ECSA	European Conference on Software Architecture	Conf.	718
WICSA	Working IEEE/IFIP Conference on Software Architecture	Conf.	221
CBSE	International Symposium on Component-Based Software Engineering	Conf.	104
QOSA	International Conference on Quality of Software Architectures	Conf.	89
JSA	Journal of Systems Architecture	Journal	1402
JSS	Journal of Systems and Software	Journal	2582
TSE	Transactions on Software Engineering	Journal	1407
SoSym	Software and Systems Modeling	Journal	950
Self-adaptive systems			
SEAMS	International Conference on Software Engineering for Adaptive and Self-Managing Systems[b]	Conf.	290
SASO	International Conference on Self-Adaptive and Self-Organizing Systems	Conf.	582
ACSOS	International Conference on Autonomic Computing and Self-Organizing Systems	Conf.	276
ICAC	International Conference on Autonomic Computing	Conf.	314
TAAS	ACM Transactions on Autonomous and Adaptive Systems	Journal	249
Robotics			
ICRA	International Conference on Robotics and Automation	Conf.	11764
IROS	International Conference on Intelligent Robots and Systems	Conf.	11817
RSS	Robotics Science and Systems	Conf.	891
IRC	IEEE International Conference on Robotic Computing	Conf.	535
T-RO	IEEE Transactions on Robotics	Journal	1907
RAL	IEEE Robotics and Automation Letters	Journal	7099
IJRR	International Journal of Robotics Research	Journal	1045
SCIROB	Science Robotics	Journal	654
JOSER	Journal of Software Engineering for Robotics[a]	Journal	35

[a] The JOSER journal has been sourced externally since it is not indexed in DBLP.
[b] Prior to 2021, SEAMS was indexed in DBLP under conf/icse rather than conf/seams.

chosen as (i) 2011 corresponds with the first edition of the SEAMS conference, *i.e.,* the primary conference on self-adaptive systems and (ii) we executed the search query in June 2024. The choice of using the SEAMS conference as lower bound of our time frame can be justified by the fact that a study's existence in the research area of self-adaptive systems is considered key in identifying potentially relevant studies for this research and ergo it is reflected in the decision of considering studies published from 2011 on. This stage leads to the identification of 45 443 potentially relevant studies.

2.2.2. Keyword-based search

Due to the magnitude of potentially relevant studies, particularly for the ICRA and IROS conferences which accounted for about 2000 studies per year, it was necessary to introduce a keyword filter on the titles of the studies. For the filtering, we defined a set of search terms for each of the three research areas considered in this study based on the expertise of the authors in each research area. Specifically, to determine if a potentially relevant study is related to software architecture we used the term `architect*`, for self-adaptive systems we used the term `self-* OR adapt* OR reconfigur*`, and for robotics we used the term `robot*`. Then, we applied those search terms in groupings within the venues of each research area, as follows:

- Search string used in software architecture venues: `(self-* OR adapt* OR reconfigur*) OR robot*`.
- Search string used in software engineering venues[4]: `(self-* OR adapt* OR reconfigur*) OR robot* OR architect*`.
- Search string used in self-adaptive venues: `architect* OR robot*`.
- Search string used in robotics venues: `architect* OR (self-* OR adapt* OR reconfigur*)`.

In short, for each venue within a specific research area, we crafted the search query by applying a logical OR between the terms that do not belong to that area and then we apply it to the titles of studies published within the venue. For example, for a self-adaptive study to make it through the filter, its title should include either the keyword "architect*" or "robot*". It should be noted that we trialed stricter search strings, *e.g.,* using logical AND operators, but this resulted in little (magnitude of 1) to no studies satisfying the search string. This stage led to a total of 3087 potentially relevant studies, of which 231 come from software architecture venues, 107 come from self-adaptive systems venues, and 2314 come from robotic venues.

2.2.3. Preliminary studies selection

The goal of this stage and the next one is to filter the 3087 potentially relevant studies emerging from the keyword-based search for obtaining only those studies that are relevant for answering our research questions.

The two selection stages involved four researchers and required a manual assessment of the studies being analyzed. In order to avoid possible biases due to different interpretations of software architecture, self-adaptive system, and robotic system, all authors of this study agreed on common definitions to be used during the selection phase. The definitions are reported below:

- **Software architecture**: we use the definition presented by Bass et al. (2003), where a software architecture is defined as "the set of structures needed to reason about the system. These structures comprise software elements, relations among them, and properties of both".
- **Self-adaptive system**: we use the definition presented by Weyns (2020), where a self-adaptive system is defined as "one which handles changes and uncertainties autonomously and has two distinct sub-systems, a managed system responsible for domain concerns, and a managing system responsible for adaptation concerns".
- **Robotic system**: we determine whether a software system can be considered robotic using the 'Springer Handbook of Robotics' by Siciliano and Khatib (2016). Within the handbook robots are defined as "operating in the three-dimensional world as a

[4] Specifically, JSS, TSE, and SoSym.

machine endowed with the capacity to interpret and to reason about a task and about its execution, by intelligently relating perception to action".

The definitions serve not only to ensure that the four researchers have a common understanding of the topic being studied, but also to provide transparency for potential replications of this research. Moreover, the three definitions, together with the goal and research questions presented in Section 2.1, guide the definition of the inclusion and exclusion criteria to be applied when considering each potentially relevant study. As recommended in guidelines for systematic studies (Kitchenham and Charters, 2007; Petersen et al., 2015), the selection criteria are defined *a priori*. In this study we used the following selection criteria:

Inclusion Criteria:

I1: Studies focusing on robotic systems, as defined above.

I2: Studies that involve software architecture, as defined above.

I3: Studies focusing on self-adaptive systems, as defined above.

I4: Peer-reviewed scientific publications (journal papers, workshop papers, book chapters, conference papers).

I5: Studies published between 2011 and 2024 (inclusive).

Exclusion Criteria:

E1: Studies on self-adaptation in which the managed system is **not** changed based on an architectural model/representation of it.

E2: Studies written in languages different than English.

E3: Studies not available as full-text documents.

E4: Secondary/tertiary literature studies.

E5: Studies in the form of theses or project reports.

Clearly, due to our use of a list of selected venues and DBLP, criteria I4, E2 and E5 are only pertinent for the snowballing stage detailed in Section 2.2.5. A potentially relevant study is added to the set of primary studies if it satisfies *all* inclusion criteria and *none* of the exclusion criteria. As suggested by Wohlin et al. (2012), in order to be reasonably confident about our selection procedure, we piloted it via a preliminary study selection step. In this step we focused on a subset of the 3087 potentially relevant studies; we selected a random sample of 120 potentially relevant studies – 40 for each research area – and then the four researchers involved in this stage proceeded with the application of the selection criteria, as follows. The first researcher assesses all 120 studies, while the other three researchers assess a disjoint random sample of 40 studies each. Then the inter-rater agreement is calculated between the first and the other three researchers using Cohen's kappa coefficient. We repeated this pilot process until we had achieved a Cohen's kappa score of >0.81 between the first researcher and each of the other authors respectively as this signifies near-perfect agreement (McHugh, 2012). We reached near-perfect agreement after two iterations over 240 potentially relevant studies.

When assessing a potentially relevant study, we applied the adaptive reading technique (Petersen et al., 2008). Adaptive reading consists in starting from the title of a potentially relevant study and discard it if it clearly does not fall within the scope of this study; if the researcher is still in doubt, then they analyze the abstract, introduction, and conclusion sections (if present); finally, the study is further assessed by considering its full text, and a final decision about its inclusion is taken. This procedure allowed us to be reasonably objective about the selection and to do it within a reasonable amount of time, as reading the full text of clearly excluded studies is not necessary.

2.2.4. Final studies selection

Once we were reasonably sure about the alignment and objectivity of the four researchers involved in the selection phase, we proceeded to selecting the primary studies from all the remaining potentially relevant studies. This stage was carried out with the same distribution of labor – a 3:1:1:1 split – and resulted in a final set of 14 primary studies. Table 2 presents all selected primary studies, within the table their provenance is indicated as stemming from either the selected venues, or snowballing (which is covered in Section 2.2.5).

Note: Fig. 1 shows 38 primary studies, while Table 2 has 37, this is due to overlapping data with P17 and is elaborated upon in 2.3.

2.2.5. Snowballing

To mitigate the potential bias coming from the manual selection of the publication venues done in stage one (*i.e.,* proceedings collection), we also perform a snowballing process (Wohlin, 2014). Snowballing allows us to expand the set of primary studies by considering each of the 13 primary studies and selecting those papers that are either cited by it (backward snowballing) or citing it (forward snowballing) (Wohlin, 2014). In this research, we carry out a 1-step snowballing procedure, both backward and forward. To perform the backwards snowballing we simply extracted the references from each primary study. For the forward snowballing we used Google Scholar[5] which has a 'cited by' feature, as we saw this being more complete than the forward citations linked by individual publishers.

The selected 13 primary studies are used as starting set for the snowballing. As shown in Table 2, such a set of 13 primary studies is a good candidate to be used as starting set for snowballing since, by following the advice of Petersen et al. (2015), (i) the studies belong to complementary publication venues, which tend to be targeted by different academic communities, (ii) the studies are relevant for our research questions by design, (iii) the number of studies is not too small, and (iv) the studies cover multiple authors, years of publication, and publishers. Particularly, the number of studies to satisfy (iii) is relative to the specificity of a study's focus. Given our study combines three disciplines, we consider a quantity with an order of magnitude 2 to be sufficient for our quite specific focus.

The snowballing activity led to 724 additional potentially relevant studies in total, to which we applied the same selection criteria used in the previous stages. This final stage led to the inclusion of 21 additional studies that met our selection criteria, giving our total of 37 primary studies.

2.3. Data extraction

In this phase, we analyzed the *full text* of the selected primary studies and collect relevant information for answering our research questions in the subsequent data synthesis phase.

In order to have a rigorous and replicable data extraction process and to facilitate elicitation of the main findings during data synthesis, a structured **classification framework** has been designed. The classification framework is composed of five main facets: the first facet is about demographics of the primary studies (*i.e.,* title, authors, publication venue, year of publication) and the other four facets correspond to the two research questions of this study. As suggested by Wohlin et al. (2012), the initial classification framework has been defined *a priori* and iteratively refined. Within each facet of the classification framework, we define a set of parameters, each of them considering a specific aspect of an architecture-based self-adaptation approach for robotics. Each parameter of the classification framework can have either a finite or an open set of possible values. When considering a parameter with a finite set of possible values (*e.g.,* the mission definition parameter, which can have only two values: `static` and `dynamic`), we directly assign one or more of the possible values to each primary study. When considering parameters with an open set of possible values (*e.g.,* the types of robots), we (i) collect notes about each primary study, enriched with fragments of text from the study itself and (ii) conduct *iterative open coding sessions* to categorize the primary studies according to emerging high-level categories. Upon the emergence of new possible values for a parameter, the classification framework is refined; in such

[5] https://scholar.google.com/.

Table 2
Primary studies of this research.

ID	Authors, title, and venue	Year
Provenance: Selected venues		
P1	F. Dietrich et al., Dynamic distribution of robot control components under hard realtime constraints — Modeling, experimental results and practical considerations, JSA (Dietrich et al., 2013)	2013
P2	D. de Leng and F. Heintz, Towards adaptive semantic subscriptions for stream reasoning in the robot operating system, IROS (de Leng and Heintz, 2017)	2017
P3	Y. Cui et al., ReFrESH: A self-adaptation framework to support fault tolerance in field mobile robots, IROS (Cui et al., 2014)	2014
P4	J. C'amara et al., Software architecture and task plan co-adaptation for mobile service robots, SEAMS (Cámara et al., 2020)	2020
P5	D. Doose et al., MAUVE Runtime: A Component-Based Middleware to Reconfigure Software Architectures in Real-Time, IRC (Doose et al., 2017)	2017
P6	S. Niemczyk and K. Geihs, Adaptive Run-Time Models for Groups of Autonomous Robots, SEAMS (Niemczyk and Geihs, 2015)	2015
P7	S. Zaman et al., An integrated model-based diagnosis and repair architecture for ROS-based robot systems, ICRA (Zaman et al., 2013)	2013
P8	D. Kent et al., Localization Uncertainty-driven Adaptive Framework for Controlling Ground Vehicle Robots, IROS (Kent et al., 2020)	2020
P9	L. Gherardi and N. Hochgeschwender, RRA: Models and tools for robotics run-time adaptation, IROS (Gherardi and Hochgeschwender, 2015)	2015
P10	C. Wang et al., How to secure autonomous mobile robots? An approach with fuzzing, detection and mitigation, JSA (Wang et al., 2021)	2021
P11	Y. Cui et al., Real-time software module design framework for building self-adaptive robotic systems, IROS (Cui et al., 2015a)	2015
P12	C. Eymuller et al., RealCaPP: Real-Time Capable Plug & Produce Service Architecture for Distributed Robot Control, IRC (Eymuller et al., 2023)	2023
P13	C. Heinzemann et al., Towards modeling reconfiguration in hierarchical component architectures, CBSE (Heinzemann et al., 2012)	2012
P14	D. Cooray et al., Proactive Self-Adaptation for Improving the Reliability of Mission-Critical, Embedded, and Mobile Software, TSE (Cooray et al., 2013)	2013
P15	D. Brugali, Runtime reconfiguration of robot control systems: a ROS-based case study, IRC (Brugali, 2020)	2020
P16	S. Pradhan et al., Achieving resilience in distributed software systems via self-reconfiguration, JSS (Pradhan et al., 2016)	2016
Provenance: Snowballing		
P17	I. Gerostathopoulos et al., Tuning self-adaptation in cyber–physical systems through architectural homeostasis, JSS (Gerostathopoulos et al., 2019)	2019
P18	Y. Cui et al., A mechanism for real-time decision making and system maintenance for resource constrained robotic systems through ReFrESH, Autonomous Robots (Cui et al., 2015b)	2015
P19	D. Bozhinoski et al., A Modeling Tool for Reconfigurable Skills in ROS, RoSE (Bozhinoski et al., 2021)	2021
P20	S. Niemczyk et al., ICE: self-configuration of information processing in heterogeneous agent teams, SAC (Niemczyk et al., 2017)	2017
P21	Y. Zou and J. Bai, Effective Crash Recovery of Robot Software Programs in ROS, ICRA (Zou and Bai, 2021)	2021
P22	S. Zaman et al., Fault Detection Using Sensors Data Trends for Autonomous Robotic Mapping, ICEET (Zaman et al., 2019)	2019
P23	S. Loigge et al., A Model-Based Fault Detection, Diagnosis and Repair for Autonomous Robotics systems, OAGM&ARW Joint Workshop (Loigge et al., 2017)	2017
P24	D. Brugali et al., Model-Based Development of QoS-Aware Reconfigurable Autonomous Robotic Systems, IRC (Brugali et al., 2018)	2018
P25	N. Hochgeschwender et al., Graph-based software knowledge: Storage and semantic querying of domain models for run-time adaptation, SIMPAR (Hochgeschwender et al., 2016)	2016
P26	D. Bozhinoski and J. Wijkhuizen, Context-based navigation for ground mobile robot in semi-structured indoor environment, IRC (Bozhinoski and Wijkhuizen, 2021)	2021
P27	Sanchez et al., Context-Based Adaptation of In-Hand Slip Detection for Service Robots, IFAC (Sanchez et al., 2016)	2016
P28	Esfahani et al., Taming uncertainty in self-adaptive software, ESEC/FSE (Esfahani et al., 2011)	2011
P29	D. de Leng and F. Heintz, DyKnow: A dynamically reconfigurable stream reasoning framework as an extension to the robot operating system, SIMPAR (de Leng and Heintz, 2016)	2016
P30	Y. Cui et al., A self-adaptation framework for resource constrained miniature search and rescue robots, SSRR (Cui et al., 2012)	2012
P31	Jamshidi et al., Machine Learning Meets Quantitative Planning: Enabling Self-Adaptation in Autonomous Robots, SEAMS (Jamshidi et al., 2019)	2019
P32	Y. Park et al., A task-based and resource-aware approach to dynamically generate optimal software architecture for intelligent service robots, Software: Practice and Experience (Park et al., 2012)	2012
P33	A. Lotz et al., Managing Run-Time Variability in Robotics Software by Modeling Functional and Non-functional Behavior, EMMSAD (Lotz et al., 2013)	2013
P34	G. Silva et al., SUAVE: An Exemplar for Self-Adaptive Underwater Vehicles, SEAMS (Silva et al., 2023)	2023
P35	A. Valdezate et al., RuVa: A Runtime Software Variability Algorithm, IEEE Access (Valdezate et al., 2022)	2022
P36	D. Brugali, Modeling variability in self-adapting robotic systems, RAS (Brugali, 2023)	2023
P37	A. Hristozov et al., Resilient Architecture Framework for Robotic Systems, ICAI (Hristozov et al., 2022)	2022

cases, the previously extracted data for the updated parameter is updated in accordance with the new set of possible values. The resulting classification framework is shown in Table 3, including all parameters and their descriptions, possible values, and targeted research questions.

The classification framework is the basis of the data extraction form, *i.e.,* a spreadsheet we use to store the data extracted from each primary study. In such a spreadsheet, rows represent primary studies and columns represent the parameters of the classification framework.

Four researchers were involved in the data extraction phase, as follows. We assigned two researchers to each parameter of the classification framework; researchers are assigned to parameters based on their expertise and different researchers can be assigned to different parameters. Then, for each parameter, the two researchers independently analyzed the primary studies (half of the primary studies each), followed by a reconciliation of the obtained results and a final check with the involvement of a third researcher. This process ends when all primary studies are analyzed.

During the analysis, we identified one study (P17) which explicitly states it is published as a direct extension of another primary study (Gerostathopoulos et al., 2016). We decide to defer to the most extended version to prevent a bias through over-representation of their identical data. Further, we have identified that several groupings of papers belong to one approach but cover it from distinct angles and/or iterations, and do not explicitly indicate being extensions of one another. The groups are as follows: (P2, P29), (P3, P11, P18,

Table 3
The classification framework of this study (parameters about demographics omitted for readability).

Parameter	Type	Possible values	Description
Key characteristics of the self-adaptive system as a whole (RQ1.1)			
Adaptation goal	Open	Recover from errors/faults, Optimize resource usage, ...	The objective(s) of the self-adaptation logic.
Quality attributes	Fixed	Reliability, Safety, Performance Efficiency, ...	The system/software quality attributes targeted by the robotic system, adapted from ISO/IEC (2023).
Managing system independence	Fixed	Detachable, Inseparable, Requires Representation	The extent to which the managing system is independent of the managed system (Weyns, 2019).
Key characteristics of the managing system (RQ1.2)			
Mechanism — Method	Open	Constraint solving/Model checking, Ontological reasoning, ...	What kind of specific algorithm/logic is being used to reason about adaptations.
Mechanism — Type	Fixed	Structural, Parametric	Whether adaptation is related to the parameters of the system components or the structure of the system, adapted from Andersson et al. (2009).
Mechanism — Organization	Fixed	Centralized, Decentralized	Whether the adaptation is done by a single component or distributed among several components, adapted from Andersson et al. (2009).
Mechanism — Scope	Fixed	Local, Global	Where in the system is the adaptation localized, ad. from Andersson et al. (2009).
Mechanism — Duration	Fixed	Short, Medium, Long	How long the adaptation lasts, adapted from Andersson et al. (2009).
Mechanism — Timeliness	Fixed	Best effort, Dependent, Guaranteed	Whether the time period for performing self-adaptation can be guaranteed, adapted from Andersson et al. (2009).
Mechanism — Trigger	Fixed	Event-triggered, Time-triggered	Whether the change that triggers adaptation is associated with an event or a time slot, adapted from Andersson et al. (2009).
MAPE-K — Monitoring	Fixed	Environment, Managed System, Mission	How is the managed system being monitored by the managing system.
MAPE-K — Analysis	Open	Comparison to threshold(s), Analyzing/Aggregating data, ...	How the monitored information is being analyzed by the managing system. Adapted from Weyns et al. (2023)
MAPE-K — Planning	Open	Determining the optimal choice, Relying on design-time rules, ...	How the managing system plans for the adaptations.
MAPE-K — Execution	Open	Component re-deployment, Reparameterization of Component(s), ...	How the managing system enacts/executes adaptations.
MAPE-K — Knowledge	Open	Knowledge Representation, Component Model, ...	How the managing system represents the managed system it adapts.
Key characteristics of the managed system (RQ1.3)			
Mission	Open	Navigation, Industrial manipulation, Search-and-rescue, ...	The type of robotic mission supported by the approach.
Mission — Evolution	Fixed	Static, Dynamic	Whether the mission as defined at the start of the operation changes during the operation or not, adapted from Andersson et al. (2009).
Change — Source	Fixed	Internal, External	The location of the source of the change, adapted from Andersson et al. (2009).
Change — Type	Fixed	Functional, Non-functional, Technological	The nature of change, adapted from Andersson et al. (2009).
Change — Anticipation	Fixed	Foreseen, Foreseeable, Unforeseen	Whether the change can be predicted, adapted from Andersson et al. (2009).
Change — Frequency	Fixed	Rare, Infrequent, Frequent, Random	Contingent on foreseen change, how often a particular change occurs, adapted from Andersson et al. (2009).
Effect — Criticality	Fixed	Harmless, Mission-critical, Safety-critical	Impact upon the system in case the self-adaptation fails, adapted from Andersson et al. (2009).
Effect — Predictability	Fixed	Deterministic, Non-deterministic	Whether the consequences of the adaptation can be predicted, adapted from Andersson et al. (2009).
Effect — Overhead	Fixed	Significant, Insignificant, Failure, Dependent	The impact of system adaptation upon the quality of services of the system, adapted from Andersson et al. (2009).
Effect — Resilience	Fixed	Resilient, Irresilient, Vulnerable, Dependent	The persistence of service delivery that can justifiably be trusted, when facing changes, adapted from Andersson et al. (2009).

