Self-adaptive Middleware Support for IoT and CPS

A Systematic Literature Review

MAHYAR T. MOGHADDAM, Univ. Grenoble Alpes, Inria, CNRS, LIG, France
ERIC RUTTEN, Univ. Grenoble Alpes, Inria, CNRS, LIG,, France
GUILLAUME GIRAUD, RTE France, France

This review classifies and analyzes studies on self-adaptation mechanisms for IoT/CPS. The role of middleware platforms to facilitate such self-adaptation is highlighted. We applied the standard extraction framework to select 62 papers among 4,274 candidate studies. We further analyzed three CPS4EU project's industrial use-cases based on the review outcomes to propose improvement solutions. Main findings are: *i)* the adaptation requirement may arise due to changes in system, environment, and their coordination; *ii)* data-driven proactive adaptation approaches are newly getting more attention; *iii)* the potential industrial adoption of middleware platforms depends on industry requirements and platforms' design approaches.

CCS Concepts: • **Software and its engineering** → *Requirements analysis*; *Software configuration management and version control systems.*

Additional Key Words and Phrases: Internet of Things, Industrial Internet of Things, Cyber-physical systems, Pervasive Computing, Middleware, Self-adaptation, Systematic Literature Review.

ACM Reference Format:
Mahyar T. Moghaddam, Eric Rutten, and Guillaume Giraud. 2020. Self-adaptive Middleware Support for IoT and CPS: A Systematic Literature Review. In ,. ACM, New York, NY, USA, 35 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The Internet of Things (IoT) and Cyber-Physical Systems (CSP) are prominent due to the current need for massive digitization and opening new market opportunities. IoT/CPS include massive devices across various domains that are interconnected to exchange data and provide services. Such domains mainly present intelligent services such as smart buildings [37, 61], smart healthcare [24, 36], smart grid [12, 35] and smart transport [46]. All the above mentioned IoT/CPS-based systems are located in the environment with which they coordinate.

Self-adaptation methods equip software systems with capabilities to cope with environmental and contextual changes occurring in real-time. In real-life IoT/CPS, while all systems are labeled as real-time, they vary in criticality concerning the real-time attribute satisfaction. Self-adaptation techniques can guarantee the dynamic nature of real-time collaborative systems. Self-adaptation is typically performed by control elements that interact with the system components and the environment to enhance the quality of service (QoS). Self-adaptive IoT/CPS are exposed to challenges associated with their massive components located in sensing, communication, processing and storage, and actuation layers [40].

The ACM Computing Surveys Journal, CSUR

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

The challenges can be due to devices' heterogeneity and the usage of many diverse programming languages, APIs, and protocols. The middleware platforms can support IoT/CPS to perform quality-driven self-adaptation. Middleware that is run on the processing and storage layer facilitates the communication between sensing and actuating components using a set of programming abstractions. Middleware facilitates the integration and communication of those heterogeneous components pervasively. IoT/CPS make middleware tasks even more challenging, since their dynamic and complex nature may require various levels of middleware distribution and collaboration. Distribution specifies whether data analysis software should be deployed on a single node or several nodes distributed across the system. The collaboration includes interaction among processing elements that satisfy the system's goal and strategy. The collaboration may appear as a level of information sharing, coordinated analysis or planning, or synchronized execution.

Although the middleware support for IoT/CPS has been investigated from more than a decade ago, the research and industry communities are still trying to define their different aspects, especially regarding their self-adaptation requirement satisfaction. The reason is that the literature is scattered across different independent research areas, such as software engineering, embedded systems, and networking. Therefore, a literature review that classifies and compares various approaches and methods for understanding *IoT/CPS self-adaptation objectives, control mechanisms* and *middleware support* is still missing. This study identifies current characteristics, challenges and publication trends, and research gaps concerning self-adaptive middleware support for IoT/CPS approach.

The significant contributions of this paper are:

- addressing to an up to date state of the art class for self-adaptive IoT/CPS and their middleware support, which can be used as a future research and implementation reference;
- presenting a self-adaptive IoT/CPS conceptual framework by focusing on the role of autonomic and functional control elements;
- classifying various middleware platforms for self-adaptation support;
- analyzing real-life use-cases that can take advantage of our systematic review results.

This study's audience is both research and industry communities interested in improving their knowledge and selecting suitable methods to design and develop their middleware for IoT/CPS.

This paper is structured as follows. Section 2 motivates the need for this study. Section 3 reveals the design of this systematic study. Section 4 presents how this study is structured, and Section 5 gives a short overview of the domain's publication trends. Sections 6, 7, and 8 elaborate on the obtained results, while Section 9 runs three horizontal analyses. The discussion by considering industrial use-cases is presented in Section 10. Section 11 analyses threats to validity, and Section 12 closes the paper and discusses future works.

2 MOTIVATION

This section discusses the motivation for handling our research and its potential scientific value. To this end, an extensive search has been carried out in Sub-section *2.1* to discover the related reviews. By comparing this research with already conducted systematic studies in the field, the current knowledge gap can be discovered. Sub-section *2.2* gives concise reasoning upon the necessity for a systematic review of self-adaptive middleware support for CPS and IoT.

2.1 Related Systematic Studies

In order to uncover previous systematic literature reviews (SLR) and systematic mapping studies (SMS) related to this research topic, we performed a search on relevant databases [1] using the following string. To include all related articles, we applied the string on title, abstract, and keywords.

[1] *ACM, IEEE Xplore, SpringerLink, Web of Science, Scopus, Wiley, and ScienceDirect.*

("mapping study" OR "literature review" OR SLR OR SMS) AND (IoT OR "Internet of Things" OR IIoT OR CPS OR "cyber-physical" OR "cyber physical" OR cyberphysical OR pervasive) AND (middleware OR "middle-ware" OR "middle ware") AND ("self-adapt" OR "self adapt*" OR "self*" OR "adapt*" OR autonomic)*

The string aims at discovering any systematic review on self-adaptive middleware support for IoT/CPS. We included all peer-reviewed systematic reviews and mapping studies that discuss any architectural, technical, or practical aspects of self-adaptive middleware support for IoT/CPS. Short articles and papers which present any IoT/CPS aspect other than self-adaptation at the middleware level were excluded.

We analyzed the search results, but we did not find any systematic study on the topic. However, six slightly related studies with different scopes have been chosen to be compared with our research. These studies were selected since they address some aspects of self-adaptation or middleware design in IoT or CPS. Table 1 shows the existing systematic studies, their focus, and the associated quality assessment (based on [19, 20]). We calculated the total score of each study [19, 34] by summing up the answer to each specific question *Q1-Q4* (Yes(Y)=1, Partly(P)=0.5, No(N)=0):

- **Q1)** Are the systematic study's inclusion and exclusion criteria described appropriately?
- **Q2)** Is the literature search likely to have covered all relevant studies?
- **Q3)** Did the authors assess the quality and validity of the included studies?
- **Q4)** Were the basic concepts and gathered data adequately described?

Table 1. Existing systematic studies on self-adaptive middleware support for IoT/CPS.

Study	*Focus*	*Year*	*Q1*	*Q2*	*Q3*	*Q4*	*Total Score*
1. Self-Adaptation for Cyber-Physical Systems:A Systematic Literature Review [43]	Architectural self-adaptation in CPS	2016	Y	Y	Y	Y	4
2. Architecting cloud-enabled systems: a systematic survey of challenges and solutions [7]	Cloud-based software systems architecture with a focus on middleware services	2016	Y	Y	Y	Y	4
3. Control-Theoretical Software Adaptation: A Systematic Literature Review [59]	control-theoretical software adaptation mechanisms	2017	Y	Y	Y	Y	4
4. Fog Computing Applications in Smart Cities: A Systematic Survey [29]	Various solutions provided by Fog computing in smart cities context	2020	Y	P	P	Y	3
5. A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things [51]	analyzing and examining load balancing remarkable methods	2019	N	P	P	Y	2
6. Service-Oriented Middleware Architectures for Cyber-Physical Systems [25]	Review on CPS middlware and presenting a conceptual middleware design	2012	P	P	P	Y	2.5

Research 1 [43] studies state-of-the-art approaches to handle self-adaptation in CPS at the architectural level. The paper follows a transparent methodology to present a reference three-layer adaptation model. The most relevant studies are included, and the results are well described. The

report analyzes the existing approaches to self-adaptation architecture in CPS to better understand state of the art and propose various solutions. While the paper considers the use of MAPE-K (Monitoring, Analysis, Planning, Execution, and Knowledge) loop for CPS self-adaptation, it does not investigate multiple interacting loops. Furthermore, the authors do not focus on analyzing novel middleware technologies. Instead, our study widens the scope to IoT and pervasive systems and proposes a set of middleware solutions for distributed systems.

Study 2 [7] respects all steps of a systematic review from inclusion/exclusion criteria to data analysis. The paper identifies 44 unique categories of challenges and associated solutions for architecting cloud-based software systems. The authors suggest that many primary studies focus on middleware services to achieve scalability, performance, response time, and efficient resource optimization. The challenge has been observed in various domains, form pervasive embedded systems and enterprise applications to smart IoT devices. While the paper addresses the use of Domain-Specific Languages in modeling secure CPS, it ignores suggesting other solutions such as service-oriented approaches. Our study characterizes device edge and fog as well, which can enhance the IoT/CPS quality.

Study 3 [59] thoroughly followed the systematic reviews' steps and protocols. This paper investigates software adaptation by modifying the software rather than the resource allocated to its execution. This paper mainly focuses on control-theoretical software adaptation and control mechanisms. The paper investigates control loops, but it ignores other IoT/CPS middleware aspects such as requirements, tools, and techniques.

Study 4 [29] follows the systematic mapping study method to obtain an overview of the existing related research literature on fog and cloud-based smart cities applications. The paper presents an analytical comparison of related works, the trends, and future research directions on Fog computing. Our study's advantage over [29] is that we present our middleware modeling solutions on top of the perceived knowledge from the reviewed literature and industrial use-cases.

Study 5 [51] respects the systematic review process, such as explaining the research questions and (partially) addressing the inclusion and exclusion criteria. The paper investigates optimizing IoT networks' usage by providing solutions for scalability, routing, reliability, security, energy conservation, network lifetime, congestion, heterogeneity, and quality of service (QoS). The authors deal with QoS issues such as latency and data packet loss using the load balancing concept by distributing loads among different routes. Our study, instead, deals with the QoS at both system and middleware architectural levels.

Study 6 [25] first present a systematic literature survey of research outputs in CPS middleware designs *i)* to present the state-of-the-art, and *ii)* to bring out some research focus on the issue. The authors further propose an early conceptual middleware designed with a service-oriented viewpoint to support CPS applications. Our study includes all architectural styles and patterns that can be useful for research and industry.

2.2 Need for an SLR on Self-adaptive IoT/CPS Middleware Support

The need for CPS modeling and development is augmented by the advent of IoT, where the relationship between physical and virtual worlds plays a fundamental role. This research complements the existing studies regarding the self-adaptation in IoT/CPS middleware support by introducing a literature-based classification of the objectives, decision methods, and tools. Although the IoT/CPS research started with concepts that appeared more than two decades ago, the research and industry communities are still progressing to define its different aspects effectively. To discover the impact of existing literature on self-adaptive IoT/CPS middleware support, we identify, describe, and classify various concepts and techniques used to engineer industry-oriented systems to help practitioners choose the best platform.

3 RESEARCH IMPLEMENTATION

This study has been carried out according to systematic reviews guidelines provided in [19, 20, 32, 33]. In this regard, we formulized our perspective by defining the purpose, issue, object, viewpoint issues ([66]).

Purpose: to provide a deep understanding of self-adaptive middleware support for IoT/CPS
Issue: by identifying, classifying, and analyzing objectives, decision methods, and tools
Object: based on existing IoT/CPS self-adaptation approaches
Viewpoint: from the research and industry viewpoints.

Such an approach comes as the primary aim of this study since there is no proper overview of self-adaptive IoT/CPS middleware support, which considers self-adaptive infrastructure and environment interaction, adaptation decision methods, and tool support with an industrial orientation. The overall process can be divided into three main phases ([33], [67]): *planning, conducting, and documenting* as thoroughly discussed in the published *Protocol*. In this paper, we only present the essential parts of the protocol to tackle the page limit.

3.1 Research Questions

To achieve the research goal, we arranged for a set of questions along with their rationale (Table 2). The classification resulting from investigating the research questions provides a solid foundation for

Table 2. Research questions and the respective rationale.

#	Questions	Sub-questions	Rationale
RQ1	What are the objectives of self-adaptation in IoT and CPS?	What changes in the environment can raise the necessity of self-adaptation?	This research question aims to identify and categorize the self-adaptation necessities due to changes in system, environment, and their coordination. These include changes in the environmental context and constraints, hardware layers, software components and connectors, and associated requirements.
		What changes in IoT/CPS HW/SW infrastructure can cause a need for self-adaptation?	
		How the dynamic coordination and interaction of IoT/CPS infrastructure and their environment can motivate self-adaptation?	
RQ2	What are the decision methods that can be adopted to realize self-adaptation in IoT/CPS?	What are the self-adaptation control times for IoT/CPS applications?	This research question focuses on IoT/CPS self-adaptation decision techniques, which imply control over IoT/CPS elements. The techniques can be categorized as model-based, rule-based, data-driven, optimization, or program-based, or a mix of them.
		What are the decision techniques that can be used as the substructure of IoT/CPS self-adaptation?	
RQ3	What kind of models, tools, or platforms are known by research and industry communities for IoT/CPS self-adaptation support?	Are the focus of self-adaptation supports on language or middleware levels, or domain-specific applications?	This question deals with various supports for IoT/CPS design and development. The subject attempts to discover existing platforms and applications, their features, and their requirements. The focus of this paper is especially on middleware support. Thus, the classified state of the art knowledge shall result in a set of middleware patterns potentially suitable for various domains.
		How can the self-adaptation support platforms satisfy industrial needs?	
		What range of application domains is addressed by each platform?	

a thorough identification and comparison of existing and future self-adaptive middleware solutions for IoT/CPS. This contribution is useful for researchers and practitioners who are willing to further contribute to new IoT/CPS modeling and development approaches or better understand or refine existing practices. The research questions listed in Table 2 will drive the whole systematic review methodology, with a notable influence on the primary studies search, the data extraction, and the data analysis processes.

It is worth mentioning that a good search strategy is expected to provide practical solutions to the following questions: *which, where, what, and when* [69].