(*continued on next page*)

Table 3 (*continued*).

Software platform	Open	ROS1, ROS2, …	The software framework, middleware, platform used to implement the approach, specifically the components that enable self-adaptation for the robotic system.
Type of robots	Open	Turtlebot 3, Pioneer3-DX, …	The used type of robot/model of robot.
Evaluation strategies (RQ2)			
System deployment	Fixed	Simulated, Real, Combined	Whether the evaluation is performed in simulation, by deploying real robots, or a combination thereof.
System realism	Fixed	Real, Synthetic	Whether the evaluation sees the robots in a realistic context (*i.e.,* on the field) or in a reduced context (*i.e.,* in the lab).
Evaluation metric	Open	Quality, Mission Performance, Overhead, ...	Which aspects are evaluated.
Evaluation depth	Fixed	No evaluation, Showcase, Experiment	How the evaluation is carried out.
Replication package	Fixed	Present, Absent	Whether the authors provide a means to reproduce their evaluation.

P30), (P4, P31), (P6, P20), (P7, P22, P23), and (P15, P24, P36). While the studies within these groups undoubtedly have some data that is overlapping, we decide to consider them separately for the sake of the data that is not. The list of groupings provided can serve as an aid to any future researcher needing to draw further conclusions from our report in removing bias. This leads us to a final set of 37 primary studies considered in our data synthesis phase. Overall, the data extraction phase led to the definition of 32 parameters and a total of 1184 extracted data points.

2.4. Data synthesis

The goal of this phase is to elicit relevant findings and implications for researchers, which are then used to substantiate the answers to the research questions of this study. By following the best practices accumulated while working on previous secondary studies (David et al., 2023; Albonico et al., 2023; Di Francesco et al., 2019; Franzago et al., 2018), our data synthesis phase is structured into two main stages: (i) vertical synthesis (see Section 2.4.1) and (ii) horizontal synthesis (see Section 2.4.2). Sections 4 through 8 present the main results emerging from our vertical and horizontal syntheses.

2.4.1. Vertical synthesis

In the vertical synthesis stage, we went through each individual parameter of the classification framework described in Section 2.3 and performed a parameter-specific *content analysis session* (Franzosi, 2010). The content analysis session is meant to obtain a quantitative assessment of the extracted data (*e.g.,* the frequency in which the various quality attributes such as performance, energy efficiency, safety, and reliability are considered across all primary studies) (Franzosi, 2010). To do so, depending on the specific parameter to be analyzed, we apply descriptive statistics and create bar plots and tables for a better understanding of the extracted data and its underlying patterns. Then, after the content analysis sessions were completed for all parameters of the classification framework, we applied the *narrative synthesis* method (Popay et al., 2006). Narrative synthesis refers to the method of synthesizing research in the context of systematic reviews where a textual narrative summary is adopted to explain the characteristics of the primary studies (Popay et al., 2006); in this specific phase of our study, we applied the narrative synthesis on each individual parameter of the classification framework. Two researchers performed a first iteration of narrative synthesis on all parameters, and then two other researchers carried out a second iteration for expanding and refining the obtained findings.

2.4.2. Horizontal synthesis

In the horizontal synthesis stage, we analyzed the extracted data to explore possible relations across the values of *pairs* of different parameters of the classification framework. The first step of the horizontal synthesis consisted of the (automatic) creation of a contingency table for all possible pairs of parameters of the classification framework; this step led to a total of 39 contingency tables. Two researchers then collaboratively created a set of pairs of parameters whose relationship is deemed potentially relevant to be investigated (*e.g.,* which quality attributes are considered when applying a local/distributed self-adaptation mechanism). Then, we iteratively analyzed the contingency table of each of the 39 potentially relevant pairs and produce notes about the main emerging results.[6] We filter out all the results which are either not supported by a sufficient number of data points or not revealing any evident pattern. This filtering step was performed manually and collaboratively by two researchers in a pair-by-pair fashion, until a full agreement about the inclusion of each pair is reached. Finally, we also performed narrative synthesis during this stage; this time the emerging findings are based on the contingency tables resulting from the previous steps.

3. Demographics

This research considers 37 primary studies in total. The distribution of our primary studies over the years is depicted in Fig. 2. The lack of any noticeable upward or downward trend indicates there is consistent interest in the field of RSASS research. Of the 21 primary studies found through snowballing, only 6 were published at venues considered in the initial search query. This indicates two things: (i) snowballing was effective in finding studies at venues outside of our list of selected venues in Table 1, and (ii) the title keyword-based search likely did not exclude many studies from the selected venues. The first is clear from the remaining 15 studies which are from non-selected venues. The second is indicated by the fact that the snowballed papers were not subject to keyword-based search. were many studies from the selected venues to appear during snowballing it may indicate that the keywords chosen were too strict and excluded relevant papers from those venues.

It is interesting to consider which type of venue contributes most to RSASS research from the demographics. There is a 2:9:5 ratio between self-adaptive systems, robotics, and software engineering/architecture venues respectively among the primary studies with the selected venues as their provenance. Meanwhile, there were about 21 times as many potentially-relevant robotics studies relative to self-adaptive systems and about 3 as many relative to software architecture. Therefore,

[6] All contingency tables, the pairs of potentially relevant parameters, and our collected notes are available in the replication package (Replication, 2024) of this study for independent verification.

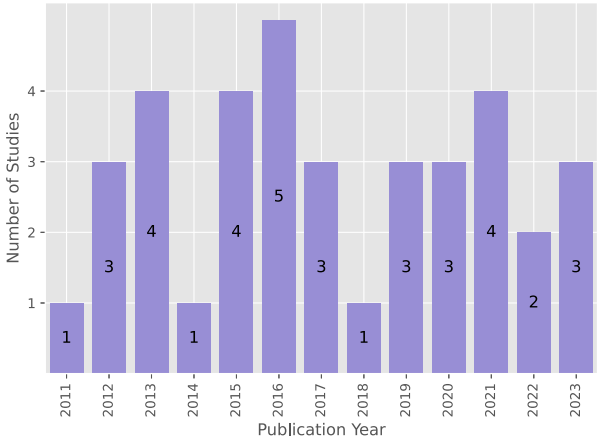


Fig. 2. Occurrence of primary studies by publication year.

robotics venues contribute fewer RSASS studies despite the larger population. One possible explanation is the larger breadth of robotics research compared to the more specialized venues and therefore narrower research areas of self-adaptive systems and software architecture. However, as we have only selected primary studies which are interdisciplinary between all three disciplines, we cannot derive any conclusions as to the prevalence of software architecture or self-adaptive systems research in isolation in robotics venues. This would be a study on its own to determine how much of a focus there is on software at robotics venues, which would be interesting to determine the prevalence of RSASS relative to that. For example, it would answer questions such as what the most common research related to software engineering within robotics venues is. It would also allow one to characterize the prevalence of research into RSASS by determining how common it is among general software engineering research at robotics venues.

4. Key characteristics of the entire self-adaptive system (RQ1.1)

This section covers the results of applying the of the classification framework described in Table 3 and its associated definitions (Table 4) to each primary study. Particularly, we focus on those parameters which characterize the entirety of a self-adaptive system. These are those aspects of the system which only emerge by virtue of the co-existence of a managed system (in our case the robotic system) and managing system (the approaches presented in primary studies). In other words, we provide an answer to RQ1.1.

Note: For all of the subsections (4.X) within this Section (4), and the three to follow (5, 6, 7) the data we report is either on singular occurrences in each of the primary study, or multiple occurrences across the primary studies. For example, Section 4.3 about Managing System Independence has singular occurrences as each study only has *one* managing system which has some degree of independence. However, Section 4.2 on quality attributes reports on occurrences across the studies, as each primary study may see its system target multiple quality attributes. We indicate the case of multiple occurrences across studies with an asterisk (*) to avoid the need to mention this fact about the results every single time.

4.1. Adaptation goal*

Fig. 3(a) depicts the different reasons for architectural self-adaptation we determine from the studies. The most common adaptation goal is that of *recovering from errors/faults* with 16 occurrences. For example, the approach proposed in P23 is designed to detect and repair faulty components by analyzing trends in the data these produce. The second most common adaptation goal is *dealing with environmental changes* with 13 occurrences. For example, the robot in P35 adds a new mapping component as the space around it widens and the number of environmental features to serve as references are reduced. A further 8 times the adaptation goal is to *optimize resource usage*. P26 serves as the system adapts by utilizing different marker detection algorithms based on available memory seeking to conserve memory usage. Less common goals are *optimizing system performance* (towards a mission), *changing functional behavior* of the robot (6 occurrences each), as well as *keep meeting quality requirements at runtime* with 5 occurrences. An example of *changing functional behavior* is P17 where the functional behavior of the robot is adapted as it decides between searching for tiles to clean, cleaning said tiles, or going to charge itself. The other two categories help distinguish between primary studies where adaptation aims to improve attributes related to the performance of the mission versus primary studies where adaptation aims to keep those attributes at acceptable levels (but not always improve them). An example of the latter is P31, where the system switches between alternative implementations of tasks to keep fulfilling the quality requirements of the mission at hand (*e.g.,* safety). Lastly, 1 study – P10 – has the unique goal of *recovering from attacks*; in the sense of cybersecurity.

4.2. Quality attributes*

As depicted in Fig. 3(b), we consider which quality attributes (QAs) are targeted by the self-adaptive systems. We identify these QAs in line with the definitions of the ISO25010:2023 standard (ISO/IEC, 2023). In addition to the main categories of QAs shown in the figure, we extract data according to the subcategories of each QA, as described in the ISO25010. It should be noted that when there are different subcategories of a QA (*e.g.,* functional completeness vs. functional correctness which both fall under *functional suitability*), we count the parent QA twice for that study.

The most common QA targeted is *performance efficiency* (33) consisting of the subcategories of *resource utilization* (18), *time behavior* (13), and *capacity* (2). Interestingly, in 7 primary studies the authors targeted energy as the resource to be optimized. Specifically, energy efficiency is targeted in studies P4, P9, P18, P19, P28, P31, and P33. P25 serves as an example of *resource utilization* where the considered type of resource is memory and memory usage is considered as a constraint while adapting. P1 serves as an example of *time behavior* in that its use case focuses on meeting real-time constraints of its components. Lastly, P6 and P20 deal with the performance *capacity*, as they deal with ensuring the throughput of information between components.

The second most common QA is *reliability* (18) with its subcategories *recoverability* (12), and *fault tolerance* (6). P7 is a clear-cut example of *recoverability*, as when the system suffers faults it repairs them with adaptations that stop and start the faulty components. For *fault tolerance* P3 serves as an example. The authors have the system target *fault tolerance* by introducing a self-adaptation mechanism which mitigates faults through reconfiguration of the system *e.g.,* swapping one image processor for another.

The third most common QA is *safety* (12), of which we primarily find the subcategory *operational constraint* (10) and 1 instance of a *fail safe*. P37 mentions it targets safety, however it provides insufficient detail to determine a subcategory. The operational constraint in these primary studies is limiting the speed of the robot, with the aim of preventing collision. P10 implements a fail safe, as when the robot is under attack (in a cybersecurity sense) it both slows down as well as changes its navigation algorithm to use pre-defined safe data.

The fourth most common QA is *functional suitability* (9) which has two subcategories in *functional appropriateness* (7), *functional correctness* (1) and *functional completeness* (1). P8 is an example of *functional appropriateness* as its self-adaption revolves around course correction to make sure it successfully navigates. P27 is the only example of *functional correctness*, where a robot manipulator has to ensure that the slip detector it is using is accurately detecting slipping of the object in

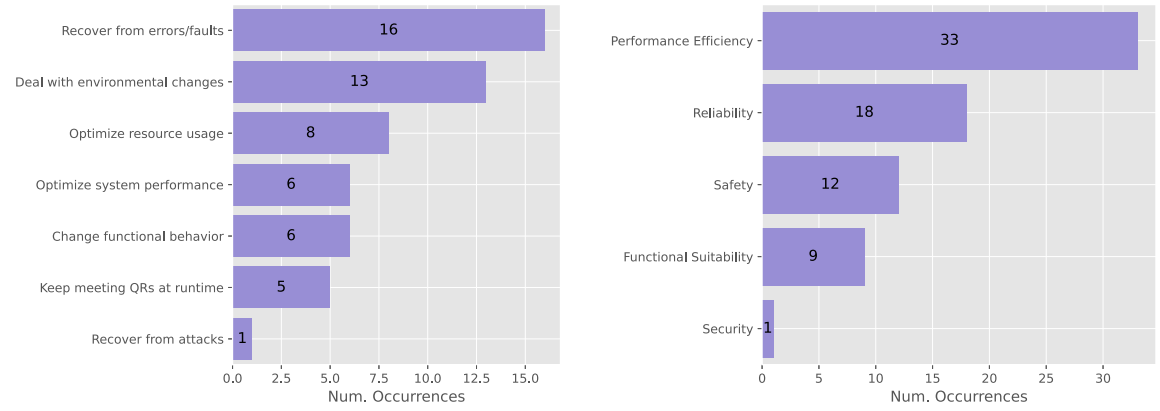


Fig. 3. RQ1.1 — Self-adaptive system characteristics: Adaptation goals and quality attributes.

its grasp. Lastly, P12 is the only example of *functional completeness* as the system adds new components at runtime to ensure it can make use of new tools to assemble different products.

Finally, P10 targets *security*, with *resistance* as its subcategory. In P10 the self-adaptation mechanism allows the robot to resist cybersecurity attacks, these attacks interfere with the robot's sensors, increasing the risk of unsafe behavior.

4.3. Managing system independence

The third characteristic we consider of each self-adaptive system is the degree of separation between the managing and managed system. This characteristic classifies the degree to which the managing system is independent from the given robotic system — and by extension its potential for re-use. Although one's assumption may be that these two systems are always separated, the separation between managing and managed systems is a *conceptual* one. We did not select the studies on the basis of their separated implementation as this would not have been straightforward to determine during the search and selection phase of this study. During the data extraction phase, we manually assessed the design of the self-adaptation approach present in each of the primary studies and consider to what extent the two (sub-) systems are independent.

The vast majority of studies (24) have managing systems that are partially independent in that they *require a representation* of the managed system to operate. Within Section 5.2 we elucidate the kinds of representations used in the primary studies. For example, P18 (and by extension P4, P17 and P30) requires the components of a given managed system to be represented as their own extension of the port-based object formalism. Another example can be found in P24 and the similar work P36 which model the managed system with an extension of UML MARTE and then uses this representation for its managing system.

We have identified 9 approaches which have managing systems that are readily *detachable* from the managed system. For example, P21 deals with restarting ROS components after they crashed. The managing system they propose includes mechanisms to recreate the state of the component prior to its crash, such as replaying communications to the component. Their proposed managing system is general to any ROS component and does not include assumptions as to the conditions of the managing system with which they choose to evaluate their approach. Another example is P2, which sees the communication of information between components in a system adapted. Once again, by design the managing system is general to the communication between any two ROS components which communicate.

Finally, only 4 of the 37 studies have what we consider *inseparable* managing and managed systems. For example, P10 uses self-adaptation as a mitigation strategy for cybersecurity breaches. In their approach, they design an adaptation logic that is entirely presumptive of the type of breach/attack faced by the managed system. This particular breach is also designed by the authors in the same study and therefore the managing system they propose has a dependency on this. Another example is in P13 where the managing system is fundamentally embedded within each component to be managed through reconfiguration.

Key insights from RQ1.1:
(1) Robotics software architecture-based self-adaptation is primarily done to recover from errors, optimize resource usage, and deal with changes in the environment.
(2) The main quality attributes targeted by robotics software architecture-based self-adaptation approaches are performance efficiency, reliability, safety, and functional suitability with a single study targeting security.
(3) The managing systems of architecture-based self-adaptive systems in robotics are mostly flexible, in that they only require a representation of the robotics system, or are completely detachable from said system, allowing for their application to different robotics systems.

5. Key characteristics of the managing system (RQ1.2)

This section reports on our application of the classification framework much in the same way as the previous. In this section specifically we report the characteristic which describe the managing system. This is a part of a self-adaptive system which is responsible for managing the adaptation concerns of the robotic system, traditionally realized through a feedback loop. We first report the mechanism responsible for adaptations in each primary study according to the dimensions by Andersson et al. (2009). Mechanism is defined by Andersson et al. (2009) as "what is the reaction of the system towards change". We report on the extraction of different details about that reaction and how it is facilitated. We consider this reaction to be the sole responsibility of the managing system, We report among other on details such as the algorithm backing its management, how the reaction is triggered and at what granularity is affects the robotic system in managing it. After the mechanism, we consider each phase of the feedback loop which

the reaction consists of, monitoring the robot, analyzing the monitored data to determine a need for adaptation, planning potentially necessary adaptations, and executing those.

5.1. Adaptation mechanism

In this subsection we report on various aspects of the adaptation mechanism used in each primary study. First, the method used by the mechanism is considered, which is our own addition to the classification framework and considers the adaptation logic backing each mechanism. We then use the modeling dimensions by Andersson et al. (2009) to report on the type, organization, scope, duration, timeliness, and triggers of each mechanism.

Method of mechanism. Fig. 4(a) shows the different mechanisms we identified regarding the manner in which RSASSs make self-adaptation-related decisions at runtime. We note that although this information could be partially derived from the synthesis of the analysis and plan phases of MAPE-K, we decided to extract this as a separate dimension; in this way, we obtain a more comprehensive and fine-grained understanding of the different solutions authors devised to perform runtime decision making. The majority of primary studies either use a *constraint solving/model checking* tool (*e.g.,* MiniZinc,[7] PRISM,[8] clingo[9]) (12) or a *search procedure* (9). For example, the authors of P7 use answer set programming, which has a declared set of constraints, whereas in P1 and P2 a graph search and tree search are performed respectively instead. Additionally, we identify 7 instances where a *design-time rules* are used , these can come in the form of if/else statements or for example the guards on transition in a state machine. For example in P33 there are transition in a task tree guarded by rules. There are also 5 cases of *ontological reasoning* and 3 uses of *AI Planner*s. In P10 and P21 *application-specific logic* is used. These are algorithms which are unique to the domain self-adaptation is being applied to, such as in P10 where the algorithm uses measures to counteract cyber attacks. In another 2 of the primary studies (P13 and P37) a *graph transformation* mechanism is used. Here the software architecture and potential configuration alternatives are all represented as graphs, the mechanism the consists of operations performed on these representations *e.g.,* finding the difference between two graphs. Finally, a single primary study uses *numerical optimization*: in P34, a linear programming solver is employed to solve a possibilistic linear programming problem for runtime decision making.

Type of mechanism.* The adaptation mechanisms are classified to be either *parametric* or *structural*, in accordance with the modeling dimensions by Andersson et al. (2009). More granularity as to how these structural or parametric mechanisms are executed is provided in Section 5.2 For example, P2 switches the 'modes' of individual components, a parameter, and also allows for the change of components by changing which dynamic groups they belong to. Overall, from the studies it results that *structural* adaptations are more common (32) than *parametric* (21). Of the primary studies, 16 make use of both mechanisms, while 16 only use a *structural* mechanism, and 5 only a *parametric* mechanism. For example, P18 exclusively has a *structural* adaptation mechanism, as it replaces faulty components in one robot with new ones by receiving the new software component over a network from other robots. An example of an exclusively *parametric* adaptation mechanism can be found in P8, where the look-ahead distance and velocity of the robot are adjusted at runtime.

Organization of mechanism. The vast majority (34) of primary studies have a *centralized* organization of the managing system. Of the 3 remaining *decentralized* mechanisms, P6 and P20 cover the ICE (Information processing self-Configuration and Exchange) approach, which targets a managed system of a heterogeneous team of robots. Therefore, by virtue of the system being multi-agent, and those agents being heterogeneous, the adaptation mechanism is distributed among the agents. However, this is not causative, as P3 sees a heterogeneous team of robots with a *centralized* adaptation manager. P25 is an example of a *decentralized* managing system featuring a single agent. Instead, there are distinct distributed managing systems for different facets of the robot to adapt; namely, both the grasp controllers and slip detectors in use by a robot hand are adapted separately, yet within the confines of one self-adaptive system and mission. Further it is clear from the design that there is no interaction between the managing systems of both adaptations (in their case the controller selector and detection selector).

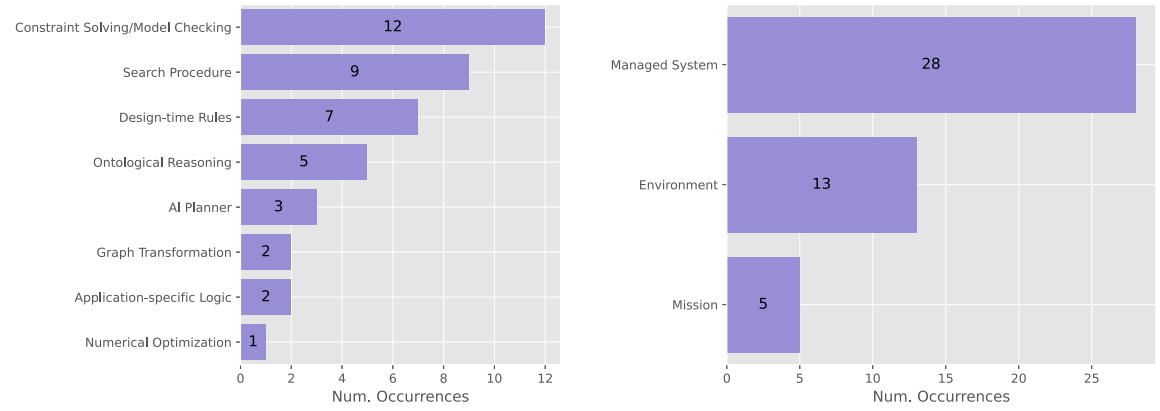
Scope of mechanism.* For the scope of the adaptation mechanism we consider to what extent adaptations affect the managed system. According to our classification framework, the scope of the adaptation mechanism can be either *local* or *global*. We identify 32 instances of local adaptation mechanisms within the primary studies and only 11 instances of *global* adaptation mechanisms. *Local* and *global* adaptation mechanisms are also applied in combination in 8 of the primary studies. For example, in P31, a configuration can adjust the overall power consumption of the robotic system (*global*), as well as the speed of each individual motor (*local*). Another example is P37, where replacing a failing component can affect the system's entire architecture, or only that singular component depending on the nature of the error and the role of the replaced component.

Duration of mechanism. The vast majority of studies have either *short* (26) or *very short* (4) adaptations. Studies classified as *short* are those that make clear no significant time overhead is induced by the adaptation taking place. Studies which have *very short* are exceedingly so, to the point that it is a design driver and clear benefit of their proposed approach. For example, P23 has as a part of their motivation that the adaptation is a very quick reaction in contrast to other approaches which use planning via a model-based approach. We are not able to determine the duration of adaptations from P27, P32, P33 as in these studies it is not clearly indicated by the authors. Only P21 sees a *medium* duration for its adaptation, as it deals with the crash recovery of ROS components. The approach of P21 seeks to reinstate the crashed node as closely as possible to approximate the situation prior to its crash. To accomplish this, it is necessary to 'replay' the experience of the component prior to its crashing, which can take a significant length of time as also elucidated in their evaluation.

Timeliness of mechanism. We consider for each primary study whether the mechanism and the adaptations enacted by it are *best-effort*, *dependent*, or *guaranteed* in their timeliness. For almost every primary study where this could be determined (31 out of 32), the timeliness is *best-effort*. This indicates that none of these studies made explicitly clear that their adaptation mechanism had a guaranteed time period, neither one which was guaranteed depending on the particular adaptation effected. The singular exception is P13. In P13 the authors encode each adaptation with specific time limits, a 'duration to success' and 'to failure', which can be used to determine and thereby guarantee the maximum time an adaptation takes.

Trigger of mechanism. For what triggers the adaptation mechanism of the managing system, the possibilities were either *event-triggered*, or *time-triggered*. For 36 primary studies, we find that the adaptation is *event-triggered*. These events are found to include among others, the failure of components (*e.g.,* P21, P22, P34), progress in the mission of the robot (*e.g.,* P32, P33), and changes in resource availability (P1). These results can be explained by the fact that most adaptation

[7] https://www.minizinc.org/.
[8] https://www.prismmodelchecker.org/.
[9] https://potassco.org/clingo/.



(a) Mechanism Method **(b)** MAPE-K Monitor

Fig. 4. RQ1.2 — Characteristics of managing system: Adaptation mechanism approaches and monitored aspects.

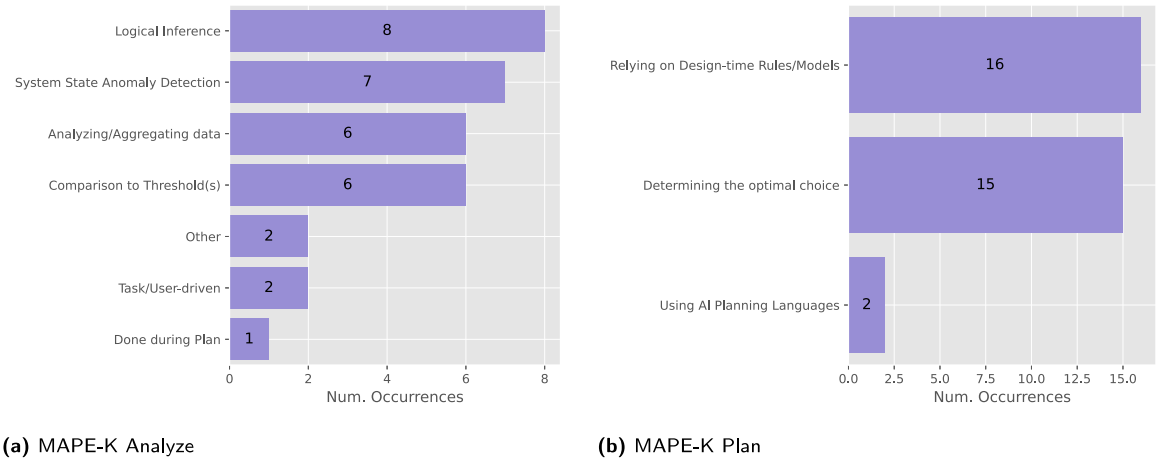
strategies aim at recovering from errors or faults (Fig. 3(a)), *i.e.,* from external or internal events of the system. The only remaining study, P28, did not explicitly make clear what acts as a trigger for adaptation in their scenario.

5.2. MAPE-K loop

In this subsection we describe the vertical synthesis of the data extracted from each primary study regarding the four phases of the MAPE-K self-adaptation loop (Kephart and Chess, 2003) and their shared knowledge.

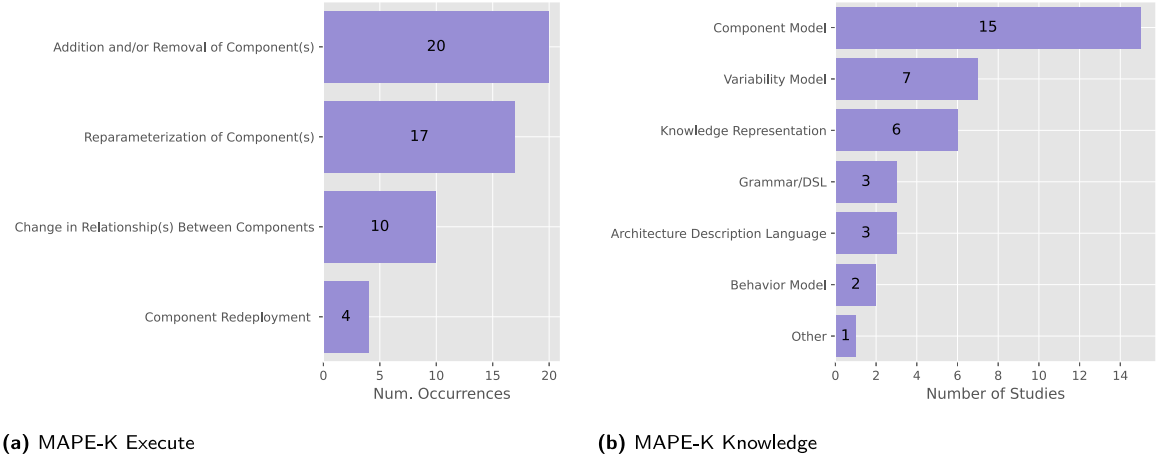
MAPE-K: Monitor.* In MAPE-K, the managing system broadly 'monitors' to obtain information on which to base its adaptation decisions. We classify the sources of this information among the primary studies as one of the: *environment*, *managed system* or *mission*, taking inspiration from the three categories of context by Turner (1998). Each of these sources potentially change at runtime and therefore monitoring is necessary to ascertain their state at a given time. As shown in Fig. 4(b), the most prevalent monitored entity is the *managed system* (28 occurrences), followed by monitoring the *environment* (13 occurrences) and monitoring the *mission* (5 occurrences). As an example, the approach of P1 monitors only the *managed system*, specifically it monitors for changes in the status on a component level, as well as the current runtime architecture of the system (in their case, the deployment of different parts of the system). P30 also monitors the *managed system*, but rather how it is performing a functional task; when the robot is assigned the task and it is incapable of performing it, this change in the *managed system* triggers an adaptation. In P27, the physical state of the *managed system* is monitored. Particularly, the managing system seeks to determine whether the object grasped by the robot is slipping or not. The same study also monitors the *mission*, as it is dynamic in its mission it needs to determine which task it is currently performing to then decide what about the *managed system* to adapt.

MAPE-K: Analyze.* With the analysis of MAPE-K we seek to answer the following question about the system in each study: how is it decided whether to adapt? The possible answers to this question are adopted by those used in the study by Weyns et al. on self-adaptation in industry, where they also attempt to classify this facet (Weyns et al., 2023). The results of this classification are depicted in Fig. 5(a). While there is even spread among each analysis method, the most common is *logical inference* with 8 instances. The idea here is that some logic-based representation such as an ontology (P6, P20, P32, P34) is used to infer whether adaptation is necessary. However, *logical inference* by our consideration can also include other logic-based representations such as in P4 where they use *Alloy* (Jackson, 2011). The idea is that these managing systems monitor information (as described in the previous subsection) and then use this information along with a query to a model to determine whether an adaptation will take place. In 7 of the primary studies the analysis is based on *system state anomaly detection*. The studies use the information monitored continually and explicitly compare these to an expected model of the system. For example, in P10 the latency of sensor values is continuously compared, if these are fabricated by a cybersecurity attack over the internet then there will be a discrepancy between the usual latency and that of the fabricated values. Further, there are 6 instances of thresholds being used to decide whether to adapt. This entails the comparison of monitored values to pre-defined values. For example, in P9 they have rules about which room the robot is in, its current battery level being higher than a certain percentage, its current velocity being a certain level among others. There are also 6 instances of using data analysis/aggregation techniques to determine whether to adapt. For example, in P22 the trend in data from two redundant sources is compared, when these deviate from one another an adaptation is triggered to remedy the faulty source. P14 uses both *comparison to thresholds* and *analyzing/aggregating data*. The authors employ discrete time Markov chains to analyze the reliability of the current software architecture of the system, and compare the resulting reliability to a specified threshold to determine whether a more reliable architecture is warranted. In the approaches of 2 of the primary studies the analysis is *task/user-driven*. As an example, in P33 there are two distinct hierarchical layers of deliberation in the system specified through domain-specific languages: the SmartTCL (Smart Task Coordination Language) and VML (Variability Modeling Language). SmartTCL determines if the current task requires an adaptation to be accomplished. Should an adaptation be necessary, the VML layer reconciles these changes with the QoS requirements placed on the system. Ultimately, it is a user who at some point (either at design- or run-time) determines the tasks the robot performs. Therefore, we make no distinction in our classification whether tasks are actively requested (*e.g.,* by voice command or keyboard input) or pre-programmed as a part of a mission for approaches belonging to *task/user-driven* class. Lastly, in 1 study (P8), analysis is done in tandem with planning in an atomic inseparable fashion. Therefore, we describe it further when considering the planning mechanisms.



(a) MAPE-K Analyze **(b)** MAPE-K Plan

Fig. 5. RQ1.2 — Characteristics of managing system: Analysis methods and planning policies.



(a) MAPE-K Execute **(b)** MAPE-K Knowledge

Fig. 6. RQ1.2 — Characteristics of managing system: Enacted changes and knowledge representation.

MAPE-K: Plan.* For the planning phase of MAPE-K we answer the following question for each approach: how is it decided which adaptation to enact? Our classification framework offers three possible categories for this aspect of self-adaptation: *relying on design-time rules/models*, *determining an optimal choice*, and *using AI Planning Languages* (Fig. 5(b)). Particularly, P7 and P22 make use of the Planning Domain Definition Language (PDDL) (Haslum et al., 2019) to choose their adaptations. Of the other two categories, there are 16 instances across the studies of planning through *determining an optimal choice* and 15 instances of *relying on design-time rules/models*. We can also provide another layer of detail when it comes to those studies using optimization. There are 4 studies which use a model checker optimize; for example, in P4 and P31 the authors use the PRISM model checker, a probabilistic model checker, which selects adaptations as actions that have a high probability of leading to high reward states. There are also studies doing optimization through search algorithms, such as in P7 and P22 where they use graph search, or P18 which does a combinatorial search, among others. For the studies which rely on design-time rules/models we can exemplify P17 which has rules in the forms of guards on transitions (adaptations) between the states in a state machine representing the modes of the system. Additionally, P16 formulates the adaptation problem as a Satisfiability Modulo Theories (SMT) problem at design time which is then solved to determine new architectural configurations.

MAPE-K: Execute.* Fig. 6(a) presents the results on how adaptation approaches enact changes. By far, the two most common executions are *component addition and/or removal* with 20 instances and *component reparameterization* with 17 instances. An example of addition and/or removal can be found in P27 as the component responsible for slip detection of an object in the robot's hand is removed and replaced by another. As for reparameterization in P28 adaptation actions determine parameters such as speed, video quality of its camera and which communication band it uses. There are also 10 instances of *changing the relationships between components*. By this, we mean that while the components individually remain static, their connection to other components changes dynamically through adaptation. For example, in P29, streams of information between components are re-arranged to handle changes in the managed system as different camera feeds are available. Which specific component receives the camera feed used for tracking an object changes then depends on the state of the robots. Lastly, there are 4 examples of component redeployment, where adaptation changes which components are running on which machine. Specifically in P1 processing tasks for the robots are distributed across different hosts to ensure requirements of being real-time and load balancing are met. In P12 instead assembly skills are transferred as components running on one robot to start running on another during operation. As a final example, in P14 the authors consider predefined deployment patterns (all components on one process, two components with two separate processes etc.) which they switch between at runtime.

MAPE-K: Knowledge. Lastly for MAPE-K we classify the knowledge, which we consider to be the representation of the managed system's architecture used to be able to modify it through adaptation. Within the primary studies we have found a fundamental distinction can be made between approaches which derive their architectural model by virtue of implementation (*e.g.,* ROS), and those which create an architectural model as a layer on top of an implementation. Even more interestingly, these two classes of approaches are not mutually exclusive, meaning that approaches which are already privy to an architectural model by virtue of implementation may still choose to create another layer of abstraction (*e.g.,* ROS plus a semantic layer). It holds for all of the cases where no extra layer of abstraction was used, that we consider them to be using a *component model* which is the case for 15 (and thereby the majority) of the studies (Fig. 6(b)). Examples of component models encountered include: ROS (P7), Extended Port-Based Objects (P3), and DEECo (P17). Fewer primary studies (7) use what we consider a variability model. These are constructs such as feature models (P9, P35) or extensions of UML (P24, P36) which formalize the changeable aspects of a system. Then, 6 studies use knowledge representation (such as ontologies); these ontologies contain specific rules and representations of the architecture, such as in P26 where ontological instances represent groups of components (to adapt between) as a layer on top of ROS' component model. The approaches of 3 of the primary studies use a grammar or domain-specific language to represent the managed system. For example, some approaches create a DSL in XText (P19, P33) which describes functionalities of the robot which are then translated into implications on the component level when it comes to adapting the system. Another 3 primary studies use architecture description languages (ADLs). These ADLs are a means to conceptually model a system's software architecture to then reason over it. For example, in P14 the authors extend xADL to represent reliability properties at the component level, to be used for further analyses regarding the overall reliability of a system's current architectural configuration. Lastly, 2 systems (P5, P10) use a behavior model, such as a state machine to represent the managed system. We also classify 1 study as being in an 'Other' category, as it is unique in their representations. Specifically, P31 trains a machine learning model during design time on the configuration space of their robot to reduce it and determine the effect of varying configurations on quality attributes.

Key insights from RQ1.2:
(1) Managing Systems of RSASS primarily have adaptation mechanisms with a basis in constraint solving, model checking, and search procedures.
(2) Managing Systems of RSASS primarily make use of both parametric and structural adaptations. They accomplish the former by re-parameterizing at the component-level and the latter by removing and adding components.
(3) The adaptation mechanisms of RSASS are primarily centralized, applying to individual components, and are enacted with a short duration. They are mostly triggered by events, and do not provide guarantees of their efficacy.
(4) When adapting, RSASS approaches tend towards monitoring the robotic system itself rather than the environment or its tasks. They primarily analyze the monitored information through logical inference or comparison to an expected state of the system. A decision on which adaptation to enact is made as the solution to an optimization problem, or by relying on models of adaptation decisions specified at design-time.

6. Key characteristics of the managed system (RQ1.3)

This section reports on the key characteristics of the managed system *i.e.,* the robotic system which is being adapted by the managing system. This entails the characteristics of both the mission the robot performs, and the robot itself (hardware and software). In descending order of granularity, we first characterize the missions assigned to each system (6.1), determining the aim and whether it evolves over time. Then, we characterize the *change i.e.,* the reason for adaptation (6.2), and the effect of adaptation (6.3) in accordance with the dimensions by Andersson et al. (2009). Lastly, and most specifically, we consider the actual robots used (6.4). We specifically consider the software platform used to implement the approaches and the specific hardware/model of robot used.