Which approaches? The search strategy consists of two phases: *i)* automatic search in scientific databases; and *ii)* snowballing. The first step is performed using a search string (see below) based on identified keywords from research questions and areas of study. The search strings are used to retrieve potential primary studies through web search engines provided by digital libraries. Snowballing refers to using the reference list of a paper (backward snowballing) or the citations to the paper (forward snowballing) to identify additional papers [66]. The start set for the snowballing procedure is composed of the selected papers retrieved by the automatic search, namely the primary studies, which are selected by applying inclusion/exclusion criteria to the automatic search results. In any case, the inclusion/exclusion criteria are applied to each paper. If an article is considered to be included, snowballing is applied iteratively, and the procedure ends when no new papers can be found.

(IoT OR "Internet of Things" OR IIoT OR CPS OR "cyber-physical" OR "cyber physical" OR cyberphysical OR pervasive) AND (middleware OR "middle-ware" OR "middle ware") AND ("self-adapt" OR "self adapt*" OR "self*" OR "adapt*" OR autonomic)*

Where to search? According to [69], it is essential to search for many different electronic sources because no single source can find all relevant primary studies. We followed the same procedure used for other systematic studies, such as [39, 42]. Table 3 shows the electronic databases that we used for the automatic search as the primary source of literature for potentially relevant studies on the domain.

Table 3. Electronic data sources targeted with search strings.

Library	Website
IEEE Xplore Digital Library	https://ieeexplore.ieee.org
ACM Digital Library	https://dl.acm.org
SpringerLink	https://link.springer.com
Web of Science	http://apps.webofknowledge.com
Wiley	http:/onlinelibrary.wiley.com
ScienceDirect	http://www.sciencedirect.com
Scopus	https://www.scopus.com

What to search? A suitable search string is the input to the electronic data sources identified in the previous section, matching with paper titles, abstracts, and keywords. Following some test executions and refinements, the search string has been finalized, as shown above. We tried to codify the string to conform to each selected electronic data source's specific syntax and criteria. Further, we combined all studies into a single dataset after the removal of impurities and duplicates.

When and what period to search? We do not consider publication year as a criterion for the search and selection steps. Thus, all studies coming from the selection steps, until June 2020, is included regardless of their publication time.

3.1.1 Selection Strategy. A multi-stage selection process (Figure 1) has been designed to give full control of the number and characteristics of the studies coming from different stages [2]. As shown in

[2]It is worth mentioning that on Springer, we considered "computer science" as the sub-discipline, and on Science-Direct, we searched titles and abstracts only. These were to avoid a considerable number of false positives results.

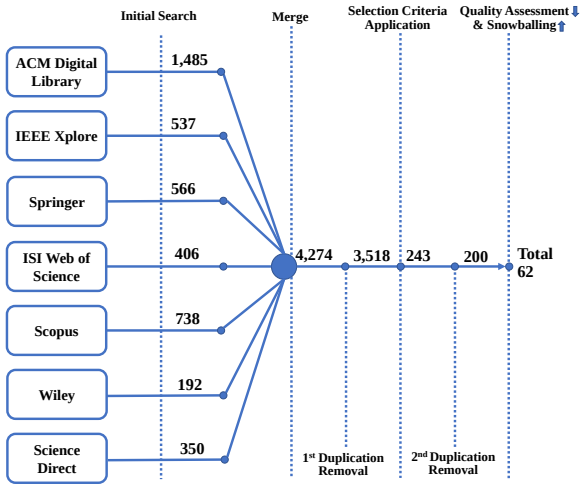

Fig. 1. Search and selection process.

Figure 1, we first applied the automatic search using the previously defined string on the electronic databases. This step resulted in 4,274 papers, which, after duplication removal, were reduced to 3,518. Researchers independently read the abstract of all studies selected and used the inclusion and exclusion criteria (Table 4) to filter out irrelevant papers. A paper was included only when it satisfied all inclusion criteria and did not satisfy any exclusion criteria. The included papers of each researcher were checked by the others to minimize the bias.

Table 4. Inclusion and exclusion criteria.

Inclusion criteria	Exclusion criteria
Studies that propose modeling and/or analysis and/or development solution, architecture, method, and/or technique, specific for engineering self-adaptive middleware support for IoT/CPS.	Studies that, while focusing on IoT/CPS, do not explicitly deal with their self-adaptive middleware modeling and/or development aspects (e.g., studies focusing only on technological aspects and inner details of IoT/CPS).
Studies subject to peer review (e.g., journal papers, papers published as part of conference proceedings, workshop papers, and book chapters).	Secondary or tertiary studies (e.g., systematic literature reviews and surveys).
Studies written in the English language and available in full-text.	Studies in the form of tutorial papers or editorials. Because they do not provide enough information.

Applying the selection criteria led us to 243 studies. Although all the selected studies were on-topic, all three of us evaluated them qualitatively. The following quality assessment criteria were considered:

- **QA1)** What are the applicability and popularity of the research?
- **QA2)** Does the research contain novel and up-to-date methods and solutions?
- **QA3)** How can the research help design self-adaptive middleware solutions and patterns?
- **QA4)** Is the contribution well established and explained?
- **QA5)** Is the approach well evaluated?

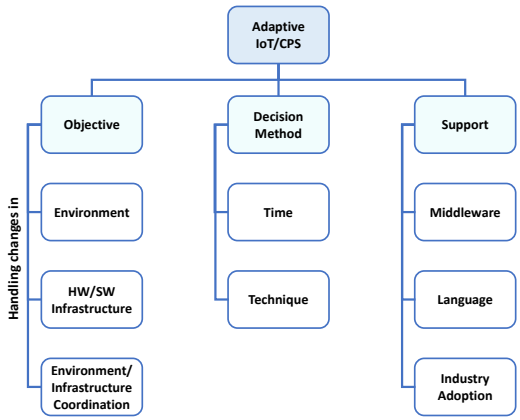

Fig. 2. Taxonomy on the covered areas of self-adaptive IoT/CPS.

The first quality assessment question evaluates if the method presented by a study is widely applied to other research or industrial cases. The second question rates the studies on the novelty of their problem-solving processes. This point can compensate for the lack of applicability required by the first question. The third question assesses the studies' system architecture to see how it can support proposing middleware solutions and design patterns. The fourth question analyzes the appropriateness of contribution, and the fifth question looks into the evaluation presented by each study. We calculated each study's total score by summing up the answer to each specific question $Q1 - Q4$ (Yes=1, Partly=0.5, No=0).

The quality assessment phase resulted in 59 studies, which increased to 62 by applying the snowballing process explained in the previous subsection. Among the reasons for which the snowballing added only a few primary studies, we bring up the effort we dedicated to design an inclusive search string and a careful selection that included almost all significant studies on the topic. After selecting a final set of primary studies, the data has been extracted to answer the research questions.

3.2 Protocol and Replication Package

In order to focus on the result of our systematic studies, we provide the following external documents:

- A peer-reviewed protocol that clarifies the review process. The document is published on HAL-INRIA: *Protocol*. The protocol includes a detailed explanation on search and selection strategies, external review, and documenting.
- A replication package that is provided to tackle the page limits of a workshop paper: *ReplicationPackage*. The package is available as an excel file with different sheets that include all necessary information such as search results, primary studies distribution, data extraction, validity examination, and quality assessment.

4 STRUCTURING THE STUDY RESULTS

By analyzing the primary studies, a set of representative concepts have been identified, as shown in Figure 2. The taxonomy shows various self-adaptation concerns on IoT/CPS. This study's focus goes to three main aspects of self-adaptation, namely *objectives, decision methods,* and *supporting tools*. As shown in Table 5, the extracted data has been clustered and classified into data items.

Table 5. Collected data items.

Data Item	Data Field	Research Question
DI1	Authors	Documentation
DI2	Year	Documentation
DI3	Title	Documentation
DI4	Venue	Documentation
DI5	Publication Trends	Documentation
DI6	IoT/CPS environmental context and constriants	RQ1
DI7	IoT/CPS hardware system	RQ1
DI8	IoT/CPS software components and connectors	RQ1
DI9	System goals, functional and non-functional requirements	RQ1
DI10	Coordination among CPS/IoT infrastructure and the environment	RQ1
DI11	Proactive and reactive adaptation	RQ2
DI12	Adaptation control and decision models	RQ2
DI13	Middleware support for IoT/CPS	RQ3
DI14	Application domain	RQ3
DI15	Industry adoption	RQ3

Data items [59] facilitate answering the identified research questions. The subsequent sections discuss each specified data field by potentially dividing them into subcategories. We here give a short clarification on each set of data items:

- *DI1-DI5.* These data items are used for documentation and trends. To realize the publication trends, we provide the distribution of publications and their types by year and venues.
- *DI6. IoT/CPS environment issues.* It includes the environmental context, constraints, and limitations that might affect the system adaptation objectives.
- *DI7. IoT/CPS Hardware system.* Extracted data are divided into four subcategories: sensors, network facilities, computing resources, and actuators. The IoT/CPS might be pushed to self-adaptation because of changes in its HW system.
- *DI8 IoT/CPS Software system.* Changes in software that is run on hardware can cause an adaptation need. From software viewpoint, elements are categorized into *components* and *connectors*. The software components are *monitoring SW, autonomic control, functional control,* and *execution SW*. The two mentioned sets of control elements are considered to divide self-adaptation concerns caused by the system and the environment.
- *DI9. The functional and non-functional requirements.* Violating the system requirements can lead the system to self-adaptation. Furthermore, some qualities can be affected by the self-adaptation process. We initially used the specification of qualities described in the *ISO/IEC 9126-1standard* [5], while focusing on the attributes that our primary studies brought up.
- *DI10. Coordination among CPS/IoT infrastructure and its surrounding environment.* The initial options for managing such coordination are *embedded systems, control of physical systems*, and *IoT/CPS distributed systems*.
- *DI11. Self-adaptation time.* This answers the question of *when the self-adaptation should take place.* In *reactive*, self-adaptation takes place after the event that causes the need for adaptation, while in *proactive*, the system identifies the need for self-adaptation before the undesired event happens [53].
- *DI12. Self-adaptation control and decision technique.* This data item deals with the method by which the self-adaptation should be performed. The decision method can be set based on various domains and fashions, such as *model-based* (e.g., model-predictive control, model-driven engineering, agent-based modeling, and architecture reconfiguration), *rule-based* (e.g., event-based, and reconfiguration rules), *data-driven* (e.g., machine learning, and reinforcement learning), *optimization-based* (e.g., cross-entropy), and *program-based*.

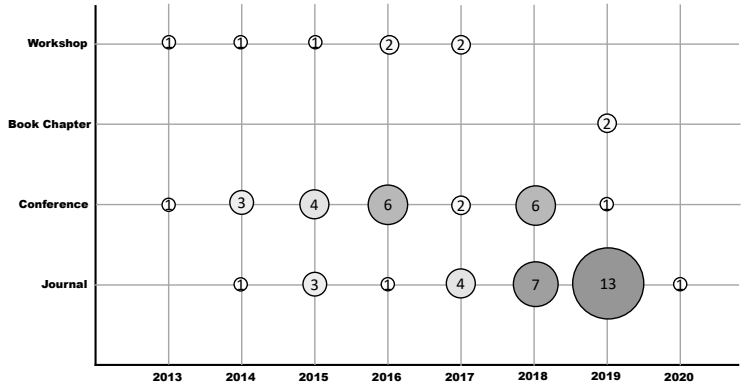

Fig. 3. Primary studies distribution by publication type.

- *DI13. Middleware support.* The support to implement self-adaptive IoT/CPS could be in *language, middleware levels,* or *specific domain requirements.*
- *DI14. Application Domain.* Each middleware will be linked to *application domains* for which they are suitable.
- *DI15. Industry adoption.* This item investigates if the middleware is widely used in the industrial context or not. A discussion on potential industrial adoption will be further provided.

5 DOCUMENTATION AND TRENDS

We extract *authors, publication year, title, type,* and *venue* of the chosen 62 primary studies. Figure 3 shows the distribution of self-adaptive IoT/CPS middleware support literature. It noticeably indicates that the number of papers grows over time, and 90% of articles are published within the last five years. This result confirms the recent scientific interest and research necessity on self-adaptive IoT/CPS middleware issues.

The most common publication type is journal paper (30/62), followed by conference (23/62), workshop (7/62), and book chapter (2/62). Such a high number of journal and conference papers may point out that self-adaptive IoT/CPS middleware support is maturing as a research topic despite it is relatively young. Furthermore, we noticed that research on self-adaptive IoT/CPS middleware support is spread across many venues, mostly in the span of IoT (e.g., WF-IoT and IoTDI), control (e.g., CCTA), networking (e.g., NOMS), and computing (e.g., SOCA). The complete list of venues can be found in the data extraction file. The focus on the aspects mentioned above can prove the significance of distributed control and networking for self-adaptive IoT/CPS middleware design.

6 SELF-ADAPTATION OBJECTIVES (RQ1)

An IoT/CPS should be adapted due to the changes in the *environment*, the *infrastructure*, and/or their *coordination*. In the following subsections, we thoroughly analyze the elements which can motivate self-adaptation.

6.1 Environment

An IoT/CPS is situated in the environment. The environment is the real world by which the system might interact. The environment might include both physical and virtual elements [62], that the system does not directly control their functionality. The system can perform regardless of changes

in the environment. However, most adaptive IoT/CPS systems interact with their environment in an ever-changing manner.

Based on our literature review, most of the primary studies mention that environmental contextual aspects and the associated constraints and limitations impact the adaptation requirements (*P31*) [8]. Several contextual dimensions can be considered to define the IoT/CPS surrounding environment:

- *The physical context:* it relates to information relating to the physical environment, such as geographic location, temperature, humidity, noise, and light.
- *The temporal context:* it concerns time-related information that can affect an IoT/CPS.
- *The social context:* it concerns the direct or indirect interaction of a system with people or objects located in the physical or virtual environment.
- *The computational context:* it is related to the external resources available for the system, such as computing resources, communication bandwidth, and storage resources.
- *The historical context:* it deals with historical data that can affect the interpretation of information or the system operation.
- *The profile:* it concerns an entity's preferences for the different contextual dimensions.

A deep understanding of the context is essential for choosing or designing the right IoT/CPS infrastructure. The lack of a uniform approach for capturing information associated with the context makes it difficult to fully understand the context model's needs and design approach based on its main characteristics. In dynamic IoT/CPS, handling context-aware adaptation by capturing and analyzing the context within the evolutive environment is complex. A reason is that the IoT/CPS elements which capture and effect the environment are heterogeneous, regarding their manufacturers, operating systems, device types, and communication protocols (*P62*) [50]. Another issue regarding contextual information is that they are often incomplete, temporal, and interrelated [56]. Thus, contextual reasoning mechanisms should derive a high level, inferred context from low-level raw contextual information. A reasoning mechanism that provides knowledge for self-adaptation decision making should also transform, unify, and verify inconsistent contextual knowledge from imperfect and faulty input.