6.1. Robotic missions

In this subsection we consider the mission assigned to each managed system *i.e.,* each robot. We first consider the aim of said mission, and then whether the mission exhibits dynamicity in its tasks.

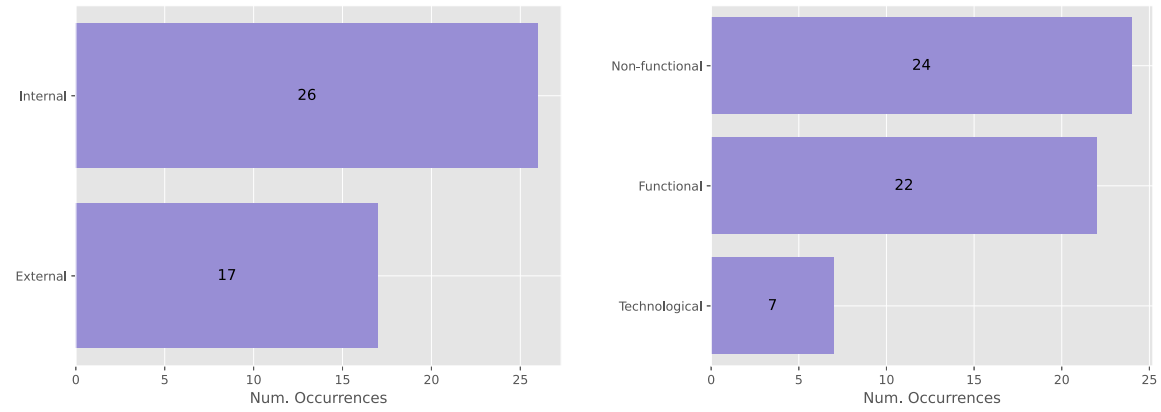
Aim of mission. The most common, and most broad mission we find is *navigation* with 18 studies. More complex missions consider navigation as a task or a side-concern to their main mission. Therefore, when it comes to missions, we only consider what the authors state as the primary task/mission of the robot. These 18 studies do so, for example, P4 has a robot patrolling and therefore navigating corridors, the same is true for P31; P26 is also concerned with navigation where corridors are of uncertain breadth. The second most common is *emergency response* with 7 studies. This includes missions such as search and rescue operations (P3, P7 P28, P30) and coordinating a team of robots during a disaster to deliver supplies to victims (P6, P20). A further 3 approaches cover a mission of acting as a *service* robot *e.g.,* butlering as in P33. Additionally, 2 approaches P2 and P18 are applied to a mission where a robot needs to keep track of the location of an object *i.e., object tracking*. Lastly, P1 and P12 both have a mission of the robot being used for *industrial assembly*, while P27 is the only study to consider the mobile manipulation of objects. For 4 of the studies, no specific mission was prescribed, with the study only describing the approach generally. Overall, beside the common navigation mission, there seems to be significant variety in the kinds of missions self-adaptation is applied to.

Evolution of mission. We have adjusted the modeling dimension related to goals from Andersson et al. (2009) to consider the evolution of the mission specifically over time. By our definition, a mission is *dynamic* if it sees the robot engage in changing tasks over the course of it, whether the timing of those changes is known or not. Of the 37 primary studies, 22 are found to have entirely *static* missions, thereby consisting of one primary task in isolation. For example, in P28 only the "maneuvering" task of the robot is considered. Several studies (6) have *dynamic* missions, and a further 9 do not provide enough detail about performing their mission to be able to determine the evolution. As an example, in P27 there is a *dynamic* mission as the robot first has to manipulate objects, before then carrying them to another location to place them. P2 sees two robots work together, with their respective tasks such as tracking and detecting an object changing dependent on the uncertainties present and therefore also *dynamic*.

6.2. Change - The cause for adaptation.

In this subsection we consider the various dimensions describing the change which leads to adaptation in the primary studies. In particular, the source of this change, its type, whether it is anticipated, and its frequency.

Source of change. The change in the managed system leading to a need for adaptation is classified as originating either from a(n) *internal* (emerging from within managed system), *external* (emerging from



(a) Source of Change **(b)** Type of Change

Fig. 7. RQ1.3 — Characteristics of managed system: Source and type of change.

outside), or *mixed* (both) source. The majority of studies consider exclusively *internal* change (18) as the impetus for adaptation (Fig. 7(a)). This stems clearly from the overall trend of adaptations driven by faults in the system. Further, 9 studies adapt due to exclusively external factors. These stem from environmental changes such as illumination as seen in P24 and P25. Lastly, 8 studies see both internal and external sources of change. As one would expect, these include combinations of the examples given for exclusive sources. They are systems with sufficient complexity to handle both sources with one approach such as P31 which deals with the robot's internal battery level but also the external uncertainty posed by the terrain it navigates.

Type of change.* The possible types of change considered are *functional* (change affecting what the system should do), *non-functional* (change affecting how the system operates), or technological (relating strictly only to the technology supporting operation *e.g.,* operating system and not the software written for self-adaptation). The most frequent type of change is *non-functional* (24 occurrences), whereas 22 are *functional*, and 7 are *technological* (Fig. 7(b)). There are 4 studies (P3, P11, P18 P30) which deal with all three types of change. However, these studies all belong to the same approach: ReFrESH. The reason is primarily that ReFrESH supports component failures which can also be due to hardware failures, such as a sensor failing, this falls under what we consider a *technological* change. Besides that, they consider the non-functional requirements being violated such as performance constraints, as well as the tasks the robots need to complete (*functional*) as changes. There are 2 studies P5 and P10 which are exclusively *technological* in their change. For P5 this stems from the changes happening in a piece of middleware supporting robotics operations. With P10 the change stems from cyber attacks, these attacks directly affect the operation of both the software and hardware (the sensors). P32 is an example of a purely *functional* change, as all of its changes stem from the task given to the robot changing ergo the functionality it has to perform changes.

Anticipation of change. We consider whether the change leading to adaptation to either be foreseeable, unforeseen, or completely foreseen. As can be seen in Fig. 8(a), 23 of the studies have foreseeable adaptation. This classification comes about when authors do not make clear whether their approach accounts for unforeseen or foreseen changes. Therefore, by virtue of the existence of a sound approach, the change must be foreseeable. Due to this, a smaller subset of approaches is those 7 approaches which handle unforeseen change. These are approaches robust to a potential change whose occurrence details are unknown. For example, P2 and similar approaches in P19 and P23 maintain a set of logical rules about the system. Independent of any specific change, these rules are asserted and when in violation remedied through the designed adaptations which solve the systems erroneous state. Lastly, in P21, P14, and P32 the change is foreseen. In P21 this is in the form of anticipating the crash of a given ROS component. In P14 proactive adaptations are performed in response to a predicted insufficient reliability of the current architecture, meaning the change is always anticipated. In P32, the changes are triggered by a change in the task of the robot, which with a planned mission is always foreseen.

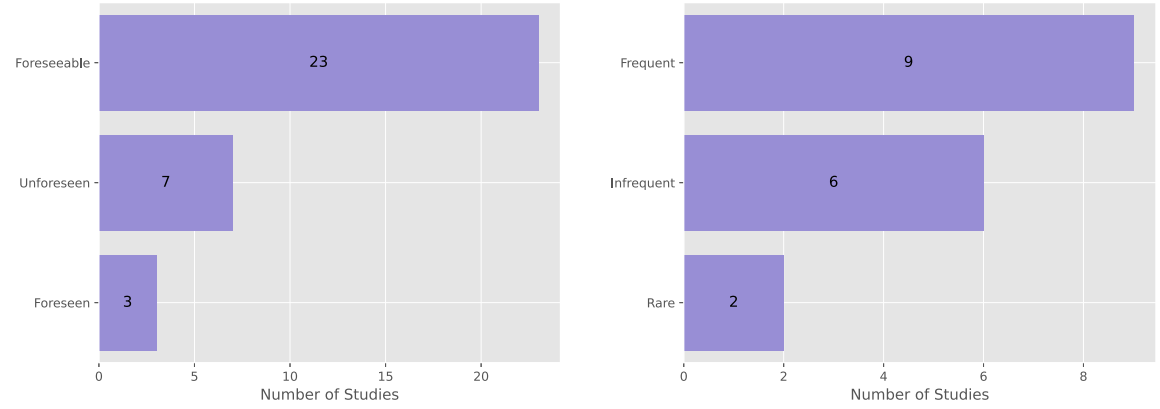
Frequency of change. In the cases where change is not unforeseen, (26 of the primary studies) we consider the frequency of the foreseeable or foreseen change, if specified. Most 20 studies do not clearly indicate a frequency of change. From the studies which do indicate a frequency, we depict in Fig. 8(b) that there are 9 studies with frequent change. For example, in P8 the change has to do with localization uncertainty; a robot relies on its current position frequently while navigating, increasing the frequency of potential changes playing a role. There are another 6 studies with infrequent change. For example, in P32, the change is driven by a user requesting a task of the robot, leading to a need to adapt the robot to allow performing said task. As these tasks are performed sequentially and take some time, this happens infrequently overall. In P10 and P21 there is a rare frequency of change. The former deals with cyber attacks and the latter with components crashing, both of which are rare occurrences yet with severe consequences.

6.3. Effects of adaptations

In this subsection we consider the effect adaptation has on the managed system of each primary study based on the modeling dimensions by Andersson et al. (2009). Particularly, we describe the criticality, predictability, overhead and resilience of the effect.

Criticality of effect. A majority of 24 primary studies see an adaptation as mission-critical. In other words, not adapting the system may jeopardize the mission assigned to the robot in some way. For the remaining 13 primary studies, 10 of them consider a lack of adaptation more severe, with it being safety-critical, while the other 3 did not provide sufficient data to determine the criticality. P28 can serve as an example of a safety-critical adaptation. In the study, adaptation ensures that the localization of the robot is accurate, therefore if this accuracy is not maintained through adaptation the robot may collide while navigating, jeopardizing its safety and that of others. It is notable that in none of the primary studies the criticality of adaptation was considered harmless.

Predictability of effect.* Here we extract whether adaptations are deterministic or non-deterministic in their effect on the managed system. Most adaptations among the studies have *non-deterministic* effects (22). For this data point, we adopted a stance of trying to prove the contrary. In other words, studies are presumed *non-deterministic* unless they provide clear evidence to the contrary. There are fewer studies



(a) Anticipation of Change **(b)** Frequency of Change

Fig. 8. RQ1.3 — Characteristics of managed system: Anticipation and frequency of change.

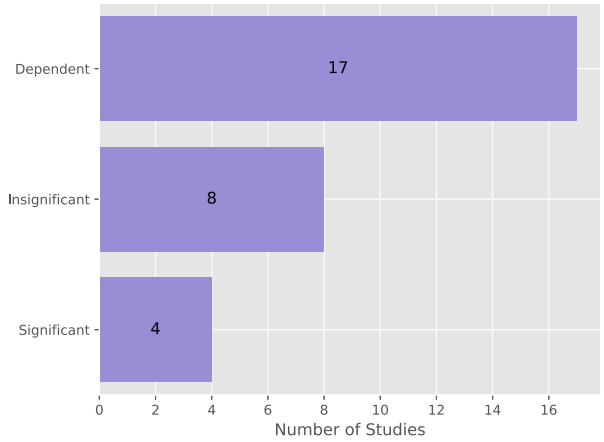


Fig. 9. RQ1.3 — Characteristics of managed system: Overhead of adaptation effect.

which provide evidence of *deterministic* effects (10). Study P10 has both deterministic and non-deterministic adaptation effects. For example, in the study they adapt by lowering the speed of the robot, which is deterministic. However, they also adapt to block attackers on the same network on the robot, which due to the uncertainty surrounding the nature of the attackers, cannot always block every attacker. Therefore, the adaptation is non-deterministic in its effect of securing the robot.

Overhead of effect. We consider the overhead of adaptations to be either *dependent* on external factors, *significant*, *insignificant*, or inducing *failure*. As depicted in Fig. 9, we find that the majority of the studies have *dependent* overhead 17. For example, in P4 and P31, there is a phase of planning in determining configurations for the robot; depending on the current context and the configuration space this can be an insignificant amount of time and resources or be so complex that it leads to failure of the system. Further, there are 8 studies with insignificant overhead. For example, in P26 there is insignificant overhead, as the adaptation is always performed from a fixed set of configurations which are predefined and chosen by an ontological reasoner. Lastly, another 4 studies have significant overhead. For example, in P7 the authors state that in one of the scenarios it takes 30 s to enact the adaptation of powering on the system from a 'faulty' state of all components being off. The remaining studies did not provide sufficient information to extract the overhead of their effects. Despite being a possible classification, none of the studies are found to induce *failure* through their overhead.

Resilience of effect. Unfortunately, we are unable to report on this parameter. Contrary to our assumption when designing the study, it proved difficult to extract this parameter from the primary studies due to the following reason. Typically each primary study describes a mechanism which is resilient to an exemplified set of changes. During data extraction, it became apparent that the primary studies do not provide evidence of their resilience beyond this set of changes. This lack of evidence made it impossible for us to report on the resilience of the mechanisms in a holistic way. There were also some exceptions to this finding, in particular P16 and P37 make clear that achieving resilience is the aim of their respective approaches. The details of which can be found in both our extracted data (Replication, 2024) and their exemplification throughout the rest of the results.

6.4. Robotic system

In this subsection we cover two specific details regarding the robotic systems which serve as managed systems across the primary studies. Particularly, the software platform supporting the system, and the model of robot being used for its physical realization.

Software platform. For every study the software platform used for implementing their approaches was extracted. In the case of 9 of the studies this was not clearly and/or explicitly indicated. For the remaining 26 we took note of any multiple occurrences, *i.e.,* the use of the same platform across approaches. The only consistent reuse seen was that of ROS, the Robotic Operating System. At the time of writing, there are two distinct versions of ROS, namely ROS 1 and ROS 2. While ROS 2 is the intended successor of ROS 1, both still see consistent usage and development support. The first official release of ROS 2 was in 2017, while ROS 1 predates the period that we consider for including primary studies. As we can be seen in 3, 16 of the primary studies were published after 2017. This means that only those 16 could feasibly make use of ROS 2. We find that only 2 studies, P19 and P34, use ROS 2 specifically. A further 16 make use of ROS 1, of those, 11 are from after 2017 and could have feasibly made use of ROS 2. The remaining 10 studies make use of what we categorize as 'Other' software platforms. These include both completely ad hoc implementations as well as usage of possibly more widely-used platforms which simply were not also used by any of the other primary studies. Careful consideration is taken here not to overstate the usage of software platforms which are clearly being used by the same authors for the same related approaches across different publications within our primary studies.

Types of robots. There is no discernible trend in the types and or models of robots used across the primary studies. Most of the studies (18) name a specific model in their work, while the remaining 12 are generic in

describing the robot. The most common specific models are iterations of the Turtlebot model of robot, with 5 instances, which is a set of affordable small mobile terrestrial robots. For example P21, specifically makes use of a simulated Turtlebot3 Waffle and a real Turtlebot2 platform and P35 also makes use of a Turtlebot2. Another set of studies P2 and P29 use the NAO robot, which is a humanoid robot. The remaining studies P11 and P18 make use of a HexManipulator robot, which is custom built by the authors and is an example of a Stewart platform (Dasgupta and Mruthyunjaya, 2000). From these results we can surmise that there is no consistency in the robot models used in architectural self-adaptation. Therefore, no conclusions can be drawn about particular robots lending themselves more easily to self-adaptation than others.

Key insights from RQ1.3:
(1) The main mission to which self-adaptation is applied is navigation; additionally the missions tend to be static as the tasks they involve do not evolve at runtime.
(2) The change in the managed system that is cause for adaptations is primarily non-functional and emerges internally. These changes are foreseeable and vary in their frequency.
(3) The effects of adaptation on the managed systems of RSASS are primarily non-deterministic and the overhead of these adaptations largely depend on each specific adaptation scenario.
(4) ROS is the most-frequently used software platform in the primary studies.

7. Evaluation strategies (RQ2)

In this section we cover the results pertaining to RQ2 "What are the *evaluation strategies* of approaches for architecture-based self-adaptation in robotics software?". Specifically, in accordance with the classification framework presented in Table 3. We first characterize the nature of the systems usage, determining whether systems are deployed in the real world, simulation, or some combination of both, and whether the context of that deployment is realistic or synthetic. Then, we characterize the evaluations performed in each primary study, determining by what metric they quantify the performance of their approach and should there be an evaluation if it is a simply a showcase or compares to a baseline. Lastly, we also report whether or not the primary studies provide a replication package for their evaluations.

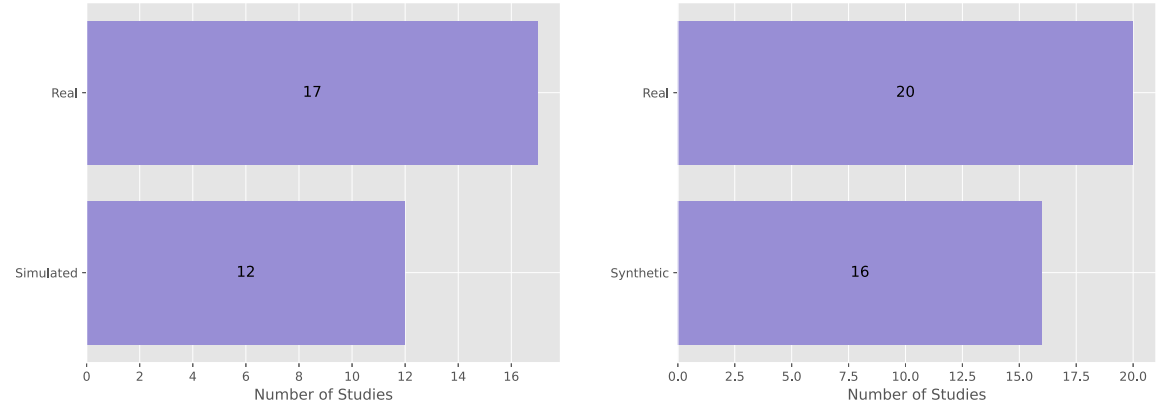
System deployment. We classify whether the system being evaluated is deployed in a *simulated* environment, in the *real* world, or using both in combination. We find that a majority of the studies (17) deploy robots in real life (Fig. 10(a)). A further 12 studies only consider robots in a simulated environment, or do not state whether the robot is deployed in the real world. For example, in P26 the robot is simulated in the well-known Gazebo simulator, which is also leveraged to provide input into their adaptation logic by way of obstacle density in the room. For the remaining 8 studies it was either ambiguous or there was no specific robot, but rather the concept of one meaning there was nothing to specifically simulate nor deploy in the real world. This also entails that there are no studies which made clear they use a combination of simulation and real-life deployments. Although likely every real-world deployment of a robot has in some part been simulated in its development, we mean here that its deployment during a mission would be in the real world while also making use of some simulation, *e.g.,* a physics simulator to predict the effect of actions in the real world. The fact that so many of the studies are done with a real-life deployment is a positive indication when it comes to the strength of evaluations, given especially the uncertainties which self-adaptive systems deal with are amplified due to the reality gap between simulation and real-life (Brooks, 1992).

System realism. With system realism we identify whether the operational context of each primary study is either *realistic* or *synthetic*. By *realistic* context we refer to robots deployed in the field whereas by *synthetic* we refer to lab deployments. With this parameter, we aim to indirectly indicate the technology readiness level of the systems as well as the extent to which researchers substantiate/evidence their claims. As seen in Fig. 10(b), the studies are divided almost evenly between the two with a slight edge (20) given to realistic contexts versus synthetic ones (16). For example, P1 is *realistic* as it revolves around an industrial robotics setup with complex and detailed architecture and a varied set of tasks to perform around automated assembly. In contrast, P31 is an example of a *synthetic* system as it features a simple mobile robotics scenario with the task of navigating through a room containing obstacles.

Evaluation metrics.* The evaluation metrics of each approach indicate what the authors of each study measure in their evaluations (*i.e.,* what is on the y-axis). During the data extraction phase, we identified the following types of evaluation metrics: *quality* of service, *mission performance*, *overhead (introduced)* by the approach, *domain-specific performance* measures pertinent to that system, and *resource consumption* (Fig. 11(a)). The most common metric by far is *mission performance* with 16 occurrences. For example, in P3 the authors measure the error in pixels of an object tracking algorithm which is disturbed by uncertainties during the mission which their self-adaptation approach then corrects. Further, there are 9 instances of considering the *overhead introduced* by the approach. For example, in P20 the authors measure the computation time in milliseconds of the system in different scenarios while using their approach and how overhead scales with complexity. There are 6 instances of considering the quality of the operation by the robot. For example, the approach in P28 is evaluated on the utility achieved by the system defined as a function combining five user preferences related to the qualities the system should uphold such as responsiveness. Another 5 instances use a *domain-specific performance* measure in their evaluation. For example, in P21 the approach for recovering crashed ROS components is evaluated partially with a binary value representing whether or not specific components recovered from a crash. This is a measure only applicable to P21, or other studies in that domain which specifically deal in crashing (ROS) components yet it is non-specific to the mission of the robot that houses the components. Lastly, 2 studies consider the resources consumed by the system, *e.g.,* power as in P9. This is used as an evaluation metric as the intention is for the adaptation to manage the consumption of a resource over time.

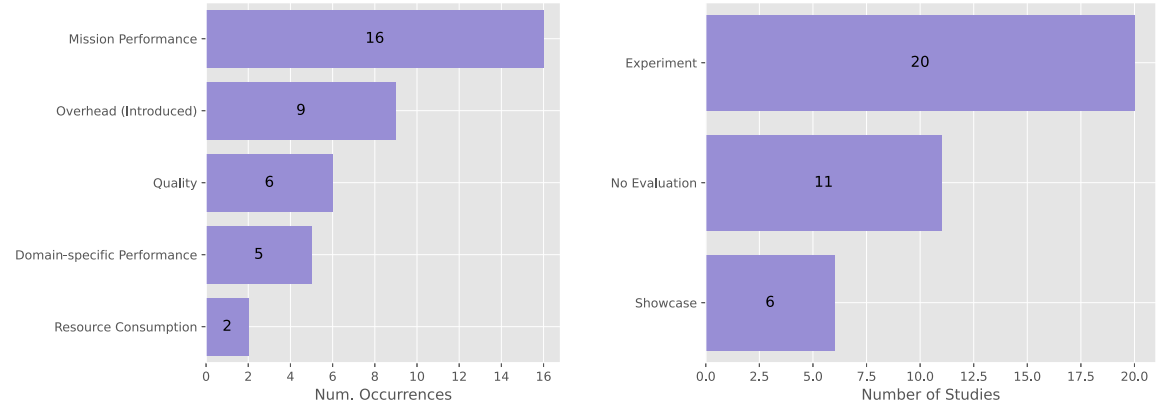
Evaluation depth. We follow the study by Gerostathopoulos et al. on the evaluation of self-adaptive systems (Gerostathopoulos et al., 2021) in considering the evaluation method used in each study. They themselves cite Wohlin et al. (2012), which distinguishes between showcases and experiments. The key difference is that a showcase concerns a single experiment configuration, while an experiment has a quantitative comparison between the results of multiple experiment configurations. In our study, the majority of studies (20) have a full-fledged *experiment* (Fig. 11(b)). For example, in P4 the authors use three different experiment configurations in a comparison: one where there are neither uncertainties nor adaptations, another with uncertainties but no adaptations, and lastly with both uncertainties and adaptations. About a fourth as many studies have only a *showcase* (6) rather than a full experiment. For example, in P22 the authors design one experiment configuration where they introduce an obstacle to a robot while it is mapping a room to cause an error and showcase their adaptation handling the error. The remaining studies have *no evaluation* (11) at all. This entails that roughly half of the primary studies do not fully evaluate their approaches, indicating a prevalent lack of soundness in their evaluation.

Replication package. We follow the methodology of the work (Cervera, 2018) on reproducibility of robotics research. In the work, each primary



(a) System Deployment **(b)** System Realism

Fig. 10. RQ2 — Evaluation strategies: Deployment and realism of system.



(a) Evaluation Metrics **(b)** Evaluation Depth

Fig. 11. RQ2 — Evaluation strategies: Evaluation metrics and depth.

study is searched for links to open-source repository hosting websites *bitbucket*, *github*, *gitlab*, *sourceforge*, and add *zenodo* to this. We find that the majority (26) of the primary studies fail to provide any kind of link to resources backing their proposed approach. The remaining 11 do provide one. This represents roughly one third of the studies, which is much higher when compared to the result of 4% found by Cervera. Notable is that Cervera's study only considered one venue ICRA which is also among our selected venues. The 10 studies which do provide a link do not indicate a clear bias towards any type of venue or specific venue.

Key insights from RQ2:
(1) RSASS are primarily deployed in real life and not only simulated, dealing with either realistic or synthetic applications.
(2) RSASS are primarily evaluated through the performance of their particular missions, and so through full-fledged empirical experiments without providing a replication package to further strengthen their results.

8. Cross-cutting results (horizontal synthesis)

In this section we elaborate on the main results of our horizontal synthesis (see Section 2.4.2). Specifically, we built a set of 39 pairs of parameters whose co-occurrences would lead to potentially-relevant results, then we built their corresponding contingency tables, and finally we collaboratively analyzed the contingency tables and synthesized the results we deemed relevant to be shared in this study. At the end of this procedure, we came up with 10 relevant pairs. We elaborate on each of those pairs in the remainder of this section; the other pairs have been discarded either because we could not observe any relevant pattern in their contingency tables or because the observed patterns were providing only marginal insights with respect to those of the vertical synthesis of the individual parameters. It is important to note that the suggestions and implications of our horizontal synthesis are based on *our interpretation* of the co-occurrences of pairs of parameters we observed in the extracted data; at the time of writing we do not have any objective evidence about whether the observed co-occurrences are causally linked or statistically confirmed, so we invite the reader to interpret our elaborations on those co-occurrences more as reflection points, rather than as objective evidence about specific aspects of architecture-based self-adaptation in robotics. For the interested reader, all contingency tables and an explanation of the rationale for building each of the 39 pairs are included in our study's replication package (Replication, 2024).

Mission by source of change. In line with our vertical synthesis, the majority of co-occurrences concern systems carrying out navigation missions (18 occurrences in total). Within those systems, we observe a certain balance among those reacting to external sources of change (12 co-occurrences) and internal sources of change (11 co-occurrences). A similar trend can be observed when considering other types of missions, such as those involving service robots, object tracking, mobile manipulation, and industrial assembly. An interesting exception to such balance concerns emergency response systems; among them, all but one of the studied approaches react exclusively to internal sources of change (6 co-occurrences). For example, the system studied in P30 involves the usage of heterogeneous miniature robots carrying out

search-and-rescue missions in an urban environment; however, despite the potential for managing external sources of changes (*e.g.,* a change in the physical environment), the proposed adaptation approach focuses primarily on internal sources of change, namely, self-diagnosis, integrating reusable hardware modules, and migrating software modules at runtime. One would expect uncertainties external to the robot, such as unstable terrain, changing temperatures, or visual obfuscation to be a more prevalent concern. To that end, it is fair to mention that the *internal* change in P3 (an example of emergency response) has to do with introducing a new component for 'dehazing' the camera feed. The reason this is not an external change is that the hazy conditions are a constant while the change is the inability of the robot to dehaze the camera feed by itself which suddenly becomes pertinent when it needs to do so. The singular exception is in P14 which has both internal and external sources of change. Although like the other co-occurrences it is still mainly concerned with internal errors, it also considers external contextual conditions when predicting the reliability of its own components.

Scope of mechanism by quality attribute. The co-occurrences of the scope of adaptation mechanism and the targeted QAs mainly follow the same trends identified in our vertical synthesis (the most common values of each parameter most commonly co-occur). There is an exception though: within our dataset, global adaptation is mostly done for keeping the performance of the system (9 occurrences) acceptable and for reliability and safety (to a lesser extent, 4 occurrences each).

Mechanism method by system realism. If we look at the data in isolation of the system realness, there is a somewhat even split between *real* and *synthetic* missions performed by the systems (20 to 16) among the primary studies. However, this split is not maintained for every one of the adaptation mechanisms found among the approaches. If we consider each mechanism in descending order of their prevalence from the vertical synthesis, we see that *search procedure* mechanism is twice as prevalent (6 to 3) with *real* missions. What we consider *real* missions carry a relatively higher complexity to their *synthetic* counterparts and provide more evidence for their claims. It is plausible then that a more complex mission carries with it a search space complex enough to require search procedures to explore rather than being solvable with ad-hoc solutions. *Constraint solving* is evenly split among the mission realness, indicating that, in the analyzed primary studies, being able to devise constraints does not seem to be influenced by the mission's complexity. It appears further that *ontological reasoning* is only applied to *real* missions with 5 co-occurrences. For the remaining mechanisms, there seems to be no deviation from the trends made clear in the vertical analyses of these two data points.

Overhead of effects by managing system independence. As one might expect from the vertical synthesis, the highest co-occurrence of 9 involves *requires representation* for the independence and *dependent* for the overhead. Both of these are most common in the vertical synthesis of the respective parameters. Similarly, for the second highest co-occurrence of 6 between *detachable* for the independence *dependent* for the overhead and both these values are quite common in the vertical synthesis. Contrary to our hypothesis that detached managing systems lead to higher overhead, we find that P1 and P21 are *detachable* and also *insignificant* in their overhead. Both have in common that the roles of their adaptations are secondary to the primary functionality of the system. In P1 the adaptation deals with the deployment of components, while in P21 it restarts components and recreates their previous state after they fail. In both cases the nature of the components themselves or their purposes holds no bearing. This entails that they are *detachable* as they are non-specific to the service of the system, as well as that their overhead of applying them is *insignificant* since efficient methods are used to deal with component re-deployment or re-starting. Even though they have a secondary nature, the implications of not doing application (better captured by our other parameter 'criticality of effect') still eventually affects service, as not managing resources through redeployment in P1 or not effectively reinstating components in P21 would clearly lead to a degradation in service.

Source of change by MAPE-K monitor. The vast majority of monitored aspects are the managed system as determined in the vertical synthesis. Less drastically, the source of change is more commonly (26 to 17) internal rather than external. There are 22 co-occurrences of *internal* and *managed system* and a comparable 14 with *external* sources of change. The former is to be expected, as an internal change is one which is derived from within the system, in this case the managed system. Looking closer at the data, it is then of interest which systems are exclusively monitoring the managed system yet also have an external source of change, something which thus far seems to be illogical. We do not consider the other instances as both pieces of data have multiplicity since multiple sources and multiple aspects can be monitored in a singular approach and we do not record the matching between these two directly. In other words, we do not have the data to directly link one source to one monitoring mechanism, but we can recreate it by considering studies where only one of each, source and monitored aspect, are recorded. There are two studies P1 and P10 which have exclusively external sources of change, yet they monitor the managed system itself. Both of these assume that the external change has some direct implications for the internal change of the system. In P1 the implication of the change is on resource availability, and in P10 this is due to security breaches affecting the operation of the robot. Crucially, the *original* source is still external, there is a knock-on change in the managed system which is monitored for, likely as this is more practical. It is logical that the system would only concern itself with those external changes which actually end up affecting the managed system, and not be distracted by other changes. For the remaining data, as is expected, a majority of external sources of changes monitor the environment. There are 7 co-occurrences of internal sources and environment being monitored, but none of these exclusively co-occur as was the case for P1 and P10. Lastly, the mission is mostly monitored when there is an internal source of change (4 occurrences) rather than an *external* source (2 occurrences).

Evaluation metric by system deployment. It seems that for all the evaluations which have some metric specific to their domain, these are real-life systems. It is likely that as these approaches are more matured or applied, and therefore develop metrics which hold more significant meaning for their particular application. For example, P3 considers an industrial robot with domain-specific evaluation metrics in terms of time/scheduling analysis of the system. This is a specialized subset of overhead, which becomes pertinent to the real system under study. In a simulated deployment, the timing could instead be orchestrated perfectly if the authors desire this to be the case. The occurrences of performance of a mission are not specific to any kind of deployment, being spread almost equally among the two categories. This confirms that, despite their deployment conditions, every robot still tends to have a mission. Resource consumption only co-occurs, although only 2 times in total, with a real deployment. Although it is plausible to measure in both deployment types, measuring resource consumption accurately on a simulated deployment can bring extra challenges as real hardware to measure is not being engaged.

Predictability of effects by managing system independence. We cross-checked the predictability of adaptation effects and the independence of the managing system in order to understand if it is feasible, according to the state of the art, to have deterministic self-adaptation at the architectural level even when the managing system is completely independent from the robotic system. We believe this to be the case, as a more separated managing system has less assumptions about the nature of its managed system to rely on to provide predictable effects. We identify 5 primary studies — P1, P6, P17, P19, and P21 with *detachable* managing systems yet *deterministic* adaptation effects. As a representative example, in the approach presented in P1, runtime adaptation maps tasks to computational components according to an algorithm that (i) is agnostic to the logic of the tasks themselves, hence detachable, and (ii) respects real-time requirements of tasks,

making the effects of adaptation deterministic hence predictable. This suggests it is plausible to have a detachable managing system with predictability, as long as the adaptation happens at a level equally removed from the specifics of the system as the managing system is itself. In other words, it is more 'meta'/second-order. This is also supported by the other four studies, such as in P6 which deals with only the relationships between components not the specifics of them, and P21 which deals with ROS components crashing, which can happen to any component.

Quality attributes by quality attributes. The two QAs which most commonly co-occur are *reliability* and *performance efficiency* (13 co-occurrences). This is somewhat unsurprising as they are also the two most common QAs derived from the vertical synthesis. We cannot concretely state from the data whether these two are part of a trade-off or targeted together, but it seems likely that the two are congruent or independent. For example, in P2 only *reliability* and *performance efficiency* are targeted. In that approach *reliability* is prioritized over *performance efficiency* but the two QAs are not directly part of a trade-off. The second most common co-occurrence is between *safety* and *performance efficiency* (6). These are more likely to be traded off, as a common way of being more efficient in moving more quickly, is typically less safe as the consequence and likelihood of collision increases. P26 is an example of a study which targets both and it indeed has a trade-off between the two for its mission of a mobile robot navigating a narrow corridor. For the *functional suitability* QA there is no meaningful co-occurrence with others. It occurs 5 times with *performance efficiency* 4 times with *reliability*, 2 times with *safety* and 1 time with *security*. This follows the general trend of the data established in the vertical synthesis. As there is only 1 instance of *security* overall, we do not consider its co-occurrences.

System deployment by MAPE-K knowledge. The correlation deviates slightly from the results of the vertical synthesis of the individual parameters. While there is almost an even split between simulated and real deployments (12 to 17), a large majority (11 to 3) of systems using a component model as their representation are deployed in a *real* environment. component models such as DeeCo in P17 tend to be a form of knowledge which is a side-effect of implementation. Therefore, the authors chose not to devise an extra layer of abstraction over the implementation of the system to be used for adaptation of the architecture. This was a choice, as we also observe that some approaches using ROS, which itself provides a form of component model, still use an extra layer of abstraction, as was done for example with OWL ontologies in P26. For the remaining knowledge types, there are only sparse instances of them being used with real deployment. This, despite the fact that there are 7 instances of *variability models*, 6 instances of *knowledge representation* and 3 *grammar/DSLs* (although 1 one of the latter has no data for the system deployment). For the remaining data there is an even split between being deployed in real life or in a simulation.

MAPE-K analyze by MAPE-K plan. As one may expect from the vertical synthesis, the most common analyze method *logical inference* has the most co-occurrences with the most common planning method *determining the optimal choice*. However, this trend does not hold for *determining the optimal choice* when it comes to the second most common analyze method *system state anomaly detection*, which only has 1 co-occurrence. This entails that when an anomaly is the reason for adaptation authors tend not to rely on optimization to determine which adaptation to then choose. Rather it seems that when *system state anomaly detection* is done for analysis, *relying on design time rules/models* (4) and *using AI planning languages* (2) are more common co-occurrences. It is notable that *analyzing/aggregating data* always co-occurs with *using AI planning languages* meaning that at least for the two approaches with this planning method *analyzing/aggregating data* is a prerequisite, the same holds for the analysis method *system state anomaly detection*. The remaining analysis methods do not deviate from the trends established in the vertical synthesis of both the planning and analysis methods.

9. Discussion

Based on the results as reported in Sections 3–8, we discuss their implication in terms of potential research gaps for RSASSs. We structure this discussion in accordance with the demographics of the primary studies and the two research questions. We offer one takeaway for Section 3, two for RQ1 (as it has three sub-questions) and one for RQ2. The takeaways are based directly on the results discussed in the previous sections, either directly considering parameters from the classification framework (Tables 3 and 4) or the overall content of the 37 primary studies resulting from our selection process. For each takeaway we also offer concrete one or more concrete action points.

9.1. Demographics takeaways

Research in self-adaptive robotics is fragmented. From the onset, as this mapping study covers interdisciplinary research, we decided to search for the primary studies at venues that are specifically about robotics alongside those which one would expect to cover architecture-based self-adaptation, *i.e.,* venues for either software architecture or self-adaptive systems. This is based on the hypothesis that research in self-adaptive robotics at disparate venues would not necessarily be connected to one another by way of citations or publications by the same authors at both kinds of venues. Despite a higher percentage of selection happening at non-robotics venues, half of the primary studies prior to snowballing are still published at robotics venues. This confirms our hypothesis in that two broad communities are involved: (i) Roboticists applying self-adaptation as a solution to a problem they face in robotics, and (ii) Self-adaptive system researchers using robots as a case study/application for their approach. We can evidence this further, as we have managed to identify groupings of research around one specific approach as mentioned in Section 2.3. We can take as an example the ReFrESH approach (P3, P11, P18, P30): despite its four publications evidently being about architecture-based self-adaptation, none of these four publications are published in software architecture or self-adaptation venues. In terms of venues, there are only few recent efforts to combine two out of the three disciplines we focused on, for example the RSA workshop (Robotics Software Architecture) at ICRA[10] and the RoSE (Robotics Software Engineering) workshop at ICSE.[11] However, there is no venue that either focuses on architecture-based self-adaptation in robotics or more generally on self-adaptation in robotics. Having a dedicated venue would accelerate research progress by having dedicated reviewers that can bridge the gaps between the involved disciplines (something difficult to ensure in venues dedicated to only one of the three disciplines) and providing a forum for dedicated discussions on *e.g.,* the gaps identified in this study and related ones.

We suggest the following action point:

> **Research on RSASSs needs to be centralized.** There need to be efforts on the parts of researchers to publish at venues outside of the discipline from which they originate. This will lead to increased awareness of research efforts between communities.

9.2. RQ1 takeaways

9.2.1. Lack of Homogeneity within RSASSs

Our study had the overarching goal of characterizing the approaches to RSASSs. The aspiration was that certain patterns would emerge with respect to the design, development, and operation of RSASSs. However, our results indicate that there seems to be no definitive set of such patterns. In principle, we have found around 27 distinct

[10] International Conference on Robotics and Automation.
[11] International Conference on Software Engineering.

approaches (considering the groupings as mentioned in Section 2.3) to introduce architecture-based self-adaptation (in our 37 primary studies) in robotic systems. Few of those approaches share the same techniques or technologies to accomplish this. For example, in terms of software platform (6.4) the only common thread is the use of the quite general ROS communication middleware by a third of the primary studies. This points towards there being little reuse and potentially redundant work taking place, as distinct groups of researchers are creating solutions to similar problems, although with each iteration small biases being introduced for the use case at hand. This stems from a broader trend of these approaches being quite mission-specific. It seems that the prevalent motivation for RSASS is improving the performance of a specific mission that is hampered by uncertainty. This is also evidenced by the results in Sections 7 and 6.3, where mission-centric data dominates other options. The RSASSs represented in our primary studies are colored by the missions the authors consider and the related uncertainties addressed by self-adaptation.