Although the environmental contextual solutions provided by the primary studies are usually designed for particular cases by using different approaches, their common characteristics are:

- The context is subdivided into dimensions or attributes that identify the relevant operating elements of the IoT/CPS, such as time, position, and temperature.
- Measuring various dimensions of the environment should be possible by defining quantities, units of measurement, and range of permissible values.

6.2 Infrastructure

The IoT/CPS infrastructure consists of various HW and SW elements. Figure 5 shows the adaptive IoT/CPS infrastructure components. The architecture component includes both HW and SW, which will be following explained.

6.2.1 Hardware. IoT/CPS *hardware* architecture can be re-structured in run-time to add, delete, replace, and combine its elements. The IoT/CPS HW elements include *sensors, network facilities, computation resources,* and *actuators.* IoT/CPS sensors and actuators directly connect with physical space to gather information and affect the environment. Network facilities can highly impact the quality of service (QoS) provided by the system, e.g., performance and resilience.

The computation resources can be located on the device, at the network edge, or cloud. Edge devices are getting increasingly powerful in terms of their hardware specifications. They have advanced beyond the simplistic notion of collecting and transferring raw data and nowadays act as fully functional processing units in their own right, widely used as effective computing systems.

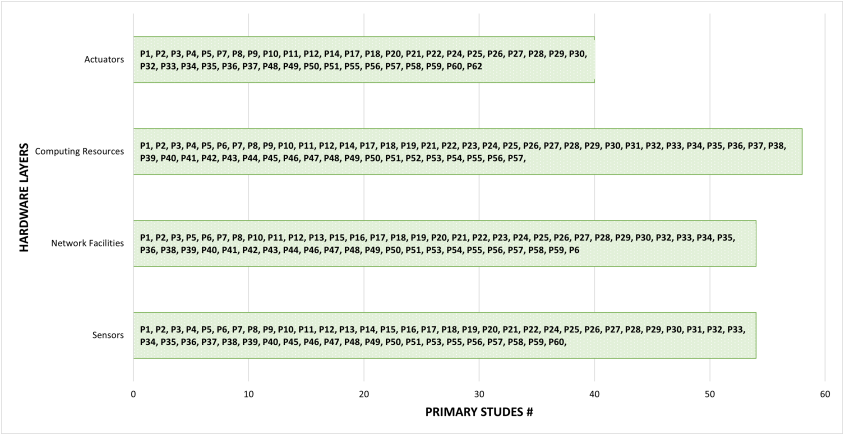

Fig. 4. Self-adaptive Systems Hardware Layers.

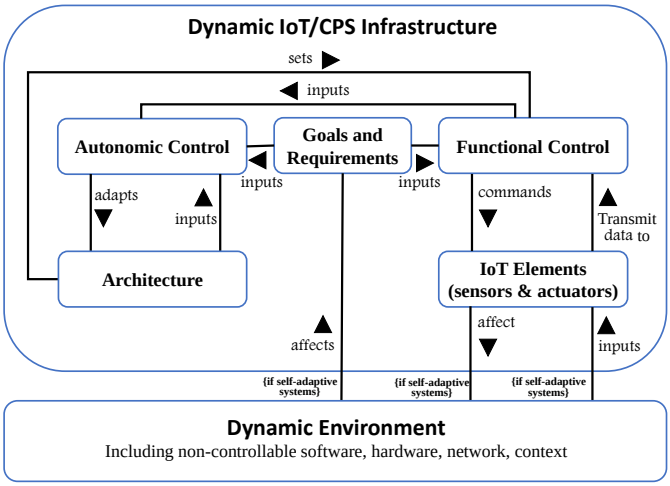

Fig. 5. IoT/CPS environment, infrastructure, and their coordination.

Pushing the powerful computation resource to the edge can be considered as a way to improve QoS. Since our literature review focused on middleware aspects of self-adaptive IoT, it is apparent from Figure 4 that most of the primary studies deal with computing resources (58/62) and network (54/62). The middleware is traditionally run on computing resources by taking advantage of network facilities to manage heterogeneous IoT elements. Thus, sensors (54/62) have received equal concern, while actuators (40/62) were not in the middle of attention across primary studies.

6.2.2 Software. The IoT/CPS *software* that is run on hardware elements includes a set of components bounded by connectors based on specific rules and constraints. Figure 5 shows a model of IoT/CPS systems that might be in interaction with the dynamic surrounding environment. To realize such interaction, the system takes advantage of *sense and actuate elements*. The *sense* elements frequently retrieve raw data [1] to input the control components, and *actuate* elements receive periodic commands to affect the environment. The mentioned data transmission is continuous

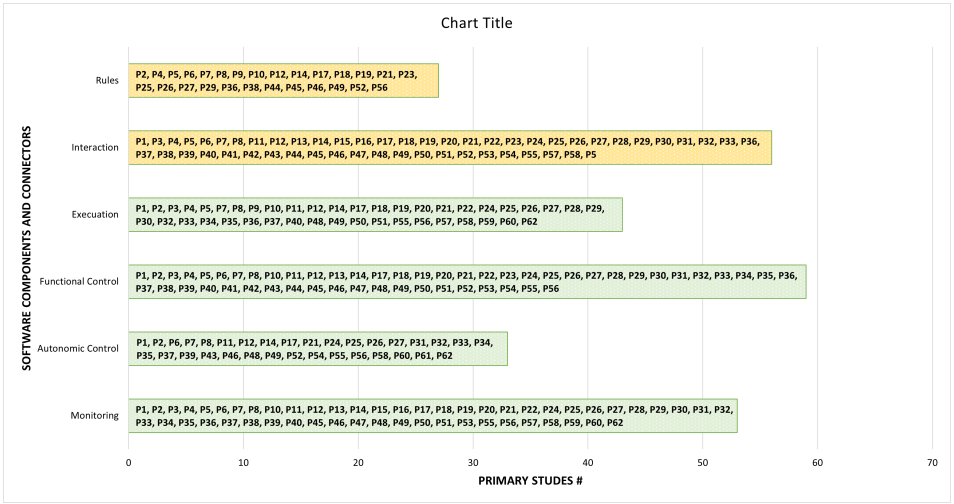

Fig. 6. Adaptive systems software layers.

since the environment is not under full control of the software system, and the dynamics of the environment should be tackled.

The *functional control* comprises the adaptation logic that allows the system to perform the intended adaptation within the environment. The *autonomic control* supports a continuous adaptation process [54]. It enables the system to monitor itself continuously and perform necessary adaptation to achieve the adaptation goals. Both control elements take input from *goal and requirement* [27] component. A system contains both *functional* and *adaptation goals* that are set by stakeholders. Functional goals specify the system's functionality under various environmental constraints, and adaptation goals mostly concern the system's quality. As shown in Figure 5, the goals might be affected by the *environment*. In other words, the environment context might enforce prioritizing a set of goals or ignoring another set of goals.

Architecture component shown in Figure 5 determines variations in both software and hardware architectures [42], which include their already explained elements. These architectures are designed by stakeholders and self-adapted by the autonomic control element during system execution [13]. It is worth mentioning that, architecture variations might determine multiple functional deployment types, which appear as architectural patterns shown in Figure 7. The patterns are composed of *IoT elements* layer and one or several *functional control* layers. The functional control can perform locally and/or centrally and remotely. Here is the point in which a centralized cloud and distributed edge and fog can form the *hierarchical* pattern. Thus, the patterns [39] characterize *IoT* systems based on their levels of *distribution* and *collaboration* [39] [42]. Distribution specifies whether data analysis software ought to be deployed on a single node (*centralized*) or on several nodes (*distributed* and *hierarchical)* that are dispersed across the *IoT* system. The collaboration deals with interaction among functional control components to satisfy the goals, requirements, and strategies. This collaboration may appear as a level of information sharing, coordinated analysis or planning, or synchronized execution [44].

As shown in Figure 7, The *centralized* pattern comprises processing on a *central local* or *remote* controller. The *distributed* pattern includes the processing on *independent* or *collaborative* controllers. The *Hierarchical* pattern contains *independent* or *hybrid* (i.e., with distributed collaborative)

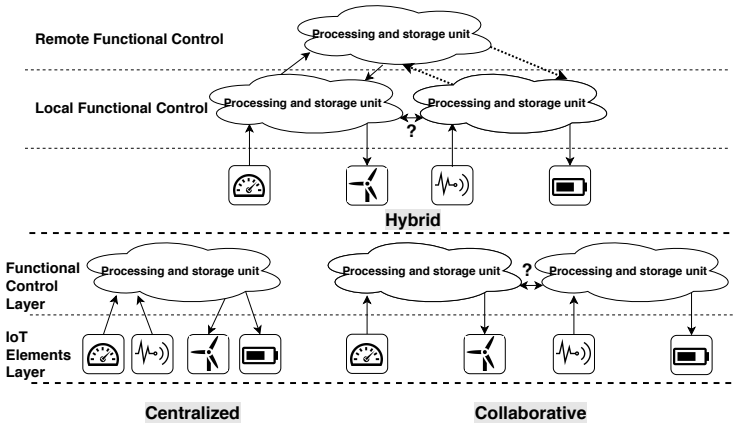

Fig. 7. *IoT* architectural patterns based on functional control components composition.

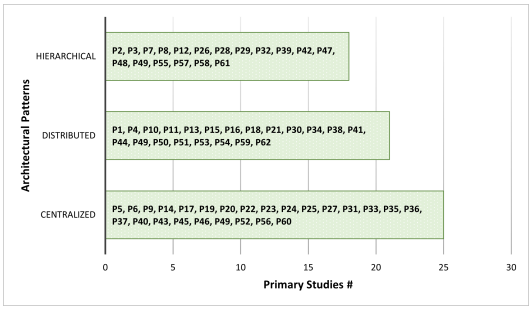

Fig. 8. Self-adaptive systems distribution levels.

controllers. As shown in Figure 8, most of the studies (25/62) follow a centralized distribution pattern, while the distributed (21/62) and hierarchical (18/62) patterns are widely addressed as well.

The self-adaptive IoT/CPS infrastructure explained above contains the mechanisms to determine the required adaption, based on intended *QoS* satisfaction level. The next subsection deals with those quality attributes.

6.2.3 Non-functional Requirements. As mentioned above, the IoT/CPS requirements are both *functional* and *non-functional*, and can necessitate changes in architecture. As shown in Figure 9, among non-functional requirements, the most recognized quality challenges at both system and middleware levels are performance (33/62), interoperability (28/62), scalability (17/62), adaptability (17/62), availability (16/62). Meanwhile, security (15/62), reliability (11/62), and dependability (8/62) are positioned in a lower degree of concern. Several quality attributes are linked to the main middleware's target: supporting large-scale, heterogeneous, and distributed architectures.

- *Performance.* deals with the system's response to performing certain actions for a certain period. The performance level depends on how much the processing and storage components are pushed to the edge in a decentralized way. The processing location and distribution impacts the real-time requirement satisfaction on the system as well.
- *Interoperability.* deals with the system and subsystems collaborative operation as well as the data transmission with external entities. Interoperability helps IoT/CPS heterogeneous

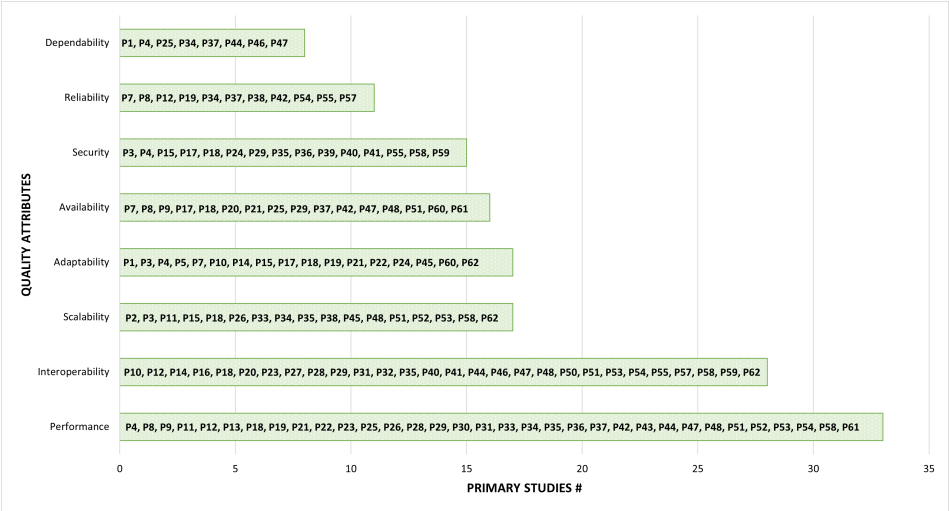

Fig. 9. Self-adaptive systems' notable quality attributes.

components to work together efficiently. It depends on how much IoT/CPS large-scale heterogeneous devices can communicate directly to gather the required data without going through the central component.

- *Scalability.* is the ability of the system to handle load increases without decreasing performance, or the possibility to increase the load rapidly. Scalability is an essential attribute since CPS should perform at an acceptable level with many devices. Scalability depends on how new resources can be added on demand.
- *Adaptability.* is the ability of the system to adapt to its own and its environmental situation. The adaptation that is a kind of system flexibility lets the system structural or algorithmic variation potentially enhance other attributes. Such an ability to evolve concerns the adaptability of IoT/CPS to new technologies and applications. This can be realized by system openness to change and extension.
- *Availability.* is the ratio of the available system time to the total working time. Availability is the ability of a system to be wholly or partly operational, as and when required. In IoT/CPS, availability is associated with the processing and storage location and processing power. For instance, the cloud provides high availability of computing resources at relatively high power consumption, while fog provides moderate availability of computing resources at lower power consumption [28].
- *Security.* concerns the system's ability to reduce the likelihood of malicious or accidental actions as well as the possibility of theft or loss of information. Designing a robust, secure, and efficient IoT/CPS is another issue that necessitates developing novel methods to protect data against undesired consequences such as cyber-attacks is crucial.
- *Reliability.* is an attribute of the system responsible for the ability to continue to operate under predefined conditions. A reliable system can fulfill its task in a given environment, assuming that the hardware is fault-free and input cases are predefined.