Uncertainty can however theoretically play a role in any mission. Specifically, it seems infeasible for practitioners to account for every single eventuality prior to deployment. It is likely then that there are many robotics missions which suffer from uncertainties which are inefficiently addressed through maintenance and evolution of the system — *i.e.,* by taking it out of operation, investigating the encountered issues due to uncertainty, and making the system robust to it. What these cases would require is a set of patterns or a reference architecture which would allow for efficient introduction of self-adaptive capabilities to their systems. After all, the added benefit of handling uncertainty at runtime rather than design time needs to outweigh the cost of its introduction to the system to be appealing (Gerostathopoulos et al., 2022; Van Der Donckt et al., 2018). From our primary studies, we observe a distinct lack of any such patterns, meaning this constitutes a research gap. An honorable mention is the SHAGE framework, which is used in P32, but unfortunately not in any of the other primary studies. It seems that while the primary studies do all conform – implicitly or explicitly – to the reference architecture of MAPE-K, there is no deeper level which acknowledges the inherent challenges common to robotics applications.

Based on the above, we suggest the following action point:

Equal focus should be placed on consolidation as is to novel contributions in the field of RSASSs. Beside our own high-level overview, studies are required which identify design patterns, reference architectures, and guidelines for the development of non-trivial, feature-rich self-adaptive robotics systems.

9.2.2. Granularity of adaptation mechanisms

Within architecture-based self-adaptive systems, adaptation loops can be specified at different granularity levels (system, subsystem, individual component) (Weyns et al., 2013). In RSASSs, this concerns whether adaptations should be applied at a system-level, a task-level, or some combination of both. Practically, the latter proves necessary yet requires careful consideration between the adaptations between contesting tasks. For example, we cannot have the 'performance efficient' task A drain all the battery resources to be used by the 'energy efficient' task B. As it stands, we can state based on our findings in Section 6.1 that for the majority of the primary studies adaptation is done within the frame of reference of a singular task. It holds in principle then that a singular task of a robotic system can constitute an entire self-adaptive system in itself. Orthogonally, the adaptation carried out within one task may itself have a local or global scope (as seen in Section 5.1). Despite its indirect relation to our current point, we can surmise that adaptations with a *global* scope would be necessary to handle systems which have contested resources between two tasks. Among the primary studies, adapting with a global scope is relatively rare (only 11 primary studies out of 37). Therefore, despite not directly extracting data as to the granularity at which adaptation mechanisms act (task vs. system-level) we can say that it is unlikely that the approaches of our primary studies coordinate tasks with contested resources.

More substantively, we identify the set of studies P4, P9, P17, P32, and P33 which have the distinct consideration of the granularity of adaptation mechanisms of a robotics system as a primary concern of their approach. For example, in P4 the self-adaptation is performed as part of a two-step process. Firstly, adaptations of the system's architecture to meet imposed requirements as is traditional in self-adaptation is done. Secondarily, within the constraints of those adapted architectures, adaptations of the behavioral/functional (*e.g.,* moving from point A to B of the robot) aspects of the robot's task that reflect the circumstances of the architectural adaptations available from the first stage. For example, the state of the software architecture can prevent certain functional behaviors *e.g.,* changing the path driven by the robot depending on the sensors it has available, as navigating a dark corridor without a light is unsafe. Ultimately, despite the distinct consideration of task, the approach of P4 still operates within what is a singular task: navigation through a space. It remains to be seen how feasible this approach is when it comes to multiple tasks having to be performed by the robots, potentially even simultaneously. For example, if a parallel task is to pick up and deliver objects throughout the space, this will influence which paths can be taken, and ultimately even the architectural adaptations, potentially leading to a deadlock between the priorities of both stages of adaptation. For example, the architecture adapted into may make it so no paths which lead to the deliver location are available.

P9, P32, and P33 are all broadly examples of adapting the software architecture of a robotic system on the basis of the task it is currently assigned. For example, in P32 a robot may be asked to bring a user a glass of water, which then sees the software architecture modified to facilitate this task. The robot then needs to have the software component for the camera feed active to identify the glass of water. Quality attributes only play a role then when it comes to alternatives between architectures which accomplish said task. This is an almost complete reversal of hierarchy relative to P4 where the task execution is what was modified to suit architectural availability. What remains to be seen from these is which angle of approach works best for which kinds of RSASSs. Perhaps a third, higher level more comprehensive approach is plausible, which trades off the prioritization of architectural adaptations and task execution. Further, none of these prominently address the peaceful co-existence of multiple self-adaptive tasks of a system, either in parallel or sequentially.

To conclude, a self-adaptive robot would have to perform a multitude of distinct tasks during a realistic mission. Making only some of those tasks self-adaptive in isolation runs the risk of having unexpected system-level interactions between the individual adaptation loops. While this is recognized by five primary studies, we believe there is still ample space for approaches that consolidate the local and global views in a systematic and comprehensive way.

We offer the following action point:

Studies should recognize the ramifications of the integration of self-adaptation into robotics systems. As of yet the focus is too often laid on the initial endowment of self-adaptive capabilities. Few studies assume a self-adaptive robotic system, and then consider further complexities such as resolving conflicts between multiple types of adaptation and their interaction.

9.3. RQ2 takeaways

Violation of open science principles. Despite characterizing many RSASS approaches, in this study we are still not able to easily reproduce these approaches. There is a striking lack in the primary studies of availability of source code, replication packages, and low-level implementation

details. This is most evident in our data parameter which shows that one quarter of the approaches do not indicate any clear software platform, and another quarter uses mostly proprietary technology. A lack of replication packages of course can bring doubts about the soundness of results published, as they cannot be readily confirmed by a third party. For prospective authors trying to align themselves with the state of the art this is a disappointing result. Ideally, in future work we as the authors and other researchers would like to take inspiration from and directly extend the approaches we characterize. By not taking the effort to provide source code or replication details, the authors of those primary studies are slowing down progress in this field.

The action points for RQ2 are thus as follows:

Studies should be made reproducible by standardizing the execution environments. Utilization of efforts such as ROS dev containers[a] should become the norm. Containerizing a system eliminates dependency issues and makes it robust against becoming outdated due to updates to both libraries and operating systems. Further, it allows the deployment across operating systems, removing the barrier to the entry of Ubuntu-like operating systems often seen in robotics software.

[a] https://github.com/devrt/ros-devcontainer-vscode.

Efforts towards providing comprehensive and working replication packages should be valued and recognized at the relevant research venues. For example, the badge system of ACM[a] can be incorporated to indicate studies are reproducible. While this may be potentially time-consuming for the author they can incentivized *e.g.,* by increasing a papers position when in contention for existing awards such as 'best paper'.

[a] https://www.acm.org/publications/policies/artifact-review-and-badging-current.

Replication packages should provide complete instructions on how to replicate experiments and how to extend the work. To aid in the latter, open-source frameworks such as ROS should be utilized, when possible.

10. Threats to validity

We follow the classification of validity threats as defined by Wohlin et al. (2012) to discuss to threats to validity of our mapping study. We cover, in order, the internal validity (the extent to which causal conclusion can be made), the external validity (generalizability of findings), construct validity (use of correct measure and scope) and reliability (reproducibility of the results).

10.1. Internal validity

Internal validity refers to the extent to which a causal conclusion can be made based on the study. A potential threat to internal validity for this study stems from its interdisciplinary nature. It is plausible that due to their own biases, studies published by researchers in particular research communities (and their venues) influence the conclusions drawn by our study. For example, approaches published in self-adaptation venues may focus more on software aspects than on the hardware and low-level control aspects than in robotics venues and vice versa. We mitigate this threat by virtue of our study design, particularly the proceedings collection as outlined in Section 2.2.1. We made sure to select a representative number of both journals and conferences from each of the three disciplines. This selection process also poses a threat, as it was not done systematically but by relying on the knowledge and experience of the authors. This is mitigated as we ensured that each conference had a rank of B or more in the CORE ranking system should they be cataloged there. This is indicative of their quality independent of the potential biases of the authors. Finally, the representation of different perspectives on RSASS is further bolstered by the lenient search strings used, for example a potentially-relevant study in robotics needed only to mention either architecture or adaptation in their title. This was a conscious trade-off as this leniency inflated the number of potentially-relevant studies to select from heavily, especially from robotics venues.

10.2. External validity

External validity refers to the extent to which the findings can be generalized. In our study, we aim to generalize results for the research that lies in the intersection of three research domains: robotics, self-adaptive systems, and software architecture. We strive for an as representative as possible set of papers from this intersection. The proceedings collection, as also discussed in the previous subsection, may constrain the studies found. It is possible for example that potentially relevant studies are published at more niche venues than those chosen, especially ones which may be interdisciplinary. To mitigate this, we performed a round of snowballing on the studies initial selected. This removed the constraint of the original list of proceedings used for our automated search to now include those from any venue. By virtue of our inclusion criteria, we still ensure these are peer-reviewed studies. This was an effective mitigation strategy since ultimately more selected studies emerged from snowballing than the selected venues. Simultaneously, this demonstrates the value of the studies selected from the selected venues as they themselves cited and were cited by the snowballed studies. What remains as a threat then is that we did not do exhaustive snowballing, meaning even more studies may be out there that could have been selected.

10.3. Construct validity

Construct validity refers to the extent to which we obtained the right measure and whether we defined the right scope in relation to the topic of our study. A potential threat to construct validity comes in the interpretation of the potentially-relevant studies due to our inclusion criteria. Specifically, criterion I2 has this risk, as it states the study must "involve software architecture". The extent to which an individual study does so is not always clear-cut. This is exacerbated by the influence of how authors choose to present their work to particular venues, with the potential existing for authors at robotics venues to downplay the software architecture aspects of their approaches. To mitigate this, we had four of the authors involved in the selection of the studies at all stages. A pilot process was followed as detailed in Section 2.2.3 to establish the inter-rater agreement between each author was significant. Further, for all the primary studies selected afterwards the selection was reviewed by the other authors to ensure every inclusion and exclusion criterion was applied correctly. This was effective to such an extent that studies were also removed from the set of primary studies during data extraction phases, as this is when more detailed cross-reviewing of the selection was done. The classification framework also poses threats to the validity of the data extracted and by extension the study as a whole. Particularly, parameters with an open classification type require interpretation from the authors to turn the raw data extracted into the final set of labels determined. To mitigate the bias introduced during this interpretation, for every parameter we assigned a secondary person (from the authors) to review the labels assigned. Additionally, the original authors responsible were required to provide an elucidation for each data point. These can also be found in the raw data of our replication package.

10.4. Reliability

Reliability refers to the extent to which we can ensure that our results are the same if our study would be conducted again. A threat is related to the biases imposed by the researchers involved in the study. We mitigated this threat by using a protocol that was reviewed by an external expert. We also make all the data available for other researchers in a replication package (Replication, 2024) including a copy of the database used to perform the automated search resulting in the potentially-relevant studies as well the inclusion and exclusion criteria and how they were applied to each of these.

11. Related work

In this section we consider work related to our own study. Particularly, we consider recent examples of secondary studies which touch upon (partial) combinations of the three disciplines of robotics, software architecture and self-adaptation. While our novelty lies in the combination of all three of these disciplines, we believe it proves useful to compare and contrast against other studies which combine pairings among these three *e.g.,* software architectures and robotics.

Bozhinoski et al. (2019) perform a systematic mapping study specifically about the quality attribute of safety when it comes to mobile robots. Specifically, they identify 58 studies which propose software engineering approaches to ensure safety in mobile robotics systems. As we do in our study with self-adaptation approaches, the authors try to characterize the approaches which manage the safety of the system. In their study, the authors also consider the "potential for industrial adoption" of the studies, something which we have not, due to differing motivation for our study. We do however touch upon the theme slightly by extracting and analyzing the depth of evaluation and the system realness in its deployment and mission. Relevantly, in their study the authors report on a research gap regarding a lack of approaches to support adaptive behavior. In our study we explore that gap further, as we find that there are only 8 instances targeting safety as a quality attribute in our primary studies. That figure is in stark contrast to quality attributes such as performance efficiency and reliability which have 29 and 15 instances respectively.

Albonico et al. (2023) perform a systematic mapping study on software engineering research in systems using ROS, seeking to characterize the research and ascertain its potential for industrial adoption. Their study is related to ours in that it focuses on software engineering of robotics systems, yet their study is broader in some respects and narrower in others. Their study is broader in that it at a surface level considers all the topics of research in the interdisciplinary field of software engineering and robotics. In our study we add a third field of self-adaptation and focus specifically on the intersection of the three. Their study is narrower in that it exclusively considers ROS-based systems. While many of our approaches use ROS (15 of them), we have not selected our studies on the basis of it and therefore consider research done on a variety of software platforms. To conclude, topic-wise our study is a subset of theirs with a particular emphasis on architecture-based self-adaptation research. Moreover, Albonico et al. identify "enabling self-adaptation" as one of the recurring challenges in the studies they identified. One of their examples of this challenge is the same study labeled P31 in our work.

Peldszus et al. (2023) perform a systematic literature review of how robotics systems re-configure their software, and of the frameworks they utilize to enable this. To do so they first reviewed literature to identify 98 studies and used these to identify 48 software artifacts which involve reconfiguration. They report primarily on characteristics of the reconfiguration process itself. The study is quite related to ours in that it particularly focuses on the Execution phase of MAPE-K which we cover in Section 5.2. The results of the study by Peldszus et al. elucidate a means to an end of adaptation, the very last step in the process according to MAPE-K. There is some similarity in that they also extract the "reconfiguration logic" which is a superset of our adaptation logic. Additionally, the aspects of reconfigurations such as the "reasons", "time", "granularity" "lifespan" etc. are reminiscent of the modeling dimension by Andersson et al. (2009) we make use of, particularly those describing the **mechanism** of adaptation. A crucial distinction is that while the studies and artifacts they consider reconfigure themselves, they are not necessarily self-adaptive systems. Particularly, these systems do not necessarily fulfill the internal principle (Weyns, 2020) which requires the existence of a distinct entity responsible for adaptation concerns beside the robotic system which covers domain concerns. Additionally, they do not only consider systems which reconfigure on an architectural basis as we do.

Ahmad and Babar (2016) perform a systematic mapping study with the aim of taxonomizing the research done on robotic software architectures. Their study is similar in that it is also a cross-disciplinary look at software architecture and robotics, the distinction being our third added dimension of self-adaptation. The implication of the similarity is that there should be overlap between the results they obtained, particularly those of ours relating strictly to software architecture, such as in Sections 6.4, 5.2, among others. In their study, Ahmad and Babar identify four "architectural frameworks to support robotic software". These in principle can facilitate the architecture-based adaptation of robotics software, by providing an architectural representation to adapt through for example swapping out components. For example, they identify PRISM-MW which is used in P14 and P28, SHAGE which is used in P32, and ORCA which is mentioned in P1 as something the authors try to improve on. Particularly, SHAGE is relevant to self-adaptation and is only found by the authors to be mentioned in three studies; it is testament to the extent of our own study that we also identified it. The remainder of their study covers exclusively software architectural aspects of the studies *e.g.,* notation, architectural evaluation. While our study instead has a focus on self-adaptation aspects.

The systematic literature review by Muccini et al. (2016) is quite similar to ours in that they identify different approaches to architecture-based self-adaptation to identify its state of the art as we do. The type of self-adaptive systems they specifically targeted, *i.e.,* "cyber–physical" systems, is different from ours, although we also consider some types of robotics systems to be cyber–physical. This entails that the study is essentially a parallel one to ours. It is notable though that the study was completed in 2016 which is prior to about half our primary studies being published. They report that 2 of the 42 studies they identify are in the robotics domain. Unfortunately, looking closer at their data we can confirm that we did not select either of these in our own study. Further, many of the data points they extract from the studies are similar to ours. For example, the authors also extract quality attributes targeted by the approaches, details of the adaptation mechanisms, and how the systems are evaluated. Ultimately, the disparity in the type of self-adaptive system has the implication that results reported in these two studies (ours and that by Muccini et al.) describe quite different sets of systems.

In 2012 Weyns and Ahmad (2013) performed a systematic literature review of architecture-based self-adaptation. Clearly, their topic is broader than ours in that they do not focus on any particular application domain of self-adaptive systems. Relevant to our study is that 7 of the studies they identify are in the robotics domain, although there is at most two years of overlap between studies considered due to the inclusion criteria. We select studies as of 2011, while the study by Weyns and Ahmad selects up until the end of 2012. However, the fact that 7 studies in robotics were selected by the study despite this indicates our lower bound of time may have been somewhat too conservative. Despite our best efforts, we are unable to identify these 7 studies. Otherwise, we could potentially compare them to our own set of primary studies. It appears that in the data provided alongside the paper as well as in its text the authors fail to report on the titles or any other demographic details of the studies that would allow us to identify them. Among their results it is interesting is that they similarly

report that performance efficiency and reliability are among the most targeted quality attributes as we do, indicating that architecture-based self-adaptation in robotics is reflective of the broader field in that dimension.

Swanborn and Malavolta (2020) perform a systematic literature review particularly considering the quality attribute of energy efficiency and the impact of robotics systems on the consumption of energy. This is of particular relevance to mobile robots which need their operation to be sustained by a battery with finite capacity. For the 15 papers they identified, the authors consider how energy is measured, the application domain of the robots, and which parts of the systems consume the most energy. There are similarities to our own study in that we also consider the quality attributes of robotics systems, but from the purview of using these to motivate adaptation decisions. The primary difference is that the study by Swanborn and Malavolta is restrictive in scope than ours as they only focus on energy efficiency while we consider any quality attribute. For example, where they identify different metrics to quantify energy consumption such as frames per second per watt, we consider more generally what metrics are used to evaluate the system in our study. The metrics they identify would then fall under the *resource consumption* category of evaluation strategy as extracted in this study. Lastly, it is clear that their study neither considers software architecture nor self-adaptation as a discerning factor, looking only orthogonally at general robotics software system which measure their energy efficiency.

All in all, it is evident that while numerous studies consider the intersection of subsets of the three disciplines covered by our study, no study comprehensively addresses the intersection of all three. Related work which is broader in scope, such as that by Weyns and Ahmad, and Muccini et al. exist, while there are also studies which are more narrow such as that by Swanborn and Malavolta (2020). It is also notable that the related work covers a large timespan indicating that there is an enduring research interest in the disciplines at hand.

12. Conclusions and future work

In this section we give a concluding summary of our study and describe the future work we foresee following this research.

12.1. Conclusions

In our systematic mapping study, we have sought to identify, classify, and analyze the characteristics, feedback loops employed by, robotics platforms supported by, and evaluation strategies present in the literature on robotics software architecture-based self-adaptive systems (RSASSs). To do so we have systematically gathered and selected from a pool of 3087 potentially relevant studies to come to a set of 37 primary studies. These primary studies describe a variety of approaches to RSASSs. To determine their characteristics, we have used a comprehensive classification framework and extracted data from each primary study. The framework describes the primary studies from the perspectives of the self-adaptive system as a whole (RQ1.1), its managing system (RQ1.2), and managed system (RQ1.3). Additionally, we have considered the evaluation strategies used within these primary studies (RQ2) by the authors to validate their contributions.

For RQ1.1, we find that the goal of RSASSs when considered in their entirety is primarily to recover from errors and faults, while targeting quality attributes such as performance efficiency, reliability, safety, and functional suitability. We also conclude that the managing systems are mostly separable from the robotic systems they manage.

For RQ1.2, we find that managing systems of RSASSs primarily make use of mechanisms based on constraint solving, model checking, and search procedures. In doing so, both parametric and structural adaptations are enacted. The feedback loops employed by these managing systems tend to monitor their managed system much more often than the environment it interacts with or the tasks it has to perform. This information is then analyzed to determine the need to adapt through techniques such as logical inference and anomaly detection through comparison. When such a need arises, the feedback loop chooses an adaptation through optimization, or decision-making models such as rules defined prior to runtime.

For RQ1.3, we find that managed robotic systems of RSASSs can be characterized as primarily having missions related to navigation and emergency response. The missions assigned to these robots tend not to evolve at runtime. The changes in these robotic systems that lead to adaptation are primarily non-functional ones emerging from internal sources such as faults/errors. The models of robots which perform these missions vary widely, without any predominant model emerging.