- *Dependability.* Dependability and availability are strongly related to each other and against system failure due to components failure. The IoT devices' massive scale makes the dependability concern more critical due to external or internal processes that might cause cyber and physical deterioration.

6.3 Coordination of Infrastructure and Environment

The coordination of IoT/CPS infrastructure and the environment can be in different manners:

- Dynamic environment with static computation infrastructure, as in, e.g., classical hard real-time systems;
- Dynamic computation infrastructure that does not consider the potential changes in the environment, as in, e.g., off-line applications like simulation;
- Dynamic computation infrastructure in coordination with the dynamic environment, in more general IoT/CPS cases.

There are various methods to design the IoT/CPS infrastructure and environment coordination. We observed that the coordination is carefully analyzed and addressed mostly by distributed embedded IoT/CPS devices that are distributed across the system. As shown in Figure 5, the environment includes software, hardware, network, and context that are not under full control of a specific IoT/CPS. However, the system can affect the environment using embedded distributed sensors and actuators. Embedded systems with specific computing capacity and low energy consumption have the necessity of being adapted to the contextual situations. Such adaptation need might be occurred due to various requirements such as changes in the embedded control system's goal, changes in the surrounding environment, or changes in the control system itself (e.g., fault recovery).

Answer to RQ1:
The self-adaptation can occur due to changes required in the system, its surrounding environment, and their coordination. According to our literature-based proposed approach, the system's architectural changes are handled by autonomic control, while the functional control manages the changes related to the environmental context. Our study revealed that while most of the primary studies focus on the adaptation aspects of functional control, the autonomic control topic is recently getting more attention.

7 SELF-ADAPTATION DECISION (RQ2)

7.1 Time

The *time* aspect of IoT/CPS adaptation is related to when the adaptation should take place. The adaptation decision can follow a *proactive* or *reactive* strategy. If the IoT/CPS performs adaptation when a goal or requirement is already violated (e.g., a change in the resources or a drop in performance), it is reactive. If it adapts because of predicting any missed goals or requirements in the future, it is performing proactively. Users prefer proactive adaptation because of its ability to avoid quality degradation within the system. However, the proactive feature requires running complex prediction algorithms that depend on the correctness of entry data as parameters. Thus, a significant part of our primary studies (59/62) focuses on reactive adaptation. In fact, the monitoring and execution activities are very much the same in reactive and proactive methods, but the analysis and planning phases make the difference. Following, we quote examples from the few studies which address the proactive approach (7/62). It is worth mentioning that some studies (*P4, P20, P25, P37*, and *P46*) use both proactive and reactive decision time.

- *P4.* [22] uses machine learning for architectural reconfiguration. They take advantage of ThingsJS middleware's scheduler to predict the execution time of each component. Afterward, a solver determines the best configuration of components to minimize the execution time while respecting the constraints. More specifically, each component periodically reports its performance metrics to the middleware database. The execution time of each component is predicted using various models. It is worth mentioning that such a prediction process adopts a feedback loop to be efficiently iterative. By using the mentioned technique, the run-time self-adaptation can be performed by, e.g., migrating already-executing components to different devices.
- *P6.* [6] uses a live learning process for configuring sensors' sampling rate and optimizing network usage. The lifetime of battery-powered IoT/CPS devices can be extended by activating deep-sleep mode. Machine learning algorithms can optimize the deep-sleep periods by predicting current or future data traffic values without requiring any sensor network call. The used approach can facilitate designing a self-adaptive platform that provides both energy efficiency and data accuracy.
- *P13.* [68] designs a pervasive system in which a self-adaptive temperature control system can predict the users' arrival time to a place based on their historical and real-time location data. Middleware is used to provide interoperability among heterogeneous devices from different suppliers. All physical devices and external systems have a corresponding virtual entity in the middleware's entities pool. When the sensor virtual entity's status perceives a new temperature data passing the filter, the message subscription and pushing module will push the subscribers' data. The prediction system uses the users' backing time together with real-time indoor temperature to determine whether the air conditioning system should be turned on or not.
- *P37.* [14] equipped their system with predictive modeling and on-line learning abilities to provide self-modeling abilities in self-aware software on-chip paradigm. They use statistical and neural network approaches such that the model accuracy can be traded-off for computational model complexity. They use regression-based linear predictors and nonlinear neural predictors to build models of the system performance, power, and energy consumption using the cross-layer events, hardware counters, and on-chip sensor data. Such predictive capabilities can improve autonomy in managing the system resources and assisting proactive resource utilization in the run-time system.

7.2 Decision Methods

The adaptation control elements need to adopt decision techniques to handle the self-adaptation. Literature addresses such techniques as *model-based, rule-based, data-driven, optimization-based,* and *program-based.* It is worth mentioning that some approaches used by primary studies fall into multiple categories of decision methods. The self-adaptation techniques should be chosen based on the IoT/CPS characteristics such as available data types, functional and non-functional goals, and adaptation time.

7.2.1 Model-based. In model-based methods, the models which represent the actual system's characteristics can provide adaptation solutions. Within *model-driven engineering* domain, models are the first entities to describe the software and its environment [18, 55]. Those models could be advantageous for both design-time and runtime applications. Within the model-based approaches, the *architecture* and its reconfiguration techniques can be modeled and analyzed. Several studies propose the use of software architectures to address IoT/CPS self-adaptation (30/62). An architecture model provides a global view of the system and its properties and behavior [21]. While architectures

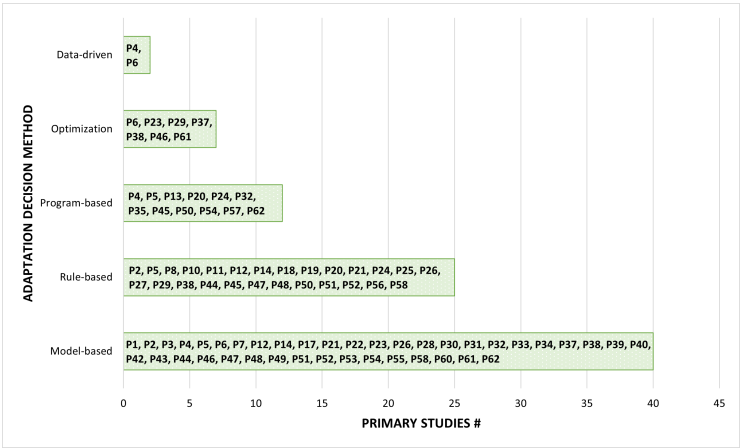

Fig. 10. Self-adaptive systems decision methods.

give a global idea of the system, IoT/CPS software systems' heterogeneity makes it challenging to design a set of self-adaptation architectural patterns for reconfiguration. Some studies argue that architectural adaptation includes an architectural model of the controllable software components that allows the feedback loop to reason about various system configurations and adapt it based on goals [64].

Feedback loops, which are another sub-category of model-based adaptation decision approaches, are widely used by the primary studies (17/62). Control loops are introduced to facilitate self-adaptation by handling changes and uncertainties. IoT/CPS sensors supply raw data (*M*) to central or distributed computational components to be refined and analyzed (*A*) towards further actuation planning (*P*) and execution (*E*). This process within comprehensive knowledge (*K*) forms the *MAPE-K* control loop. Each element of the MAPE-K loop should dynamically react to changes in the system's goals and requirements. Works on using feedback control loops (such as *MAPE-K*) and their interaction can be presented as patterns [65], in which the functions from multiple loops are coordinated in different ways. Such interactive coordination mechanisms are indeed crucial to model ever-growing distributed IoT/CPS systems.

In the agent-based self-adaptation approach, each agent is an autonomous problem-solver able to operate in dynamic environments. Some agent-based modeling features make it suitable for engineering self-adaptive systems, namely loose coupling (since they are self-contained goal-directed), context-sensitivity (since they include a specification of the context), and robustness to failures (since failures cause reposting the goal without the need for usual complexity of process failures handling) [11, 63].

Mathematical models of dynamic IoT/CPS can facilitate analyzing the effect of changes on the system and triggering adaptation reconfiguration [17]. As such, model checking usually use probabilistic algorithms for optimal adaptation decision. It is worth mentioning that model checking techniques are known to be computationally expensive, since by increasing the number of state variables, the size of the system state space grows exponentially [26]. Another instance of mathematical adaptation that is linked with feedback control loops is control theory. Control theory proposes a systematic way to design feedback control loops to handle unpredictable changes at runtime for software applications [16].

7.2.2 Rule-based. In rule-based approaches, self-adaptation is performed for specific events and under determined conditions. In other words, rule-based adaptation has a *WHEN (event) IF (condition) THEN (action)* form in which events trigger the rules which assess some conditions to act. Rule-based adaptation has some advantages, such as the readability of adaptation rules and highly-efficient decision-making for adaptation. It also suffers from some disadvantages, such as lack of guarantee for optimal or nearly-optimal adaptation results, and weak support to cope with a dynamic environment and runtime goals [23, 70]. Thus, rule-based adaptation generally supports design-time adaptation, suitable for static IoT/CPS, with low flexibility regarding system and environment changes.

7.2.3 Data-driven. IoT/CPS sensors gather a massive amount of data that can facilitate proactive adaptation. A data-driven approach can feed the system with real-time sensory data and prediction models to draw a system's behavior and its surrounding environment. Data-driven methods usually take advantage of Machine learning (ML) techniques. ML is a set of principles, algorithms, and techniques rooted in statistics to let the systems automatically learn, improve, and perform tasks without being explicitly programmed, but instead based on the data that it is fed to. ML offers various approaches, such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Using data-driven approaches can advantage the IoT/CPS quality of service (QoS) enhancement. ML techniques that can predict QoS degradation and select suitable preventive strategies are applicable to perform QoS-driven adaptations. Machine learning can be integrated with the feedback loop approach in analysis and planning steps. ML can help analyze data to select a set of proper adaptation strategies to be assessed and optimally enforced by the feedback loop.

7.2.4 Optimization. Some studies utilize optimization algorithms to guarantee optimized functionality and/or quality of IoT/CPS. *Self-optimized* systems aim to adapt to changes that may occur in their operational contexts, environments, and system requirements in an optimized manner [30]. Self-optimization through runtime adaptation guarantees not only the functionality but also an optimal trade-off among QoS requirements. Thus, the optimization objective seeks ways to, e.g., improve performance while keeping the reliability and availability at a suitable level. The optimization algorithm can perform as the core of IoT/CPS software architecture, where functional and autonomic controllers impact the environment and system situation.

7.2.5 Program-based. The IoT/CPS self-adaptation decision can be made at the program level. As such, code migration is the process of redeploying software code among hardware resources. It is actually the activity associated with promoting new and modified code, configuration, and scripts to support the adaptive system. Another example of program-based adaptation is a protocol adaptation that is driven and controlled by a specific set of adaptation policies.

Answer to RQ2: The time of adaptation (reactive or proactive) has an undeniable impact on choosing adaptation decision methods. There are various methods to adopt based on the IoT/CPS characteristics and adaptation needs, namely model-based, rule-based, data-driven, optimization, and program-based. Most of the primary studies used reactive model-based methods, while the community is being oriented to data-driven proactive adaptation approaches.

8 MIDDLEWARE SUPPORT FOR SELF-ADAPTIVE IOT/CPS (RQ3)

The *support* to implement self-adaptive IoT/CPS could be in *language, middleware,* or *specific domain requirements*. However, the focus of this study is on the middleware approach.

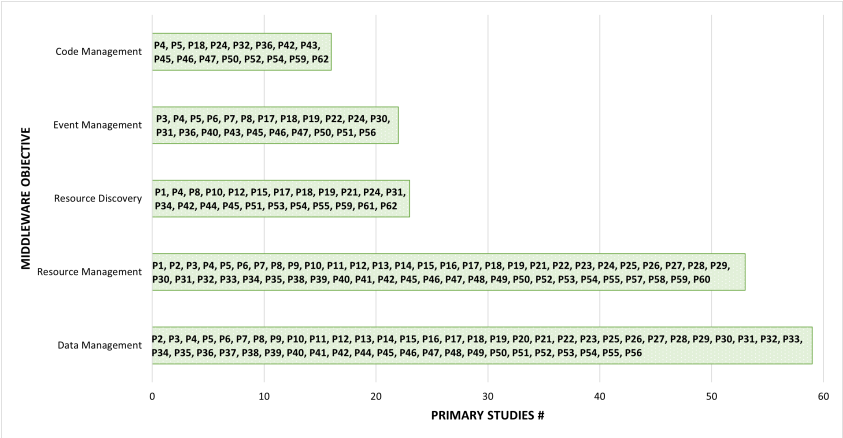

Fig. 11. Adaptive systems' middleware objectives.

8.1 Middleware Goals

Middleware platforms follow various objectives to provide functional and non-functional requirements satisfaction. The middleware non-functional requirements are explained before. Regarding the functional design, a middleware-based IoT/CPS may address one or several [52] following goals.

- *Resource Discovery.* IoT/CPS distributed resources, including hardware devices and software components, are not always available and interoperable. This concept is highlighted when the system deals with massive mobile nodes which should frequently be added or removed. Thus, the resource discovery that needs to be automated based on adaptation techniques should also properly scale, and there should be an efficient distribution of discovery load.
- *Resource Management.* A proper level of quality is expected for all IoT/CPS applications. It is also important that applications are provided with service managers. In architecture design for self-adaptive IoT/CPS, middleware should facilitate automatic resource composition to satisfy the goals and requirements.
- *Data Management.* data is a crucial element in IoT/CPS applications that is sensed, propagated, transmitted, processed, and stored through the system.Data coming from heterogeneous devices potentially have various types and frequencies. To provide applications that get input from those devices, middleware should be equipped with fusion and orchestration techniques.
- *Event Management.* A massive number of events generally generate in IoT/CPS applications. Middleware is responsible for managing and integrating those events. The event management that is a part of the analysis process transforms observed events into meaningful events.
- *Code Management.* Deploying code in IoT/CPS environments is complex and should be supported by middleware. Code allocation (to select the set of devices to be used to accomplish a user or application level task) and code migration (transferring one device's code to another) services are specially required.

Figure 11 shows the functional objectives adopted by the primary studies. Data management (59/62) was addressed as the primary concern of middleware design for IoT/CPS, followed by resource management (53/62). This result shows the challenge behind managing and harmonizing big data coming from heterogeneous resources. Discovering resources (23/62) is another challenge that is widely addressed by primary studies. Event management (22/62) and code management (16/62) has got relatively lower attention as functional objectives of middleware design.

8.2 Middleware Solutions

Middleware provides common services for applications, eases their development, and makes the heterogeneous IoT/CPS components transparent for the application layer. In self-adaptive IoT/CPS domain, many solutions with various design approaches have been used. We classified [52] the primary studies based on six middleware design categories:

- *Event-based.* Some middleware platforms provide *event-based* solution, in which all IoT/CPS elements, including middleware, interact by events. Events that have a state are propagated from producers to consumers. The most recognized pattern which falls under event-based middleware is publish/subscribe. In this method, subscribers can access the data stream (events) coming from publishers through a database. In our literature review, (11/62) studies use the publish/subscribe concept: *P2, P4, P5, P20, P24, P25, P38, P44, P47, P51, p58.*
- *Service-oriented.* Some middleware platforms use *service-oriented* approach to provide the applications as service. The service-oriented approach for massive IoT/CPS may be exposed to heterogeneity, resource-constrained, and mobility issues. However, it supports adaptive service composition in the case of unavailable services. We observed several primary studies that use service-oriented approach: *P3, P15, P16, P17, P18, P20, P41, P50, P51, P53, P56, P57.*
- *Virtual Machine-based (VM).* In this approach, the middleware programming support virtualizes the IoT/CPS computation and application infrastructure. The applications have a form of separated modules distributed over the network as VM nodes. The advantages of using a VM-based approach include adaptability and transparent inseparability [15]. Within our SLR, we found few studies which take advantage of the VM-based approach: *P2, P4, P8, P20, P29,* and *P49.*
- *Agent-based.* Some studies (e.g. *P18* in our study) apply the agent-based approach to middleware design, in which the distributed mobile agents are used to represent the components of an IoT/CPS. Agents migrate across the nodes of the network while maintaining their execution state. The decentralized IoT/CPS design using an agent-based approach can provide some advantages such as fault-tolerance.
- *Tuple-spaces.* In this approach, each IoT/CPS component holds a tuple data repository that forms a federated tuple space on, e.g., gateway. The approach suits the IoT/CPS mobile components to share data under gateway network constraints. The communication of applications takes place through federated tuple space by writing and reading intended tuple data patterns. We found out that studies *P34, P38, P50,* and *P60* use tuple-spaces approach.
- *Database-oriented.* In this middleware approach, the IoT/CPS are seen as a virtual database from which an application can retrieve intended data. This approach is suitable for distributed data-oriented systems focusing on interoperability. Studies *P10, P13, P15, P17, P24, P31, P35, P36, P45, P53,* and *P55* used database-oriented approach.
- *Application-specific.* This middleware focuses on the specific application domain and its requirements. The architecture designed with this approach well suits the specific IoT/CPS infrastructure. The studies *P3, P8, P16, P17, P21, P26, P32, P54,* and *P60.*

8.3 Middleware Tools

According to our SLR, some middleware proposals have the form of conceptual architectures. Some others customize their middleware, and the rest design or use reliable middleware platforms that usually are open-source and can be reused. As the focus of this study, we look more deeply into the reliable open-source platforms categories that our primary studies use, specify if they are open-source, and assess their specifications and level of *industrial adoption* (Table 6). Afterward, each middleware will be linked to *application domains* for which they are suitable. It is worth

mentioning that the other types of middleware platforms (conceptual and customized) are analyzed in our *data extraction form*.

DEECO [31]. DEECo (Dependable Emergent Ensembles of Components) is a model and framework for developing complex smart CPS. DEECo provides the holistic view that combines the goals of a system, the system's operational model (including real-time constraints), and realistic communication model (including limited communication and latency). Its main specifications are as follows: *i)* is a component-based framework with ensembles for dynamic CPS middleware support; *ii)* is mapped to Java via an internal domain-specific language; *iii)* is implemented via distributed tuple-space middleware or periodic broadcast; *iv) shortcoming:* not still industrialized.

LinkSmart [4, 60]. This middleware implements applications that communicate with the gateway device to capture, e.g., environmental data. The retrieved data are published by the middleware to upper layers, and the middleware further receives commands to act upon the environment. The main specifications of Linksmart are as follows: *i)* is an ambient intelligent middleware; *ii)* uses a lower-level data acquisition component to collect accurate data from context providers; *iii)* includes self-management features comprising goal management, change management, and component control; *iv)* has four layers: semantic, service, network, and security; *v) shortcoming:* not very much applied on real use-cases.

ThingsJS [22]. Is a Javascript-based middleware and runtime environment to address some QoS such as dependability, security and interoperability. Its key characteristics are as follows: *i)* abstracts large-scale distributed systems considerations, such as scheduling, monitoring and self-adaptation; *ii)* uses a constraint model, a multi-dimensional resource prediction approach and a SMT-based scheduler; *iii)* suitable to manage heterogeneous IoT devices; *iv) shortcoming:* not industrialized.

DeviceHive [41]. Is a scalable open-source IoT platform for data collection, processing and analysis, visualization, and device management. It has a wide range of device integration options. Three steps should be followed to use DeviceHive, *a)* creating the instances of the cloud, *b)* connecting the devices and cloud using a dedicated gateway, *c)* visualizing the data via the web. Its main characteristics are: *i)* ease of installation, the rich documentation and the high integration with a wide range of programming languages and IoT protocols; *ii)* is designed with security approaches; *iii)* supports public and private clouds and hybrid deployment; *iv)* has container-based service-oriented architecture approach with linear scalability, managed and orchestrated by Kubernetes for production loads; *v) shortcoming:* measurement data on the device is cached, so the data will be lost when the server is restarted.

Eclipse OM2M [2, 3, 46, 47]. Is an open-source middleware layer that provides a RESTful API for XML data exchange through even unreliable connections within a highly distributed environment. Among its characteristics, we highlight the followings: *i)* provides a horizontal Service Common Entity (CSE) that can be deployed in an M2M server, a gateway, or a device; *ii)* is built as an Eclipse product using Maven and Tycho; *iii) shortcoming:* lack of supported protocols and localization management.

Open-HAB [57, 58]. Is mostly used for smart home applications to control different systems in one single graphical user interface or app. OpenHAB can connect and communicate with heterogeneous IoT/CPS devices through its binding connection modules. Users can define flexible logic rules using a rule engine that commands the IoT/CPS devices. It notably: *i)* has pluggable architecture which supports different technologies, systems, and devices; *ii)* has flexible engine to design time and event-based rules; *iii)* runs the users' server on various operating systems to be

accessed by mobile and web apps; *iv) shortcomings:* little support for Wi-Fi devices, and difficult to transfer between web and file system settings.

OpenIoT [4, 48]. Enables the unification of diverse IoT applications at the cloud level. It provides an integrated development environment for managing mashups of the available IoT services. Its significant specifications are as follows: *i)* provides semantic interoperability in the Cloud; *ii)* applies the Semantic Sensor Networks (SSN) ontology for semantic unification of IoT systems; *iii)* its architecture has seven elements, including a sensor middleware, cloud data storage, and scheduler; *iv) shortcoming:* uses a virtual sensor approach in which data are still provided to the application, even if the service is not available, which can lead to the delivery of out-of-date data.

CHOReOS [48]. Is a service-oriented middleware which consists of mechanisms that facilitate the access to services, the discovery of services, and the composition of services. Its key specifications are as follows: *i)* enables large-scale choreographies of adaptable, QoS-aware, and heterogeneous services; *ii)* relies on development of a composed service, based on the choreography of services; *iii) shortcoming:* service composition component is aimed for a specific application in their project.

GSN [48]. Is a middleware designed to facilitate the deployment and programming of sensor networks. Its design follows four main goals, namely simplicity, adaptability, scalability, and light-weight implementation. GSN is mainly characterized as follows: *i)* provides dynamic adaption of the system configuration during operation; *ii)* uses a container-based approach, which allows different sensors to be easily identified; *iii)* network analysis techniques can be applied; *iv) shortcomings: a)* does not cover the interoperation and context awareness; *b)* sensor search functionality problem by scaling the connected sensors.

UBIWARE [48]. This middleware consists of several agents and agent types with specific set of tasks. A group of agents is associated with running the platform, and another group takes care of applications and users. The most significant specifications of UBIWARE are as follows: *i)* is agent-based and provides a platform for the development of self-managed complex industrial systems; *ii)* integrates Ubiquitous Computing with Semantic Web to address IoT requirements; *iii) shortcomings: a)* there is just one agent to manage all the requests; *b)* there is a huge number of messages that the process generates.

M-Hub [24]. The Mobile Hub (M-Hub) is a general-purpose middleware that enables mobile personal devices to become a gateway for the lower level IoT devices. Its most recognized specifications are as follows: *i)* is a general-purpose middleware; *ii)* enables mobile personal devices to become the propagator nodes; *iii)* provides context information such as local time, date, and location; *iv) shortcoming:* lack of significant applications.

JCL [9]. Is a distributed lightweight Java-based middleware that supports a collaborative multi-developer cluster environment where applications can interact without explicit dependencies. Its other key characteristics are as followed: *i)* incorporates a single application programming interface to program different device categories; *ii)* provides the interoperability of sensing, processing, storage, and actuating services; *iii)* facilitates the integration with MQTT technology; *iv)* allows enabling/disabling data encryption through API, during code execution; *v) shortcoming:* not properly evaluated.

Xively [9]. Uses a central message bus to route the message from devices to other components. The middleware is cloud-based and provides many development tools and resources for supporting developers to connect and obtain data from their sensors. The specifications are as follows: *i)* is a data-driven platform with the ability to give fine-grain access to data streams and data feed;

ii) has additional services which allow for business services, systems integration, and business opportunities for companies; *iii) shortcomings: a)* very limited extensibility models to incorporate new capabilities into the platform, *b)* cloud-only model limitations.

Table 6. Open-source IoT/CPS Middleware platforms used by the primary studies.

Middleware	Associated Studies	Language	Open-source	Industrial Adoption	Used Domain
DEECO [31]	P1	Java	Yes	No	Smart Parking
LinkSmart [4, 60]	P3, P51	Python	Yes	Yes	Smart Home
ThnigsJS [22]	P4	Javascript	Yes	No	Smart Home
DeviceHive [41]	P5	Python, Javascript	Yes	Yes	Smart Cities, Automotive, Energy
Eclipse OM2M [2, 3, 46, 47]	P8, P12, P21, P26	Java	Yes	Yes	Monitoring, Smart Cities
Open-HAB [57, 58]	P17, P60	Java	Yes	Yes	Smart Home
OpenIoT [4, 48]	P18, P51	Java	Yes	Yes	Smart Cities, Mobile Crowd Sensing, and Assistance Living
CHOReOS [48]	P18	Java	Yes	Yes	Smart Home, Smart Energy, Smart Health
GSN [48]	P18	Java, Scala	Yes	Within the project	Smart Cities
UBIWARE [48]	P18	S-APL	Yes	Yes	Industrial systems, Smart Cities
M-Hub [24]	P19	Java	Yes	Yes	Healthcare
JCL [9]	P59	Java	Yes	No	Smart Home
Xively [9]	P59	Java, Javascript, Python, C++	Yes	Yes	Smart Cities, Smart Home

8.4 Industrial Adoption

This section assesses the primary studies' evaluation approaches to learn their maturity level, industry participation, and tool specification [10]. The focus of this section is on self-adaptive middleware support for *industrial IoT/CPS*.

Readiness Level. As suggested by [10, 38, 45], the maturity of research can be assessed in a three-level scale: *i) low:* the study is formulated, validated, or demonstrated in a lab-based environment; *ii) medium* the study is validated or demonstrated in an industrial context; *iii) high:* the study is completed, demonstrated, or proven in the operational environment. Our SLR reveals that in self-adaptive middleware for IoT/CPS domain, several studies (11/62) are not adequately validated. However, several studies have the medium (29/62) or even high (22/62) level of maturity. As shown in Table 6, several papers provide middleware platforms that are widely proven in the operational environment. This emphasizes that self-adaptive middleware for IoT/CPS is going towards maturity regarding technology transfer to industry.

Industry Involvement. Several conferences and journals dedicate industry-oriented tracks and issues to encourage industrial partners' involvement in producing articles. We consider a paper as *pure academic* if all authors come from universities and research institutes, and *industry touched* if at least one author affiliated by a company is among the authors. We observed that several studies (10/62) confirm knowledge transfer among researchers and industrial communities.

Tool Support. We categorized the tools as conceptual, customized, and operative. Table 6 shows the operative open-source tools, which are part of the last category mentioned above. Overall, (52/62) provided a kind of tool to evaluate their research approach, from which 29 tools were operative, 13 tools were customized, and 10 tools had a conceptual form. This result shows that while the IoT/CPS self-adaptive middleware tools have a high-level maturity to be operational, they are mostly specific to projects and use-cases and require additional development efforts.

Open-source. We found 16 open-source middleware in our SLR, from which 13 platforms with solid documentation and reliable source-code are shown in Table 6. As specified in the table, some middleware platforms are used by several primary studies, and some are only used within a project. In the following sections, we discuss how the primary studies' reliable open-source middleware platforms can benefit industrial use-cases with different characteristics and adaptation needs.

Application Domains. As shown in Table 6, the self-adaptive middleware domains span from smart cities and smart buildings to e-healthcare. Despite that most of the use-cases presented by the primary studies deal with pervasive systems, some address industrial CPS, smart energy, and IoT-based automotive systems. We noticed that various application domains have different quality requirements. For instance, while smart production systems need a high dependability level, intelligent monitoring needs an adequate level of performance, and smart homes require low energy consumption. Therefore, the application domain can drive both the software architecture and middleware design.

Answer to RQ3: The primary studies significantly investigate middleware support for self-adaptation. Middleware platforms can have various goals, such as managing and discovering the IoT/CPS resources, managing the data perceived and transmitted by heterogeneous IoT/CPS elements, managing events generated by IoT/CPS applications, and managing the code allocation and migration. The solutions corresponding to those objectives might follow various methods based on events, services, virtual machines, agents, tuple spaces, databases, and applications. Middleware platforms' potential industrial adoption depends on industrial use-cases' adaptation and functional requirements and middleware design approaches.

9 HORIZONTAL ANALYSIS

This section reports the results orthogonal to the vertical analysis presented in the previous sections. For this section, we cross-tabulated and grouped the data, we compared pairs of concepts of our classification framework and identified perspectives of interest.