For RQ2 we find that evaluations are performed primarily in real-world conditions, involving either synthetic or real missions. As a means of quantifying the performance of the system, we find that how well the mission is performed is most prevalent, followed by measures of overhead and quality of service. The evaluations display depth, in that a significant majority use empirical experiments with at least two system configurations to evaluate the proposed approaches. When it comes to being able to reproduce the evaluations for further validation, we find that only few primary studies provide the means to do so.

Based on the results obtained in answering each research question, we have determined a number of takeaways with concrete action points. Among these we identify a need for research into RSASS to become more centralized in terms of community. We find there should be a focus on consolidating the state of RSASSs (as we have partially done in this study) alongside the primarily novel contributions which exist now. We also find that the maturity in solving the complexities of RSASSs is lacking. We hence call upon researchers to consider the deeper implications of an RSASSs operating in realistic scenarios. Lastly, we call upon researchers to strive to make their RSASS systems reusable and their results reproducible by others. This not only strengthens the validity of their achieved results, but also provides other researchers with foundations upon which to build their own contributions. This can go hand in hand with addressing more complex issues which arise in the sophisticated use of RSASSs. For example, the balancing of mission and quality of service concerns, and the self-adaptation of robotic systems at differing levels of granularity.

12.2. Future work

Based on our results and discussion, we consider three avenues for future research. We consider how in what ways future studies can better consolidate the state of the art, broaden our consideration of the state of the art, and include more than just academic sources.

12.2.1. Consolidation

As was hinted upon in the action point of Section 9.2.1, potential future work lies into consolidating the state of the art of RSASSs at the design and implementation level. For example, a reference architecture can be devised based on our set of primary studies and others which recognizes the unique challenges of RSASSs compared to other types of self-adaptive systems. The creation of such a reference can provide what we have found to be a missing foundation upon which to develop and then research RSASSs. One potential way of doing so would be to look deeper into the set of primary studies covered in this study. By using the existing source code made available in few of them, and establishing lines of contact with the authors of others, one can start to extract design-time tactics/patterns of these types of systems as a whole. This would be similar to previous efforts such as the study by Malavolta et al. (2020). These kinds of design tacits/patterns can then be analyzed *e.g.,* to determine the quality of service they provide, and then organized in a catalogue of reusable best practices. In an ideal scenario, future developers of RSASSs are then equipped with a useful toolkit which helps them tackle recurrent problems and avoid common pitfalls in the implementation and operation of RSASSs.

Table 4
The classification framework with each possible value defined.

Parameter	Possible values	Definition
Adaptation goal	Change functional behavior	The actions of the system which relate to performing its function as specified by its mission are to be modified.
	Deal with environmental changes	Undesirable impacts of changes originating outside of the system are to be mitigated.
	Keep meeting quality requirements at runtime	A set of requirements imposed on the system by the user which relate to quality attributes of the system are to be satisfied continually.
	Optimize resource usage	The resources available to the system are to be allocated according to a defined sense of optimality.
	Optimize system performance	The efficiency by which the system performs its assigned mission is to be optimized.
	Recover from attacks	The system is to be robust towards undesirable changes originating through the malicious actions of external actors.
	Recover from errors/faults	The impact of an undesirable change originating within the system is to be mitigated.
Quality attributes	Functional Suitability	"capability of a product to provide functions that meet stated and implied needs of intended users when it is used under specified conditions" (ISO/IEC, 2023)
	Performance Efficiency	"capability of a product to perform its functions within specified time and throughput parameters and be efficient in the use of resources under specified conditions" (ISO/IEC, 2023)
	Reliability	"capability of a product to perform specified functions under specified conditions for a specified period of time without interruptions and failures" (ISO/IEC, 2023)
	Safety	"capability of a product under defined conditions to avoid a state in which human life, health, property, or the environment is endangered" (ISO/IEC, 2023)
	Security	"capability of a product to protect information and data so that persons or other products have the degree of data access appropriate to their types and levels of authorization, and to defend against attack patterns by malicious actors" (ISO/IEC, 2023)
Managing system independence	Detachable	The managing system can theoretically be used or is shown to be used with another managed system.
	Inseparable	The managing system is intertwined with the managed system to an extent that the boundary between the two is difficult to delineate.
	Requires Representation	The managing system operates using an explicit representation of the managed system which can theoretically be recreated for another managed system.
Mechanism — Method	AI Planner	Runtime decision-making is based on an AI planner (*e.g.,* PDDL).
	Constraint Solving/Model Checking	Runtime decision-making is based on constraint solving (*e.g.,* SAT) or model checking (*e.g.,* PRISM).
	Application-specific Algorithm	Runtime decision-making is based on adhoc logic specific to the application.
	Numerical Optimization	Runtime decision-making is based on solving a numerical optimization problem (see https://neos-guide.org/guide/types/).
	Ontological Reasoning	Runtime decision-making involves ontological modeling and reasoning (*e.g.,* OWL, SWRL).
	Search Procedure	Runtime decision-making is based on an ad-hoc or generic search algorithm (*e.g.,* genetic algorithm).
	Design-time Rules	Runtime decision-making is based on rules/actions (e.g. Event-Condition-Action rules, state transitions) provided at design time.
	Graph Transformation	Runtime decision-making is based on transforming or switching between graph structures representing the managed system.
Mechanism — Type	Parametric	Adaptation is related to the parameters of system components.
	Structural	Adaptation is related to the structuring of system components.
Mechanism — Organization	Centralized	Adaptation is done by a single component.
	Decentralized	Adaptation is distributed amongst multiple components.
Mechanism — Scope	Global	Adaptation affects the entirety of the system.
	Local	Adaptation is localized to one part of the system.
Mechanism — Duration	Short	The system is adapting for seconds to hours.
	Medium	The system is adapting for hours to months.
	Long	The system is adapting for months to years.
Mechanism — Timeliness	Best effort	It cannot be guaranteed that a planned adaptation is executed.
	Dependent	Whether a planned adaptation has guaranteed execution is not predictable and is influenced by external factors.
	Guaranteed	Every planned adaptation is guaranteed to be executed.
Mechanism — Trigger	Event-triggered	The change that triggers adaptation is associated with an event.
	Time-triggered	The change that triggers adaptation is associated with a time slot.
MAPE-K — Monitoring	Environment	The managing system monitors the real world environment in which the robotic system exists.
	Managed System	The managing system monitors the state (hardware or software) of the robotic system.
	Mission	The managing system monitors the state (e.g. progress, changes in, or transition between tasks) of the mission assigned to the robotic system.

(*continued on next page*)

12.2.2. Broadening the scope of study

In this study we have specifically targeted **architecture-based** self-adaptation in robotics. However, through the selection process we are aware that there are several examples of non-architectural adaptation being employed in robotics. These stem both from the robotics domain *e.g.,* adaptive control, and software engineering *e.g.,* self-organization. Particularly, we believe an umbrella term for self-organization in the form of task-based adaptation can be promising expansion of this study. Self-organizing systems, instead of relying on a runtime representation of the software architecture, reconsider their task planning at runtime.

Table 4 (*continued*).

MAPE-K — Analysis	Analyzing/Aggregating Data	The decision to adapt is made on the basis of a monitored data analysis and or aggregation such as through the use of statistical methods.
	Comparison To Threshold(s)	The decision to adapt is made on the basis comparing monitored values to (possibly pre-defined) dynamic or static thresholds.
	Done During Plan	The decision to adapt is inseparably made alongside the decision of which adaptation to enact within the planning phase of MAPE-K.
	Logical Inference	The decision to adapt is made on the basis of a reasoning over a formalized logic.
	System State Anomaly Detection	The decision to adapt is made on the basis of detecting an anomalous potentially detrimental state of the system.
	Task/User-Driven	The decision to adapt is made on the basis of a directive a by human user, or is the determined by the task plan of the system corresponding to its mission.
MAPE-K — Planning	Determining The Optimal Choice	The adaptation to enact is determined through continually determining which adaptation within a set of options is optimal as defined by the designer.
	Relying On Design-Time Rules/Models	The adaptation to enact is determined by relying on a model of the system or a set of rules which describe it, which is defined prior to deployment.
	Using AI Planning Languages	The adaptation to enact is determined by the output of solving a planning problem specified in an artificial intelligence planning language such as PDDL.
MAPE-K — Execution	Addition and/or Removal of Components	Adaptation(s) either make it so that an existing active component is now in a state in which it no longer affects or can be affect by other components (unload, delete, remove) or introduce a component which held that status prior to now be able to affect and be affected by other components.
	Change in Relationship(s) Between Components	Adaptation(s) make it so that a defined relationship between two or more components (parent and child, publish subscribe) is augmented by for example reversing hierarchy or flow or information, or introducing/removing components from the relationship.
	Component Redeployment	Adaptation(s) change which machine (hardware-wise) executes the process supporting a component without affecting its relationships with other components.
	Reparameterization of Components	Adaptation(s) change the values assigned to the parameters provided by a component.
MAPE-K — Knowledge	Component Model	An abstract representation removed from the code-level implementation which represents the system as a component-based system
	Knowledge Representation	Knowledge representation as it exists within artificial intelligence, is used to describe the software architecture of the system
	Variability Model	A representation of the variability points within the software architecture (e.g. which components fulfill the same role)
	Grammar/DSL	A syntax is defined for a language/grammar which describes the software architecture of the system.
	Behavior Model	A representation of adaptations as the possible actions (behavior) of components of the systems is used to manipulate its architecture e.g. a state machine
	Architecture Description Language	A computer language with the express purpose of describing, representing, and communicating a software architecture.
Mission	Approach Only	No mission is considered, only an approach for potentially performing missions is reported on.
	Emergency Response	The robotic system is tasked with aiding efforts in responding to an emergency situation with an uncertain environment such as one resulting from a disaster.
	Industrial Assembly	The robotic system is tasked with the creation of products in an industrial capacity.
	Mobile Manipulation	The robotic system involves the combination of manipulating objects (e.g., with a robotic arm) and navigating through a space.
	Navigation	The robotic system is tasked with moving itself from one point in a space to another.
	Object Tracking	The robotic system is tasked with continually determining the location of an object as it potentially moves throughout a space
	Service	The robotic system performs tasks which tend to involve the potential for interaction with humans while rendering a service.
Mission — Evolution	Dynamic	The mission assigned to the robotic system may change during operation.
	Static	The mission assigned to the robotic system does not change during operation.
Change — Source	External	The cause for adaptation originates outside of the system e.g. the environment around it.
	Internal	The cause for adaptation originates from within the system.
Change — Type	Functional	The cause for adaptation is related to the functionality the system provides.
	Non-functional	The cause for adaptation is related to quality concerns regarding the operation of the system.
	Technological	The cause for adaptation is related to the software and hardware which supports delivery of service.
Change — Anticipation	Foreseeable	The cause for adaptation is planned for prior.
	Foreseen	The cause for adaptation is already taken care of.
	Unforeseen	The cause for adaptation is not planned for.
Change — Frequency	Rare	The cause for adaptation emerges regularly yet rarely.
	Infrequent	The cause for adaptation emerges regularly yet infrequently.
	Frequent	The cause for adaptation emerges regularly with some frequency.
	Random	The cause for adaptation emerges stochastically i.e. irregularly
Effect — Criticality	Harmless	Should self-adaptation fail this has no significant drawbacks.
	Mission-critical	Should self-adaptation fail it jeopardizes completion of the mission assigned.
	Safety-critical	Should self-adaptation fail it jeopardizes the safety either of the robotic system itself or its surroundings.

(*continued on next page*)

For example, if due to a robot being stuck it can no longer perform a task of navigating, it can choose to move to perform a different task which does not require moving — like manipulation. To capture self-organization, one can then consider cases where upon the task being

Table 4 (*continued*).

Effect — Predictability	Deterministic	The consequence of an adaptation is predictable.
	Non-deterministic	The consequence of an adaptation possibly unpredictable *i.e.,* no guarantees are provided.
Effect — Overhead	Significant	The system's performance is effectively significantly by performing self-adaptation.
	Insignificant	The system's performance is effectively insignificantly by performing self-adaptation.
	Failure	The system fails to deliver service upon performing self-adaptation.
	Dependent	The impact on the system of an adaptation is not predictable, and is influenced by external factors.
Effect — Resilience	Resilient	The system is resilient to some kinds of changes faced.
	Irresilient	The system is not resilient to some kinds of changes faced.
	Vulnerable	The system is vulnerable to all changes faced.
	Dependent	The resilience of the system to a particular change is not predictable, and is influenced by external factors.
Software platform	ROS1/2	The Robot Operating System, see https://www.ros.org/
Type of robots	Turtlebot	https://www.turtlebot.com/
	NAO	https://www.aldebaran.com/en/nao
	HexManipulator	Ad-hoc, see primary studies P11 and P18 for the details.
System deployment	Combined	The robotic system as evaluated is considered both in simulated capacity and in the real world, integrated or not.
	Real	The robotic system as evaluated is only considered in the real world (within the study).
	Simulated	The robotic system as evaluated is only considered in simulated capacity and not in the real world.
System realism	Real	The evaluation scenario stems from a clear real-world application and the evaluation is performed in a context which is true to that scenario.
	Synthetic	The evaluation is simplified by introducing synthetic elements such as a lab environment, which simplifies the context in which it performs.
Evaluation metric	Quality	The quality of service provided by the system is quantified independent of its mission is used to evaluate the approach.
	Mission Performance	A metric which indicates progress towards or quality in completion of the mission is used to evaluate the approach.
	Overhead (Introduced)	The overhead introduced by the approach proposed in the study is measured to evaluate it.
	Domain-Specific Performance	A metric specific and pertinent to the domain the mission assigned belongs to is used to evaluate the approach.
	Resource Consumption	The consumption of a (potentially finite) resource by the system over time is used to evaluate the approach.
Evaluation depth	No Evaluation	No evaluation section or any other form of evaluative reporting regarding the proposed approach is provided in the study.
	Showcase	The evaluation of the proposed approach is performed with a singular experiment configuration (e.g., independent variables).
	Experiment	The evaluation of the proposed approach is performed with a quantitative comparison between at least two experiment configurations.
Replication package	Absent	No reference or link to replication material is provided which pertains to the proposed approach in the primary study.
	Present	The primary study provides a reference or link to some for of replication (source code, data) material regarding the proposed approach.

impossible, the task is reassigned to a different robot/agent which is not stuck. The consideration of these different cases for self-adaptation not only cover what is an unexplored angle relative to this study, but can also complement the results of this study by allowing a comparison and contrast between the characteristics extracted.

12.2.3. Going beyond academia

Considering the prevalence of robotics in industry, it seems an especially promising future research direction to consider how self-adaptation is used for these robots. Such research could take a similar form as the one conducted by Weyns et al. (2023) which comprised of a survey regarding the use of self-adaptation with industry professionals. Collecting empirical evidence via case studies, surveys and interview studies is instrumental in understanding the real problems and solutions of the state of the practice, and can act as a solid basis for impactful research contributions. It is undoubtedly so that robotics in industry face uncertainties, and that companies remedy these to prolong operation. However, there is no natural incentive for those companies to share this technology openly. It would be interesting for example to use the classification framework of this study and apply it to a set of systems used in industry which are also using self-adaptation. On a more general note, this kind of exchange of knowledge strengthens the efforts of both parties, as academics become privy to the requirements of realistic applications, and practitioners can incorporate novel contributions from research.

CRediT authorship contribution statement

Elvin Alberts: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Ilias Gerostathopoulos:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Data curation, Conceptualization. **Ivano Malavolta:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Carlos Hernández Corbato:** Writing – review & editing, Supervision, Investigation, Funding acquisition, Data curation, Conceptualization. **Patricia Lago:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Carlos Hernandez Corbato reports a relationship with Ahold Delhaze that includes: funding grants. co-author served on editorial board of Journal of Systems and Software - Patricia Lago If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Classification framework definitions

See Table 4.

Data availability

We provide a link to a replication package in the paper.

References

Ahmad, Aakash, Babar, Muhammad Ali, 2016. Software architectures for robotic systems: A systematic mapping study. J. Syst. Softw. 122, 16–39.

Albonico, Michel, Dordevic, Milica, Hamer, Engel, Malavolta, Ivano, 2023. Software engineering research on the robot operating system: A systematic mapping study. J. Syst. Softw. 197, 111574.

Andersson, Jesper, De Lemos, Rogerio, Malek, Sam, Weyns, Danny, 2009. Modeling dimensions of self-adaptive software systems. Softw. Eng. Self-Adapt. Syst. 27–47.

Bass, Len, Clements, Paul, Kazman, Rick, 2003. Software Architecture in Practice. Addison-Wesley Professional.

Behery, Mohamed, Trinh, Minh, Brecher, Christian, Lakemeyer, Gerhard, 2023. Self-optimizing agents using mixed initiative behavior trees. In: 2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, IEEE, pp. 97–103.

Bozhinoski, Darko, Aguado, Esther, Oviedo, Mario Garzon, Hernandez, Carlos, Sanz, Ricardo, Wąsowski, Andrzej, 2021. A modeling tool for reconfigurable skills in ROS. In: 2021 IEEE/ACM 3rd International Workshop on Robotics Software Engineering. RoSE, pp. 25–28.

Bozhinoski, Darko, Di Ruscio, Davide, Malavolta, Ivano, Pelliccione, Patrizio, Crnkovic, Ivica, 2019. Safety for mobile robotic systems: A systematic mapping study from a software engineering perspective. J. Syst. Softw. 151, 150–179.

Bozhinoski, Darko, Wijkhuizen, Jasper, 2021. Context-based navigation for ground mobile robot in semi-structured indoor environment. In: 2021 Fifth IEEE International Conference on Robotic Computing. IRC, pp. 82–86.

Brooks, Rodney A., 1992. Artificial life and real robots. In: Proceedings of the First European Conference on Artificial Life. pp. 3–10.

Brugali, Davide, 2020. Runtime reconfiguration of robot control systems: a ROS-based case study. In: 2020 Fourth IEEE International Conference on Robotic Computing. IRC, pp. 256–262.

Brugali, Davide, 2023. Modeling variability in self-adapting robotic systems. Robot. Auton. Syst. 167, 104470.

Brugali, Davide, Capilla, Rafael, Mirandola, Raffaela, Trubiani, Catia, 2018. Model-based development of qos-aware reconfigurable autonomous robotic systems. In: 2018 Second IEEE International Conference on Robotic Computing. IRC, pp. 129–136.

Brun, Yuriy, Di Marzo Serugendo, Giovanna, Gacek, Cristina, Giese, Holger, Kienle, Holger, Litoiu, Marin, Müller, Hausi, Pezzè, Mauro, Shaw, Mary, 2009. Engineering self-adaptive systems through feedback loops. Softw. Eng. Self-Adapt. Syst. 48–70.

Caldiera, Victor R. Basili1 Gianluigi, Rombach, H. Dieter, 1994. The goal question metric approach. Ency. Softw. Eng. 528–532.

Cámara, Javier, Schmerl, Bradley, Garlan, David, 2020. Software architecture and task plan co-adaptation for mobile service robots. In: Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS '20, Association for Computing Machinery, New York, NY, USA, pp. 125–136.

Cervera, Enric, 2018. Try to start it! the challenge of reusing code in robotics research. IEEE Robot. Autom. Lett. 4 (1), 49–56.

Cooray, Deshan, Kouroshfar, Ehsan, Malek, Sam, Roshandel, Roshanak, 2013. Proactive self-adaptation for improving the reliability of mission-critical, embedded, and mobile software. IEEE Trans. Softw. Eng. 39 (12), 1714–1735.

Cruzes, Daniela S., Dyba, Tore, 2011. Recommended steps for thematic synthesis in software engineering. In: 2011 International Symposium on Empirical Software Engineering and Measurement. pp. 275–284.

Cui, Yanzhe, Lane, Joshua T., Voyles, Richard M., 2015a. Real-time software module design framework for building self-adaptive robotic systems. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 2597–2602.

Cui, Yanzhe, Voyles, Richard M., He, Miao, Jiang, Guangying, Mahoor, Mohammad H., 2012. A self-adaptation framework for resource constrained miniature search and rescue robots. In: 2012 IEEE International Symposium on Safety, Security, and Rescue Robotics. SSRR, pp. 1–6.