9.1 Architectural Distribution Patterns VS Quality Attributes

Here the question is "*Which IoT quality attributes should remarkably be assured to design an appropriate IoT pattern?*" A software architecture distribution pattern over another exposes specific quality attributes for the IoT/CPS. As shown in Figure 12, a hierarchical design is the best option to address availability, security, and reliability. In general, a hierarchical architecture could reduce the risk of a single point of failure. If one fog node fails, the IoT/CPS can shift the computation to another fog to keep the system operational. The global control of the hierarchical pattern can facilitate security aspects as well.

Within our SLR, the distributed architectures best guaranteed interoperability, scalability, and dependability. In IoT/CPS applications with heterogeneous static and mobile elements, the system should ensure the possibility of adding new devices which can faultlessly communicate with other resources. Furthermore, several studies used a centralized pattern to guarantee their performance and adaptability requirements. While using a centralized local processing element can avoid network

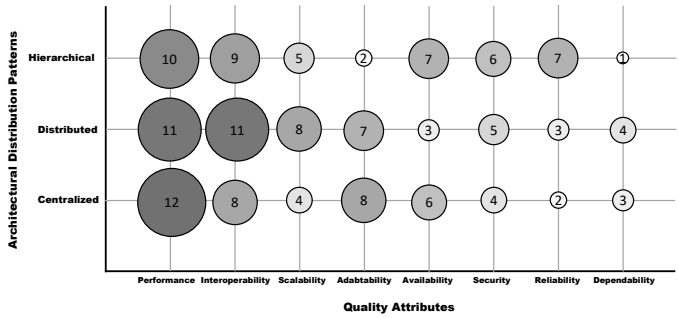

Fig. 12. Architectural distribution patterns VS quality attributes.

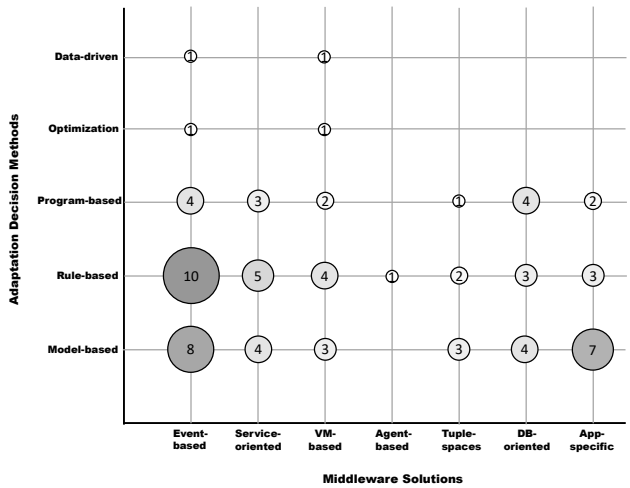

Fig. 13. Adaptation decision methods VS middleware solutions.

delays, it might instead cause computation delays. Assessing the trade-off between a centralized approach and adaptability depends on the application domain and shall be further investigated.

9.2 Adaptation Decision Methods VS Middleware Solutions

Figure 13 shows adaptation decision methods adopted by various middleware solutions. As previously mentioned, the *event-based middleware* interact by event[condition]/action adaptation *rules* with potentially using publish/subscribe *models*. Furthermore, *service-oriented* and *virtual machine-based* middleware approaches take advantage of rule management containers for adaptation. In agent-based middleware, the distributed mobile agents follow some contextual and behavioral rules to adapt and migrate across the network. The figure shows that tuple-based and data-based oriented middleware solutions follow either program-based or rule-based adaptation models, and application-specific solutions mostly deal with model-based adaption techniques.

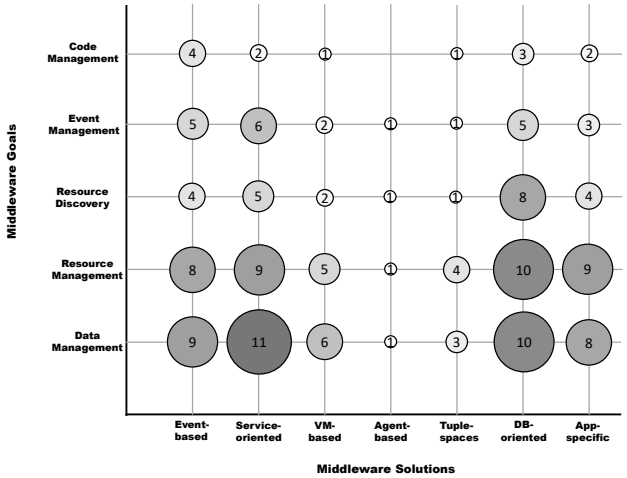

Fig. 14. Middleware goals VS middleware solutions.

9.3 Middleware Goals VS Middleware Solutions

Figure 14 shows the relationship between middleware goals and solutions. Raw data collected by IoT/CPS sensors should be filtered, aggregated, and analyzed. Data management is mainly linked with service-oriented and database-oriented solutions. The cost of data transmission compared with local processing should also be well evaluated for various application domains. In IoT/CPS, conflict among different types of sensory data and corresponding databases is undeniable. Such disharmony can happen in other resources as well. Thus, conflict resolution is often required to resolve conflicts in resources utilized for multiple services. A way to deal with such conflicts could be using agent-based cooperative solutions [52]. From the relation between resource discovery and database-oriented solutions, we understand that number of registries and registry distribution should have a trade-off. While a centralized registry and discovery approach can provide consistency, it can suffer from scalability issues. Besides, since many events are generated in IoT/CPS, the middleware service might become a bottleneck point in both processing and storage dimensions. As a final point, IoT/CPS should support code allocation and migration. Despite that management and event-based solutions are perceived to be correlated, virtual machine-based and application-specific middleware solutions offer support for code management.

10 DISCUSSION CONSIDERING SOME INDUSTRIAL USE-CASES

From the literature reviewed above, we understood that self-adaptation and its support could be well-adopted by industry. Taking advantage of some real-life industrial use-cases within the CPS4EU project [3], we analyzed their functionality and quality requirements and their decision support potentials towards self-adaptation.

[3]Part of the use-cases' explanations comes from CPS4EU deliverable. CPS4EU is a three years project funded by the H2020-ECSEL-2018-IA. The project develops four vital IoT technologies, namely computing, connectivity, sensing, and cooperative systems. It incorporates those IoT technologies through pre-integrated architectures and design tools. It instantiates the architectures in dedicated use-cases from a strategic application viewpoint for automotive, smart grid, and industrial automation. https://cps4eu.eu

10.1 Case 1: RTE Smart Grid System

Renewable energy systems that convert wind and sun into electricity are growing as the primary electricity source. Such renewable generation is mainly connected to the distribution grid but impacts the transmission grid as well. Considering the RTE case, some areas of the grid include many wind-farms as sources of energy, so if a strong wind blows and the generation becomes excessive, an overload will occur on transmission lines. Therefore, to avoid the risk of overloading the lines and creating danger for goods and people's safety, the peak current has to be managed. Instead of developing new power lines, the French Transmission System Operator's policy, called optimal development, investigates new exploitation methods of the existing electrical installations and favors their optimal operation. Wind-farm generation can be limited by opening their feeder's circuit breaker, or more efficiently, by modulating their generation. Additional means can also be used, such as batteries, power electronics, and *IoT*. Dealing with transmission overload risk necessitates considering some information from sensors such as values of currents and voltages on every line, state of the network circuit breakers, and state of battery's charge.

Non-functional Requirements. *Dependability:* the system should be tolerant to faults which might happen in sensors, controllers, and actuators nodes. A dependability requirement is that no more than one unwanted order to the actuators shall be sent every ten years. Furthermore, the system should guarantee the availability in 99.9% of operation time. *Performance:* modulating wind-farms' generation is a solution exposed to a high actuation time when batteries can store electricity in a few seconds. Thus, the system needs to make quick decisions to keep the performance within an acceptable threshold. More precisely, the calculation shall occur in less than 2s. *Security:* Security requirements are related to both the facility that houses the system(s) and the operational security requirements of the system itself. In RTE's operating system level, the use of a secured Linux CentOS (7.4) is mandatory. In the identification process, the use of the RTE industrial Active Directory is mandatory. For the event log, a log shall trace all events linked to identification, access control, resource access, and operation.

Adaptation decision technique proposal. *Architecture reconfiguration:* In order to keep the mentioned non-functional requirements in a proper threshold, the CPS might need to perform self-adaptation at the architectural level [42]. RTE uses a traditional central architecture that can be enhanced to a hierarchical pattern with distributed collaborative controllers that can turn into centralized or distributed patterns if needed. The control algorithm can be run on gateways, local server, or the cloud. The dynamic architecture adaptation scenarios should be simulated to provide more detailed proposals on the pattern's suitability. *MAPE-K loops:* The MAPE-K logic is proposed to be used in architectural reconfiguration. The functional and autonomic controllers can take advantage of MAPE-K control loops to ensure the functionalities and intended qualities. The functional controllers use the MAPE-K loop to sense and affect the environment towards goals. The autonomic control elements adopt a MAPE-K loop to assess the conformity between the architectural components, including functional controllers' situation and goals.

Middleware proposal. RTE intends distributed implementation using middleware that should: *i)* use REST API to exchange data with the central controller, *ii)* communicate with gateways, sensors, remote transmission units, battery management system, generators, and distribution system operators.

Middleware Goal. *Resource management:* based on the requirements mentioned above, resource management is the key objective of using middleware. Distributed and heterogeneous sensors and actuators should be managed in a way to enhance quality attributes. *Data management:* the RTE smart grid case takes advantage of various data sources. As an example, current values are read every second at each end of the transmission lines. This various data source for a unique decision

necessitates precise data filtering and fusion to be performed by middleware. *Event Management:* as we mentioned above, the middleware ought to react and adapt the system to sudden events, e.g., excess of current on the transmission lines.

Middleware Method: *Event-based:* in the event-based method, the components with various states interact with events. In the RTE case, various modes specify the state of control elements. For instance, activating circuit breakers in a critical environmental situation requires the normal calculation of levers' usage or data-driven generation prediction. *Service-oriented:* from another point of view, various applications can be defined as services. For instance, a service could be providing situational awareness to operators by a dashboard, and another could be saving electricity in batteries.

Tool proposal. *Open-HAB:* we suggest using Open-HAB since it supports the adaptation of event-based systems. Furthermore, the RTE system does not need for device discovery using Wi-Fi. *DeviceHive:* is validated in the smart energy domain. It enhances security and provides data insights. It also enables the practical usage of resources and allows a faster and more flexible self-adaptation.

10.2 Case 2: ACOEM Smart Cities System

IoT/CPS are used to provide urban security, which involves monitoring crowds, cars' congestion, and air quality. The concept of distributed collaborative sensor networks is generally used for such smart city applications. In the ACOEM case, the sensors network's purpose is to provide a well-being index to the population and the city's safety aspect. The two main applications developed in this case are: *i)* Environment Quality Index, and *ii)* Geo-localized threats alerts.Traditional environmental quality systems need expensive metrological stations. Thus, establishing many of those infrastructures to provide local information is unfeasible. ACOEM aims to develop environmental pods to measure different pollutants. With specific data analysis on the measurement performed by the pod network installed in a city, it is possible to increase each pod's accuracy and understand the origin of the pollution. This service's critical point is to provide accurate and local measurement with a cost-effective network of pods.

For the threat alert system, the system detects an abnormal situation in real-time, identifies the threat (e.g., gunshot), and provides the sound origin's azimuth/elevation. When an abnormal noise is detected, many parameters are computed based on the audio signal. All these parameters are the inputs of a neuronal network specially trained to identify a gunshot. This neuronal network can realize the difference between, e.g., a firecracker, a nozzle gunshot noise, and a supersonic bullet noise. To train this neuronal network, a gunshot database from a defense product experience was used. Self-adaptation plays a substantial role in both mentioned ACOEM services. In both systems, the system architecture can be switched in run-time to provide better quality. The system goals might enforce data analysis sometimes on the device and some other times at the network edge or cloud.

Non-functional Requirements. In smart IoT/CPS with a lot of sensory nodes, *scalability, performance, dependability*, and *energy efficiency* are among important requirements. The system must be open to scaling with new static and mobile sensors and, consequently, computation resources. Scaling the system should keep the performance in an acceptable range. Especially in threat alert service, the system response time should be kept small to be compliant with real-time requirements. The system should also be dependable so that faulty nodes do not impact its operation. Since many IoT/CPS resources are battery-powered, energy efficiency is another concern.

Adaptation decision technique proposal. We propose architectural reconfiguration, which adopts feedback control loops to decide the combination of computation elements dynamically. Based on the threat alerts system's specification, it requires a data-driven approach as well.

Middleware Proposal. Both above-mentioned IoT/CPS-based services require a middleware which provides interoperability in a massive network of sensors.

Middleware Goal. The middleware should not only manage the IoT/CPS resources but also refine, fuse, and analyze the various source of data coming from different sensors.

Middleware Method. We propose a service-oriented approach for the environment quality index system in which the service should be available in real-time. The threat analysis system, which should be activated based on specific events, can take advantage of an event-based solution.

Tool Proposal. GSN is the right choice since it is designed for sensor networks and supports system configuration's scalability and adaptability.

10.3 Case 3: ARCURE Pedestrian Detection Around Off-road Construction Trucks

Detecting pedestrians around heavy vehicles enhances the autonomy of IoT/CPS-based industrial machinery (such as off-road construction trucks) regarding urban security. In this context, some smart cameras can distinguish a person from an obstacle in real-time and alert the operator with a visual and sound signal in case of danger. A control screen in the cabin allows the driver to judge the critical nature of the situation and then avoid an accident. This process can be realized by a Deep Learning algorithm for person detection, and automatized by self-adaptation techniques. The main features to develop are: *i)* detecting pedestrian with atypical position; *ii)* detecting the pedestrians from a distance of 0.3 to 15 meters; and *iii)* reaching a high level of reliability necessary for autonomous machinery. Achieving these goals needs the algorithm to be trained using a large and diverse data-set of images in an industrial context. The self-adaptive IoT/CPS can perform architectural reconfiguration at sensors, computing, and actuators levels.

Non-functional Requirements. Within the IoT/CPS-based pedestrian detection system, self-adaptation techniques should guarantee the detection function and enhance the system qualities such as *performance, reliability*, and *energy efficiency*. The processing performance has to be at least 550 tera-operations per second. Furthermore, the system must detect the pedestrian and alert the user in less than 200ms. Respecting the performance requirements improves the environmental and urban safety measures. Another important aspect is that the system should be reliable and tolerated to any possible fault. It is worth mentioning that in such safety-critical IoT/CPS, the system shall detect all internal failures, and it has to warn the user if the detection is not working correctly. Another essential requirement is that the power consumption shall be kept less than 15W.

Adaptation decision technique proposal. The *architectural reconfiguration* is necessary for respecting the precision needed for pedestrian detection. If a sensor becomes faulty or the delay becomes more than the specified threshold, the architecture can be adapted to take advantage of additional sensors or computation resources. The reconfiguration can adopt *MAPE-K* feedback loop to monitor the system and environment situation to react continuously. Besides, since the detection system is based on deep learning algorithms, *data-driven* adaptation is a must.

Middleware Proposal. Using a middleware can provide inseparability among heterogeneous sensors and provide a suitable dashboard for user decision making when needed.

Middleware Goal. In this particular case, all IoT/CPS resources are known, but their run-time *management* is necessary. *Data management* is an essential middleware task due to the data-driven nature of the detection system. Apart from human detection, other types of internal data can be managed, such as HW temperature, energy consumption, and computation delay.

Middleware Method. We propose a *database-oriented* approach from which applications retrieve data from a virtual database. This necessitates *virtual machine-based* method in which computation and application resources are virtualized.

Tool Proposal. *JCL* could be a proper middleware since it provides the interoperability of all IoT/CPS resources, including sensing, processing, and actuating.

11 THREATS TO VALIDITY

According to Petersen et al. [49], the quality rating for this systematic mapping study was assessed and scored as 73% (see the *Replication Package*). This value is the ratio of the number of actions taken compared to the total number of actions reported in the quality checklist. Our study's quality score is far beyond the scores obtained by existing systematic mapping studies in the literature, which have a distribution with a median of 33% and 48% as an absolute maximum value. However, the threats to validity are unavoidable. Below we shortly define the main threats to our study's validity and the way we mitigated them.

External Validity: In our study, the most severe threat related to external validity may consist of having a set of primary studies that are not representative of the whole research on IoT architectural styles. We mitigated this potential threat by *i)* following a search strategy including both automatic search and backward-forward snowballing of selected studies; *ii)* defining a set of inclusion and exclusion criteria. Along the same lines, gray and non-English literature are not included in our research as we want to focus exclusively on state of the art presented in high-quality scientific studies in English.

Internal Validity: It refers to the level of influence that extraneous variables may have on the study's design. We mitigated this potential threat to validity by *i)* rigorously defining and validating the structure of our research, *ii)* defining our classification framework by carefully following the keywording process, *iii)* and conducting both the vertical and horizontal analysis.

Construct Validity: It concerns the validity of extracted data with respect to the research questions. We mitigated this potential source of threats in different ways. *i)* performing an automatic search on multiple electronic databases to avoid potential biases; *ii)* having a strong and tested search string; *iii)* Complementing the automatic by the snowballing activity; *iv)* rigorously screen the studies according to inclusion and exclusion criteria.

Conclusion Validity: It concerns the relationship between the extracted data and the obtained results. We mitigated potential threats to conclusion validity by applying well accepted systematic methods and processes throughout our study and documenting all of them in the excel package.

12 CONCLUSION

This paper presents a systematic literature review study to classify and identify the domain state-of-the-art and extract a set of middleware solutions for self-adaptive IoT/CPS. Starting from over 4,274 potentially relevant studies, we applied a selection procedure resulting in 62 primary studies. The results of this study are both research and industry-oriented and are intended to design and develop middleware for their self-adaptive applications. In future work, we will more extensively assess the potential integration of existing research to various IoT/CPS industrial use-cases within the CPS4EU project.

REFERENCES

[1] Claudio Arbib, Davide Arcelli, Julie Dugdale, Mahyar Moghaddam, and Henry Muccini. 2019. Real-time emergency response through performant IoT architectures. In *International Conference on Information Systems for Crisis Response and Management (ISCRAM)*.

[2] MN Bahiri, A Zyane, and A Ghammaz. 2018. An enhancement for the Autonomic Middleware-Level Scalability Management within IoT System using Cloud Computing. In *International Conference on Electronic Engineering and Renewable Energy*. Springer, 80–88.

[3] Yassine Banouar, Saad Reddad, Codé Diop, Christophe Chassot, and Abdellah Zyane. 2015. Monitoring solution for autonomic Middleware-level QoS management within IoT systems. In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*. IEEE, 1–8.

[4] Ayoub Benayache, Azeddine Bilami, Sami Barkat, Pascal Lorenz, and Hafnaoui Taleb. 2019. MsM: A microservice middleware for smart WSN-based IoT application. *Journal of Network and Computer Applications* 144 (2019), 138–154.

[5] Pere Botella, X Burgués, JP Carvallo, X Franch, G Grau, J Marco, and C Quer. 2004. ISO/IEC 9126 in practice: what do we need to know. In *Software Measurement European Forum*, Vol. 2004. Citeseer.

[6] Cyril Cecchinel, François Fouquet, Sébastien Mosser, and Philippe Collet. 2019. Leveraging live machine learning and deep sleep to support a self-adaptive efficient configuration of battery powered sensors. *Future Generation Computer Systems* 92 (2019), 225–240.

[7] Muhammad Aufeef Chauhan, Muhammad Ali Babar, and Boualem Benatallah. 2017. Architecting cloud-enabled systems: a systematic survey of challenges and solutions. *Software: Practice and Experience* 47, 4 (2017), 599–644.

[8] Keling Da, Philippe Roose, Marc Dalmau, Joseba Nevado, and Riadh Karchoud. 2014. Kali2Much: a context middleware for autonomic adaptation-driven platform. In *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT*. 25–30.

[9] Leonardo de Souza Cimino, José Estevão Eugênio de Resende, Lucas Henrique Moreira Silva, Samuel Queiroz Souza Rocha, Matheus de Oliveira Correia, Guilherme Souza Monteiro, Gabriel Natã de Souza Fernandes, Renan da Silva Moreira, Junior Guilherme de Silva, Matheus Inácio Batista Santos, et al. 2019. A middleware solution for integrating and exploring IoT and HPC capabilities. *Software: Practice and Experience* 49, 4 (2019), 584–616.

[10] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2019. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software* 150 (2019), 77–97.

[11] Nathalia Moraes do Nascimento and Carlos José Pereira de Lucena. 2017. FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things. *Information Sciences* 378 (2017), 161–176.

[12] Michael Donohoe, Brendan Jennings, and Sasitharan Balasubramaniam. 2015. Context-awareness and the smart grid: Requirements and challenges. *Computer Networks* 79 (2015), 263–282.

[13] Julie Dugdale, Mahyar Tourchi Moghaddam, and Henry Muccini. 2020. Human Behaviour Centered Design: Developing a Software System for Cultural Heritage. In *International Conference on Software Engineering. ICSE-SEIS'2020*. ACM, 85–94.

[14] Nikil Dutt, Axel Jantsch, and Santanu Sarma. 2015. Self-aware cyber-physical systems-on-chip. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 46–50.

[15] Mohammad Mehdi Faghih and Mohsen Ebrahimi Moghaddam. 2011. SOMM: A new service oriented middleware for generic wireless multimedia sensor networks based on code mobility. *Sensors* 11, 11 (2011), 10343–10371.

[16] Soodeh Farokhi, Pooyan Jamshidi, Ivona Brandic, and Erik Elmroth. 2015. Self-adaptation challenges for cloud-based applications: A control theoretic perspective. In *10th international workshop on feedback computing*, Vol. 2015.

[17] Antonio Filieri, Giordano Tamburrelli, and Carlo Ghezzi. 2015. Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. *IEEE Transactions on Software Engineering* 42, 1 (2015), 75–99.

[18] Robert France and Bernhard Rumpe. 2007. Model-driven development of complex software: A research roadmap. In *Future of Software Engineering (FOSE'07)*. IEEE, 37–54.

[19] Mirco Franzago, Davide Di Ruscio, Ivano Malavolta, and Henry Muccini. 2016. Protocol for a systematic mapping study on collaborative model-driven software engineering. *arXiv preprint arXiv:1611.02619* (2016).

[20] Mirco Franzago, Davide Di Ruscio, Ivano Malavolta, and Henry Muccini. 2017. Collaborative model-driven software engineering: a classification framework and a research map. *IEEE Transactions on Software Engineering* 44, 12 (2017), 1146–1175.

[21] David Garlan, Bradley Schmerl, and Shang-Wen Cheng. 2009. Software architecture-based self-adaptation. In *Autonomic computing and networking*. Springer, 31–55.

[22] Julien Gascon-Samson, Mohammad Rafiuzzaman, and Karthik Pattabiraman. 2017. Thingsjs: Towards a flexible and self-adaptable middleware for dynamic and heterogeneous iot environments. In *Proceedings of the 4th Workshop on Middleware and Applications for the Internet of Things*. 11–16.

[23] Sona Ghahremani, Holger Giese, and Thomas Vogel. 2017. Efficient utility-driven self-healing employing adaptation rules for large dynamic architectures. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 59–68.

[24] Berto de Tácio Pereira Gomes, Luiz Carlos Melo Muniz, Francisco José Da Silva e Silva, Davi Viana Dos Santos, Rafael Fernandes Lopes, Luciano Reis Coutinho, Felipe Oliveira Carvalho, and Markus Endler. 2017. A middleware with comprehensive quality of context support for the internet of things applications. *Sensors* 17, 12 (2017), 2853.

[25] Dat Dac Hoang and Hye-Young Paik Chae-Kyu Kim. 2011. Service-oriented middleware architectures for cyber-physical systems. (2011).

[26] M Usman Iftikhar and Danny Weyns. 2014. Activforms: Active formal models for self-adaptation. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 125–134.

[27] ISO/IEC/IEEE. 2011. ISO/IEC/IEEE 42010, Systems and software engineering – Architecture description. (2011).

[28] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. 2016. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications* 34, 5 (2016), 1728–1739.

[29] Ghazaleh Javadzadeh and Amir Masoud Rahmani. 2020. Fog Computing Applications in Smart Cities: A Systematic Survey. *Wireless Networks* 26, 2 (2020), 1433–1457.

[30] Hamzeh Khazaei, Alireza Ghanbari, and Marin Litoiu. 2018. Adaptation as a service.. In *CASCON*. 282–288.

[31] Michal Kit, Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, and Frantisek Plasil. 2015. An architecture framework for experimentations with self-adaptive cyber-physical systems. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 93–96.

[32] Barbara Kitchenham and Pearl Brereton. 2013. A systematic review of systematic review process research in software engineering. *Information and software technology* 55, 12 (2013), 2049–2075.

[33] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).

[34] Barbara Kitchenham, Rialette Pretorius, David Budgen, O Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen Linkman. 2010. Systematic literature reviews in software engineering–a tertiary study. *Information and software technology* 52, 8 (2010), 792–805.

[35] Georgia Koutsandria, Reinhard Gentz, Mahdi Jamei, Anna Scaglione, Sean Peisert, and Chuck McParland. 2015. A real-time testbed environment for cyber-physical security on the power grid. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. 67–78.

[36] Philippe Lalanda, Stéphanie Chollet, Colin Aygalinc, and Eva Gerbert-Gaillard. 2015. Service-based architecture and frameworks for pervasive health applications. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 1–8.

[37] Georgios Lilis and Maher Kayal. 2018. A secure and distributed message oriented middleware for smart building applications. *Automation in Construction* 86 (2018), 163–175.

[38] John C Mankins. 1995. Technology readiness levels. *White Paper, April* 6 (1995), 1995.

[39] Mahyar Tourchi Moghaddam and Henry Muccini. 2019. Fault-Tolerant IoT. In *International Workshop on Software Engineering for Resilient Systems*. Springer, 67–84.

[40] Mahyar T Moghaddam, Eric Rutten, Philippe Lalanda, and Guillaume Giraud. 2020. IAS: an IoT Architectural Self-adaptation Framework. In *European Conference on Software Architecture*. Springer, 333–351.

[41] Marina Mongiello, Tommaso Di Noia, Francesco Nocera, Eugenio Di Sciascio, and Angelo Parchitelli. 2016. Context-aware design of reflective middleware in the internet of everything. In *Federation of International Conferences on Software Technologies: Applications and Foundations*. Springer, 423–435.
[42] Henry Muccini and Mahyar Tourchi Moghaddam. 2018. Iot architectural styles. In *European Conference on Software Architecture*. Springer, 68–85.
[43] Henry Muccini, Mohammad Sharaf, and Danny Weyns. 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*. 75–81.
[44] Henry Muccini, Romina Spalazzese, Mahyar T Moghaddam, and Mohammad Sharaf. 2018. Self-adaptive IoT architectures: An emergency handling case study. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*. 1–6.
[45] European Commission [Online]. 2017. Technology readiness levels (TRL). (2017), 1. https://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2016_2017/annexes/h2020-wp1617-annex-g-trl_en.pdf
[46] Clovis Anicet Ouedraogo, Samir Medjiah, and Christophe Chassot. 2018. A modular framework for dynamic qos management at the middleware level of the iot: Application to a onem2m compliant iot platform. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.
[47] Clovis Anicet Ouedraogo, Samir Medjiah, Christophe Chassot, and Khalil Drira. 2018. Enhancing middleware-based IoT applications through run-time pluggable Qos management mechanisms. application to a oneM2M compliant IoT middleware. *Procedia computer science* 130 (2018), 619–627.
[48] Andrei Palade, Christian Cabrera, Fan Li, Gary White, Mohammad Abdur Razzaque, and Siobhán Clarke. 2018. Middleware for Internet of Things: an evaluation in a small-scale IoT environment. *Journal of Reliable Intelligent Environments* 4, 1 (2018), 3–23.
[49] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18.
[50] Jesús MT Portocarrero, Flavia C Delicato, Paulo F Pires, Bruno Costa, Wei Li, Weisheng Si, and Albert Y Zomaya. 2017. RAMSES: a new reference architecture for self-adaptive middleware in wireless sensor networks. *Ad Hoc Networks* 55 (2017), 3–27.
[51] Behrouz Pourghebleh and Vahideh Hayyolalam. 2019. A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things. *Cluster Computing* (2019), 1–21.
[52] Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and Siobhán Clarke. 2015. Middleware for internet of things: a survey. *IEEE Internet of things journal* 3, 1 (2015), 70–95.
[53] Matthias Rohr, Simon Giesecke, Marcel Hiel, Willem-Jan van den Heuvel, Hans Weigand, and Wilhelm Hasselbring. 2006. A classification scheme for self-adaptation research. (2006).
[54] Eric Rutten, Nicolas Marchand, and Daniel Simon. 2017. Feedback control as MAPE-K loop in autonomic computing. In *Software Engineering for Self-Adaptive Systems III. Assurances*. Springer, 349–373.
[55] Douglas C Schmidt. 2006. Model-driven engineering. *Computer-IEEE Computer Society-* 39, 2 (2006), 25.
[56] Sebastian Scholze, José Barata, and Oliver Kotte. 2013. Context Awareness for self-adaptive and highly available Production Systems. In *Doctoral Conference on Computing, Electrical and Industrial Systems*. Springer, 210–217.
[57] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Aßmann. 2019. Toward a framework for self-adaptive workflows in cyber-physical systems. *Software & Systems Modeling* 18, 2 (2019), 1117–1134.
[58] Ronny Seiger, Steffen Huber, and Thomas Schlegel. 2018. Toward an execution system for self-healing workflows in cyber-physical systems. *Software & Systems Modeling* 17, 2 (2018), 551–572.
[59] Stepan Shevtsov, Mihaly Berekmeri, Danny Weyns, and Martina Maggio. 2017. Control-theoretical software adaptation: A systematic literature review. *IEEE Transactions on Software Engineering* 44, 8 (2017), 784–810.
[60] Alberto MC Souza and Jose RA Amazonas. 2013. A novel smart home application using an internet of things middleware. In *Smart SysTech 2013; European Conference on Smart Objects, Systems and Technologies*. VDE, 1–7.
[61] Feng Wang, Liang Hu, Jin Zhou, and Kuo Zhao. 2015. A data processing middleware based on SOA for the internet of things. *Journal of Sensors* 2015 (2015).
[62] Danny Weyns. 2017. Software engineering of self-adaptive systems: an organised tour and future challenges. *Chapter in Handbook of Software Engineering* (2017).
[63] Danny Weyns and Michael Georgeff. 2009. Self-adaptation using multiagent systems. *IEEE software* 27, 1 (2009), 86–91.
[64] Danny Weyns, M Usman Iftikhar, Danny Hughes, and Nelson Matthys. 2018. Applying architecture-based adaptation to automate the management of internet-of-things. In *European Conference on Software Architecture*. Springer, 49–67.
[65] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M Göschka. 2013. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*. Springer, 76–107.
[66] Claes Wohlin. 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*. 1–10.
[67] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.
[68] Yu Wu and Minbo Li. 2018. An IoT Middleware of Data Service. In *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 121–128.
[69] He Zhang, Muhammad Ali Babar, and Paolo Tell. 2011. Identifying relevant studies in software engineering. *Information and Software Technology* 53, 6 (2011), 625–637.
[70] Tianqi Zhao, Wei Zhang, Haiyan Zhao, and Zhi Jin. 2017. A reinforcement learning-based framework for the generation and evolution of adaptation rules. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 103–112.