Cui, Yanzhe, Voyles, Richard M, Lane, Josh T, Krishnamoorthy, Akshay, Mahoor, Mohammad H, 2015b. A mechanism for real-time decision making and system maintenance for resource constrained robotic systems through ReFrESH. Auton. Robots 39, 487–502.

Cui, Yanzhe, Voyles, Richard M., Lane, Joshua T., Mahoor, Mohammad H., 2014. ReFrESH: A self-adaptation framework to support fault tolerance in field mobile robots. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1576–1582.

Dasgupta, Bhaskar, Mruthyunjaya, TS1739334, 2000. The Stewart platform manipulator: a review. Mech. Mach. Theory 35 (1), 15–40.

David, Istvan, Latifaj, Malvina, Pietron, Jakob, Zhang, Weixing, Ciccozzi, Federico, Malavolta, Ivano, Raschke, Alexander, Steghofer, Jan-Philipp, Hebig, Regina, 2023. Blended modeling in commercial and open-source model-driven software engineering tools: A systematic study. Softw. Syst. Model. 22 (1), 415–447.

de Lemos, Rogério, Giese, Holger, Müller, Hausi A., Shaw, Mary, Andersson, Jesper, Litoiu, Marin, Schmerl, Bradley, Tamura, Gabriel, Villegas, Norha M., Vogel, Thomas, Weyns, Danny, Baresi, Luciano, Becker, Basil, Bencomo, Nelly, Brun, Yuriy, Cukic, Bojan, Desmarais, Ron, Dustdar, Schahram, Engels, Gregor, Geihs, Kurt, Göschka, Karl M., Gorla, Alessandra, Grassi, Vincenzo, Inverardi, Paola, Karsai, Gabor, Kramer, Jeff, Lopes, Antónia, Magee, Jeff, Malek, Sam, Mankovskii, Serge, Mirandola, Raffaela, Mylopoulos, John, Nierstrasz, Oscar, Pezzè, Mauro, Prehofer, Christian, Schäfer, Wilhelm, Schlichting, Rick, Smith, Dennis B., Sousa, João Pedro, Tahvildari, Ladan, Wong, Kenny, Wuttke, Jochen, 2013. Software engineering for self-adaptive systems: A second research roadmap. In: de Lemos, Rogério, Giese, Holger, Müller, Hausi A., Shaw, Mary (Eds.), Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–32.

de Leng, Daniel, Heintz, Fredrik, 2016. Dyknow: A dynamically reconfigurable stream reasoning framework as an extension to the robot operating system. In: 2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots. SIMPAR, pp. 55–60.

de Leng, Daniel, Heintz, Fredrik, 2017. Towards adaptive semantic subscriptions for stream reasoning in the robot operating system. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 5445–5452.

Di Francesco, Paolo, Lago, Patricia, Malavolta, Ivano, 2019. Architecting with microservices: A systematic mapping study. J. Syst. Softw. 150, 77–97.

Dietrich, Franz, Maaß, Jochen, Hagner, Matthias, Steiner, Jens, Goltz, Ursula, Raatz, Annika, 2013. Dynamic distribution of robot control components under hard realtime constraints – Modeling, experimental results and practical considerations. J. Syst. Archit. 59 (10, Part C), 1047–1066, URL https://www.sciencedirect.com/science/article/pii/S1383762113000027. Embedded Systems Software Architecture.

Doose, David, Grand, Christophe, Lesire, Charles, 2017. MAUVE runtime: A component-based middleware to reconfigure software architectures in real-time. In: 2017 First IEEE International Conference on Robotic Computing. IRC, pp. 208–211.

Edrisi, Farid, Perez-Palacin, Diego, Caporuscio, Mauro, Giussani, Samuele, 2023. Adaptive controllers and digital twin for self-adaptive robotic manipulators. In: 2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, IEEE, pp. 56–67.

Esfahani, Naeem, Kouroshfar, Ehsan, Malek, Sam, 2011. Taming uncertainty in self-adaptive software. In: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering. In: ESEC/FSE '11, Association for Computing Machinery, New York, NY, USA, pp. 234–244.

Eymuller, Christian, Hanke, Julian, Poeppel, Alexander, Wanninger, Constantin, Reif, Wolfgang, 2023. RealCaPP: Real-time capable plug & produce service architecture for distributed robot control. In: 2023 Seventh IEEE International Conference on Robotic Computing. IRC, pp. 352–355.

Franzago, Mirco, Di Ruscio, Davide, Malavolta, Ivano, Muccini, Henry, 2018. Collaborative model-driven software engineering: a classification framework and a research map. IEEE Trans. Softw. Eng. 14 (12), 1146–1175.

Franzosi, Roberto, 2010. Quantitative Narrative Analysis. Sage, Number 162.

Garlan, David, Cheng, S-W, Huang, A-C, Schmerl, Bradley, Steenkiste, Peter, 2004. Rainbow: Architecture-based self-adaptation with reusable infrastructure. Computer 37 (10), 46–54.

Garousi, Vahid, Felderer, Michael, Mäntylä, Mika V., 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf. Softw. Technol. 106, 101–121.

Garousi, Vahid, Mäntylä, Mika V., 2016. A systematic literature review of literature reviews in software testing. Inf. Softw. Technol. 80, 195–216.

Geihs, Kurt, Reichle, Roland, Wagner, Michael, Khan, Mohammad Ullah, 2009. Modeling of context-aware self-adaptive applications in ubiquitous and service-oriented environments. Softw. Eng. Self-Adapt. Syst. 146–163.

Gerostathopoulos, Ilias, Raibulet, Claudia, Alberts, Elvin, 2022. Assessing self-adaptation strategies using cost-benefit analysis. In: 2022 IEEE 19th International Conference on Software Architecture Companion. ICSA-C, IEEE, pp. 92–95.

Gerostathopoulos, Ilias, Skoda, Dominik, Plasil, Frantisek, Bures, Tomas, Knauss, Alessia, 2016. Architectural homeostasis in self-adaptive software-intensive cyber-physical systems. In: Tekinerdogan, Bedir, Zdun, Uwe, Babar, Ali (Eds.), Software Architecture. Springer International Publishing, Cham, pp. 113–128.

Gerostathopoulos, Ilias, Skoda, Dominik, Plasil, Frantisek, Bures, Tomas, Knauss, Alessia, 2019. Tuning self-adaptation in cyber-physical systems through architectural homeostasis. J. Syst. Softw. 148, 37–55, URL https://www.sciencedirect.com/science/article/pii/S016412121830236X.

Gerostathopoulos, Ilias, Vogel, Thomas, Weyns, Danny, Lago, Patricia, 2021. How do we evaluate self-adaptive software systems?: A ten-year perspective of SEAMS. In: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, IEEE, pp. 59–70.

Gherardi, Luca, Hochgeschwender, Nico, 2015. RRA: Models and tools for robotics runtime adaptation. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 1777–1784.

Haslum, Patrik, Lipovetzky, Nir, Magazzeni, Daniele, Muise, Christian, Brachman, Ronald, Rossi, Francesca, Stone, Peter, 2019. An Introduction to the Planning Domain Definition Language. Vol. 13, Springer.

Heinzemann, Christian, Priesterjahn, Claudia, Becker, Steffen, 2012. Towards modeling reconfiguration in hierarchical component architectures. In: Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering. CBSE '12, Association for Computing Machinery, New York, NY, USA, pp. 23–28.

Hochgeschwender, Nico, Schneider, Sven, Voos, Holger, Bruyninckx, Herman, Kraetzschmar, Gerhard K., 2016. Graph-based software knowledge: Storage and semantic querying of domain models for run-time adaptation. In: 2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots. SIMPAR, pp. 83–90.

Hristozov, Anton D, Matson, Eric T, Gallagher, John C, Rogers, Marcus, Dietz, Eric, 2022. Resilient architecture framework for robotic systems. In: 2022 International Conference Automatics and Informatics. ICAI, IEEE, pp. 18–23.

International Federation of Robotics, 2022. World robotics report 2022.

ISO/IEC, 2023. Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (Square) — Product Quality Model. Standard, International Organization for Standardization, Geneva, CH, ISO 25010:2023(en).

Jackson, Daniel, 2011. Software Abstractions, Revised Edition: Logic, Language, and Analysis. MIT Press.

Jamshidi, Pooyan, Cámara, Javier, Schmerl, Bradley, Käestner, Christian, Garlan, David, 2019. Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots. In: 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, pp. 39–50.

Kent, Daniel, McKinley, Philip K., Radha, Hayder, 2020. Localization uncertainty-driven adaptive framework for controlling ground vehicle robots. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 7079–7086.

Kephart, Jeffrey O., Chess, David M., 2003. The vision of autonomic computing. Computer 36 (1), 41–50.

Kitchenham, Barbara, Brereton, Pearl, 2013. A systematic review of systematic review process research in software engineering. Inf. Softw. Technol. 55 (12), 2049–2075.

Kitchenham, Barbara Ann, Charters, Stuart, 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, URL https://www.elsevier.com/_data/promis_misc/525444systematicreviewsguide.pdf.

Li, Jialong, Yamauchi, Takuto, Li, Nianyu, Chen, Zhengyin, Zhang, Mingyue, Hirano, Takanori, Tei, Kenji, 2023. Demonstration of a real-world self-adaptive robot path-finding using discrete controller synthesis. In: 2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion. ACSOS-C, pp. 27–28.

Loigge, Stefan, Mühlbacher, Clemens, Steinbauer, Gerald, Gspandl, Stephan, Reip, Michael, 2017. A model-based fault detection, diagnosis and repair for autonomous robotics systems. In: OAGM/AAPR ARW 2017: Joint Workshop on "Vision, Automation & Robotics".

Lotz, Alex, Inglés-Romero, Juan F., Vicente-Chicote, Cristina, Schlegel, Christian, 2013. Managing run-time variability in robotics software by modeling functional and non-functional behavior. In: Nurcan, Selmin, Proper, Henderik A., Soffer, Pnina, Krogstie, John, Schmidt, Rainer, Halpin, Terry, Bider, Ilia (Eds.), Enterprise, Business-Process and Information Systems Modeling. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 441–455.

Macenski, Steven, Foote, Tully, Gerkey, Brian, Lalancette, Chris, Woodall, William, 2022. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 7 (66), eabm6074.

Malavolta, Ivano, Lewis, Grace, Schmerl, Bradley, Lago, Patricia, Garlan, David, 2020. How do you architect your robots? State of the practice and guidelines for ROS-based systems. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice. pp. 31–40.

McHugh, Mary L., 2012. Interrater reliability: the kappa statistic. Biochem. Med. 22 (3), 276–282.

Muccini, Henry, Sharaf, Mohammad, Weyns, Danny, 2016. Self-adaptation for cyber-physical systems: a systematic literature review. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 75–81.

Niemczyk, Stefan, Geihs, Kurt, 2015. Adaptive run-time models for groups of autonomous robots. In: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 127–133.

Niemczyk, Stefan, Opfer, Stephan, Fredivianus, Nugroho, Geihs, Kurt, 2017. ICE: Self-configuration of information processing in heterogeneous agent teams. In: Proceedings of the Symposium on Applied Computing. SAC '17, Association for Computing Machinery, New York, NY, USA, pp. 417–423.

Oreizy, Peyman, Medvidovic, Nenad, Taylor, Richard N., 1998. Architecture-based runtime software evolution. In: Proceedings of the 20th International Conference on Software Engineering. IEEE, pp. 177–186.

Park, Yu-Sik, Koo, Hyung-Min, Ko, In-Young, 2012. A task-based and resource-aware approach to dynamically generate optimal software architecture for intelligent service robots. Softw. - Pract. Exp. 42 (5), 519–541.

Peldszus, Sven, Brugali, Davide, Strüber, Daniel, Pelliccione, Patrizio, Berger, Thorsten, 2023. Software reconfiguration in robotics. arXiv preprint arXiv:2310.01039.

Petersen, Kai, Feldt, Robert, Mujtaba, Shahid, Mattsson, Michael, 2008. Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12. pp. 1–10.

Petersen, Kai, Vakkalanka, Sairam, Kuzniarz, Ludwik, 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Softw. Technol. 64, 1–18.

Popay, Jennie, Roberts, Helen, Sowden, Amanda, Petticrew, Mark, Arai, Lisa, Rodgers, Mark, Britten, Nicky, Roen, Katrina, Duffy, Steven, et al., 2006. Guidance on the conduct of narrative synthesis in systematic reviews. In: A Product from the ESRC Methods Programme Version. Vol. 1, p. b92.

Pradhan, Subhav, Dubey, Abhishek, Levendovszky, Tihamer, Kumar, Pranav Srinivas, Emfinger, William A, Balasubramanian, Daniel, Otte, William, Karsai, Gabor, 2016. Achieving resilience in distributed software systems via self-reconfiguration. J. Syst. Softw. 122, 344–363.

Replication package of this study. 2024. Online. https://doi.org/10.5281/zenodo.13886651.

Sanchez, Jose, Schneider, Sven, Hochgeschwender, Nico, Kraetzschmar, Gerhard K., Plöger, Paul G., 2016. Context-based adaptation of in-hand slip detection for service robots. IFAC-PapersOnLine 49 (15), 266–271, URL https://www.sciencedirect.com/science/article/pii/S2405896316310461, 9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016.

Siciliano, Bruno, Khatib, Oussama, 2016. Robotics and the handbook. In: Springer Handbook of Robotics. Springer, pp. 1–6.

Silva, Gustavo Rezende, Päßler, Juliane, Zwanepol, Jeroen, Alberts, Elvin, Tarifa, S. Lizeth Tapia, Gerostathopoulos, Ilias, Johnsen, Einar Broch, Corbato, Carlos Hernández, 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, pp. 181–187.

Swanborn, Stan, Malavolta, Ivano, 2020. Energy efficiency in robotics software: a systematic literature review. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering Workshops. pp. 144–151.

Turner, Roy M., 1998. Context-mediated behavior for intelligent agents. Int. J. Hum.-Comput. Stud. 48 (3), 307–330.

Valdezate, Alejandro, Capilla, Rafael, Crespo, Jonathan, Barber, Ramón, 2022. Ruva: A runtime software variability algorithm. IEEE Access 10, 52525–52536.

Van Der Donckt, M Jeroen, Weyns, Danny, Iftikhar, M Usman, Singh, Ritesh Kumar, 2018. Cost-benefit analysis at runtime for self-adaptive systems applied to an internet of things application.. In: ENASE. pp. 478–490.

Wang, Chundong, Tok, Yee Ching, Poolat, Rohini, Chattopadhyay, Sudipta, Elara, Mohan Rajesh, 2021. How to secure autonomous mobile robots? An approach with fuzzing, detection and mitigation. J. Syst. Archit. 112, 101838, URL https://www.sciencedirect.com/science/article/pii/S1383762120301302.

Weyns, Danny, 2019. Software engineering of self-adaptive systems. Handb. Softw. Eng. 399–443.

Weyns, Danny, 2020. An Introduction to Self-Adaptive Systems: a Contemporary Software Engineering Perspective. John Wiley & Sons.

Weyns, Danny, Ahmad, Tanvir, 2013. Claims and evidence for architecture-based self-adaptation: A systematic literature review. In: European Conference on Software Architecture. Springer, pp. 249–265.

Weyns, Danny, Gerostathopoulos, Ilias, Abbas, Nadeem, Andersson, Jesper, Biffl, Stefan, Brada, Premek, Bures, Tomas, Di Salle, Amleto, Galster, Matthias, Lago, Patricia, et al., 2023. Self-adaptation in industry: A survey. ACM Trans. Auton. Adapt. Syst. 18 (2), 1–44.

Weyns, Danny, Malek, Sam, Andersson, Jesper, 2012. FORMS: Unifying reference model for formal specification of distributed self-adaptive systems. ACM Trans. Auton. Adapt. Syst. 7 (1).

Weyns, Danny, Schmerl, Bradley, Grassi, Vincenzo, Malek, Sam, Mirandola, Raffaela, Prehofer, Christian, Wuttke, Jochen, Andersson, Jesper, Giese, Holger, Göschka, Karl M., 2013. On patterns for decentralized control in self-adaptive systems. In: de Lemos, Rogério, Giese, Holger, Müller, Hausi A., Shaw, Mary (Eds.), Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers. In: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 76–107.

Wohlin, Claes, 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. pp. 1–10.

Wohlin, Claes, Runeson, Per, Höst, Martin, Ohlsson, Magnus C, Regnell, Björn, Wesslén, Anders, 2012. Experimentation in Software Engineering. Springer Science & Business Media.

Zaman, Safdar, Ahmad, Farhan, Qasim Khan, Mohammad, Ali Shah, Shabir, Jabeen, Asma, Aftab, Nouman, 2019. Fault detection using sensors data trends for autonomous robotic mapping. In: 2019 International Conference on Engineering and Emerging Technologies. ICEET, pp. 1–6.

Zaman, Safdar, Steinbauer, Gerald, Maurer, Johannes, Lepej, Peter, Uran, Suzana, 2013. An integrated model-based diagnosis and repair architecture for ROS-based robot systems. In: 2013 IEEE International Conference on Robotics and Automation. pp. 482–489.

Zou, Yong-Hao, Bai, Jia-Ju, 2021. Effective crash recovery of robot software programs in ROS. In: 2021 IEEE International Conference on Robotics and Automation. ICRA, pp. 9498–9504.

Elvin Alberts is a Ph.D. Candidate affiliated with both the Software and Sustainability research group of the Computer Science department at the Vrije Universiteit Amsterdam as well as the Knowledge-based Autonomous Systems Laboratory of the Cognitive Robotics department at the Technical University of Delft. He received a M.Sc. in Software Engineering at the University of Amsterdam and a B.Sc. in Computer Science at the Vrije Universiteit Amsterdam. His research focuses on the development and integration of self-adaptive capabilities for robotics software.

Ilias Gerostathopoulos is an Assistant Professor of computer science with Vrije Universiteit Amsterdam, The Netherlands. His research interests include software engineering, software architecture, and self-adaptive systems. He received the Ph.D. degree in computer science in 2015 from the Department of Distributed and Dependable Systems, Faculty of Mathematics and Physics, Charles University, Prague. He was also as a Postdoctoral Researcher with the Department of Informatics, Technical University of Munich.

Ivano Malavolta is an Associate professor in the Software and Sustainability research group and Director of the Network Institute at the Vrije Universiteit Amsterdam (The Netherlands). His research focuses on (empirical) software engineering, with a special emphasis on green software, software architecture, robotics software. He authored more than 150 scientific articles in international journals and peer-reviewed international conference proceedings. He is a program committee member and reviewer of international conferences and journals in the software engineering field. He received a Ph.D. in computer science from the University of L'Aquila in 2012. He is a member of ACM, IEEE, VERSEN, and Amsterdam Data Science.

Carlos Hernández Corbato is Associate Professor at the Cognitive Robotics Department in the Faculty Mechanical Engineering of TU Delft, The Netherlands. He participates in the EU projects CoreSense, METATOOL, and REMARO, has served as scientific coordinator in other EU projects, and won with Team Delft the Amazon Robotics Challenge 2016. His research interests include software architectures for robotics, knowledge representation and reasoning, model-based systems engineering, and self-adaptive systems, and teaches about these topics in the M.Sc. Robotics Program. Carlos holds M.Sc. degrees in engineering (2006) and automation and robotics (2008), and a Ph.D (2013) from Universidad Politecnica de Madrid.

Patricia Lago is full professor in software engineering at Vrije Universiteit Amsterdam, where she founded the Software and Sustainability research group in the Computer Science Department, and the DiSC — Digital Sustainability Center. Her research focuses primarily on software architecture design and decision making, software quality assessment, and software sustainability. She is the recipient of an Honorary Doctorate at NTNU, Norway, for her contribution to the field of software sustainability, and of the 2023 IEEE-CS-TCSE New Directions Award. She has a Ph.D. in Control and Computer Engineering (Politecnico di Torino) and a Master in Computer Science (University of Pisa).