PRIMARY STUDIES

- **P1**: Kit, Michal, Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, and Frantisek Plasil. "An architecture framework for experimentations with self-adaptive cyber-physical systems." *In 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 93-96. IEEE, 2015.
- **P2**: Park, Soojin, and Sungyong Park. "A Cloud-based Middleware for Self-Adaptive IoT-Collaboration Services." *Sensors 19*, no. 20 (2019): 4559.

- **P3**: Souza, Alberto MC, and Jose RA Amazonas. "A novel smart home application using an internet of things middleware." *In Smart SysTech 2013; European Conference on Smart Objects, Systems and Technologies*, pp. 1-7. VDE, 2013.
- **P4**: Gascon-Samson, Julien, Mohammad Rafiuzzaman, and Karthik Pattabiraman. "Thingsjs: Towards a flexible and self-adaptable middleware for dynamic and heterogeneous iot environments." *In Proceedings of the 4th Workshop on Middleware and Applications for the Internet of Things*, pp. 11-16. 2017.
- **P5**: Mongiello, Marina, Tommaso Di Noia, Francesco Nocera, Eugenio Di Sciascio, and Angelo Parchitelli. "Context-aware design of reflective middleware in the internet of everything." *In Federation of International Conferences on Software Technologies: Applications and Foundations*, pp. 423-435. Springer, Cham, 2016.
- **P6**: Cecchinel, Cyril, François Fouquet, Sébastien Mosser, and Philippe Collet. "Leveraging live machine learning and deep sleep to support a self-adaptive efficient configuration of battery powered sensors." *Future Generation Computer Systems 92* (2019): 225-240.
- **P7**: Sylla, Adja Ndeye, Maxime Louvel, Eric Rutten, and Gwenaël Delaval. "Modular and hierarchical discrete control for applications and middleware deployment in iot and smart buildings." *In 2018 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1472-1479. IEEE, 2018.
- **P8**: Banouar, Yassine, Saad Reddad, Codé Diop, Christophe Chassot, and Abdellah Zyane. "Monitoring solution for autonomic Middleware-level QoS management within IoT systems." *In 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1-8. IEEE, 2015.
- **P9**: Kim, Hyun-Woo, Jong Hyuk Park, and Young-Sik Jeong. "Efficient resource management scheme for storage processing in cloud infrastructure with internet of things." *Wireless Personal Communications 91*, no. 4 (2016): 1635-1651.
- **P10**: Handte, Marcus, Pedro José Marrón, and Gregor Schiele, eds. Adaptive Middleware for the Internet of Things: The GAMBAS Approach. *River Publishers*, 2019.
- **P11**: Lilis, Georgios, and Maher Kayal. "A secure and distributed message oriented middleware for smart building applications." *Automation in Construction 86* (2018): 163-175.
- **P12**: Ouedraogo, Clovis Anicet, Samir Medjiah, and Christophe Chassot. "A modular framework for dynamic qos management at the middleware level of the iot: Application to a onem2m compliant iot platform." *In 2018 IEEE International Conference on Communications (ICC)*, pp. 1-7. IEEE, 2018.
- **P13**: Wu, Yu, and Minbo Li. "An IoT Middleware of Data Service." *In 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 121-128. IEEE, 2018.
- **P14**: Mohalik, Swarup Kumar, Nanjangud C. Narendra, Ramamurthy Badrinath, Mahesh Babu Jayaraman, and Chakri Padala. "Dynamic semantic interoperability of control in IoT-based systems: Need for adaptive middleware." *In 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 199-203. IEEE, 2016.
- **P15**: Wang, Feng, Liang Hu, Jin Zhou, and Kuo Zhao. "A data processing middleware based on SOA for the internet of things." *Journal of Sensors* 2015 (2015).
- **P16**: Kazmi, Aqeel, Zeeshan Jan, Achille Zappa, and Martin Serrano. "Overcoming the heterogeneity in the internet of things for smart cities." *In International workshop on interoperability and open-source solutions*, pp. 20-35. Springer, Cham, 2016.
- **P17**: Seiger, Ronny, Steffen Huber, and Thomas Schlegel. "Toward an execution system for self-healing workflows in cyber-physical systems." *Software & Systems Modeling 17*, no. 2 (2018): 551-572.
- **P18**: Palade, Andrei, Christian Cabrera, Fan Li, Gary White, Mohammad Abdur Razzaque, and Siobhán Clarke. "Middleware for Internet of Things: an evaluation in a small-scale IoT environment." *Journal of Reliable Intelligent Environments 4*, no. 1 (2018): 3-23.
- **P19**: Gomes, Berto de Tácio Pereira, Luiz Carlos Melo Muniz, Francisco José Da Silva e Silva, Davi Viana Dos Santos, Rafael Fernandes Lopes, Luciano Reis Coutinho, Felipe Oliveira Carvalho, and Markus Endler. "A middleware with comprehensive quality of context support for the internet of things applications." *Sensors 17*, no. 12 (2017): 2853.
- **P20**: Bao, Kaibin, Ingo Mauser, Sebastian Kochanneck, Huiwen Xu, and Hartmut Schmeck. "A microservice architecture for the intranet of things and energy in smart buildings." *In Proceedings of the 1st International Workshop on Mashups of Things and APIs*, pp. 1-6. 2016.
- **P21**: Ouedraogo, Clovis Anicet, Samir Medjiah, Christophe Chassot, and Khalil Drira. "Enhancing middleware-based IoT applications through run-time pluggable Qos management mechanisms. application to a oneM2M compliant IoT middleware." *Procedia computer science 130* (2018): 619-627.
- **P22**: Rahman, Mahmudur, Amatur Rahman, Hua-Jun Hong, Li-Wen Pan, Md Yusuf Sarwar Uddin, Nalini Venkatasubramanian, and Cheng-Hsin Hsu. "An adaptive IoT platform on budgeted 3g data plans." *Journal of Systems Architecture 97* (2019): 65-76.
- **P23**: Qin, Zhijing, Grit Denker, Carlo Giannelli, Paolo Bellavista, and Nalini Venkatasubramanian. "A software defined networking architecture for the internet-of-things." *In 2014 IEEE network operations and management symposium (NOMS)*, pp. 1-9. IEEE, 2014.
- **P24**: Andersen, Michael P., Gabe Fierro, and David E. Culler. "Enabling synergy in iot: Platform to service and beyond." *Journal of Network and Computer Applications 81* (2017): 96-110.
- **P25**: Benson, Kyle E., Guoxi Wang, Nalini Venkatasubramanian, and Young-Jin Kim. "Ride: A resilient iot data exchange middleware leveraging sdn and edge cloud resources." *In 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 72-83. IEEE, 2018.
- **P26**: Bahiri, M. N., A. Zyane, and A. Ghammaz. "An enhancement for the Autonomic Middleware-Level Scalability Management within IoT System using Cloud Computing." *In International Conference on Electronic Engineering and Renewable Energy*, pp. 80-88. Springer, Singapore, 2018.
- **P27**: Dobrescu, Radu, Daniel Merezeanu, and Stefan Mocanu. "Context-aware control and monitoring system with IoT and cloud support." *Computers and Electronics in Agriculture 160* (2019): 91-99.
- **P28**: Weißbach, Martin, Nguonly Taing, Markus Wutzler, Thomas Springer, Alexander Schill, and Siobhan Clarke. "Decentralized coordination of dynamic software updates in the Internet of Things." *In 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 171-176. IEEE, 2016.
- **P29**: Naranjo, Paola G. Vinueza, Enzo Baccarelli, and Michele Scarpiniti. "Design and energy-efficient resource management of virtualized networked Fog architectures for the real-time support of IoT applications." *The journal of Supercomputing 74*, no. 6 (2018): 2470-2507.
- **P30**: Vanneste, Simon, Jens de Hoog, Thomas Huybrechts, Stig Bosmans, Reinout Eyckerman, Muddsair Sharif, Siegfried Mercelis, and Peter Hellinckx. "Distributed uniform streaming framework: an elastic fog computing platform for event stream processing and platform transparency." *Future Internet 11*, no. 7 (2019): 158.
- **P31**: Da, Keling, Philippe Roose, Marc Dalmau, Joseba Nevado, and Riadh Karchoud. "Kali2Much: a context middleware for autonomic adaptation-driven platform." *In Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT*, pp. 25-30. 2014.

- **P32**: Lalanda, Philippe, Stéphanie Chollet, Colin Aygalinc, and Eva Gerbert-Gaillard. "Service-based architecture and frameworks for pervasive health applications." *In 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1-8. IEEE, 2015.
- **P33**: Qin, Zhijing, Luca Iannario, Carlo Giannelli, Paolo Bellavista, Grit Denker, and Nalini Venkatasubramanian. "MINA: A reflective middleware for managing dynamic multinetwork environments." *In 2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1-4. IEEE, 2014.
- **P34**: Caporuscio, Mauro, Vincenzo Grassi, Moreno Marzolla, and Raffaela Mirandola. "G o P rime: A Fully Decentralized Middleware for Utility-Aware Service Assembly." *IEEE Transactions on Software Engineering 42*, no. 2 (2015): 136-152.
- **P35**: Chen, Chunhua, and Junwei Yan. "HyTube: A Novel Middleware Layer for Smart Building Systems." In 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, *16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress* (DASC/PiCom/DataCom/CyberSciTech), pp. 135-142. IEEE, 2018.
- **P36**: Koutsandria, Georgia, Reinhard Gentz, Mahdi Jamei, Anna Scaglione, Sean Peisert, and Chuck McParland. "A real-time testbed environment for cyber-physical security on the power grid." *In Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, pp. 67-78. 2015.
- **P37**: Dutt, Nikil, Axel Jantsch, and Santanu Sarma. "Self-aware cyber-physical systems-on-chip." *In 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 46-50. IEEE, 2015.
- **P38**: Siegemund, Gerry, and Volker Turau. "A self-stabilizing publish/subscribe middleware for iot applications." *ACM Transactions on Cyber-Physical Systems 2*, no. 2 (2018): 1-26.
- **P39**: Razouk, Wissam, Daniele Sgandurra, and Kouichi Sakurai. "A new security middleware architecture based on fog computing and cloud to support IoT constrained devices." *In Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, pp. 1-8. 2017.
- **P40**: Singh, Jatinder, Thomas Pasquier, Jean Bacon, Julia Powles, Raluca Diaconu, and David Eyers. "Big ideas paper: Policy-driven middleware for a legally-compliant Internet of Things." *In Proceedings of the 17th International Middleware Conference*, pp. 1-15. 2016.
- **P41**: Prazeres, Cássio, Jurandir Barbosa, Leandro Andrade, and Martin Serrano. "Design and implementation of a message-service oriented middleware for fog of things platforms." *In Proceedings of the Symposium on Applied Computing*, pp. 1814-1819. 2017.
- **P42**: Merlino, Giovanni, Rustem Dautov, Salvatore Distefano, and Dario Bruneo. "Enabling Workload Engineering in Edge, Fog, and Cloud Computing through OpenStack-based Middleware." *ACM Transactions on Internet Technology (TOIT) 19*, no. 2 (2019): 1-22.
- **P43**: Rosa, Nelson, Gláucia Campos, and Davi Cavalcanti. "Using software architecture principles and lightweight formalisation to build adaptive middleware." *In Proceedings of the 16th Workshop on Adaptive and Reflective Middleware*, pp. 1-7. 2017.
- **P44**: Naber, Jens, Martin Pfannemüller, Janick Edinger, and Christian Becker. "PerFlow: configuring the information flow in a pervasive middleware via visual scripting." *In Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 434-443. 2019.
- **P45**: Paspallis, Nearchos, and George A. Papadopoulos. "A pluggable middleware architecture for developing context-aware mobile applications." *Personal and Ubiquitous Computing 18*, no. 5 (2014): 1099-1116.
- **P46**: Portocarrero, Jesús MT, Flávia C. Delicato, Paulo F. Pires, Taniro C. Rodrigues, and Thais V. Batista. "SAMSON: self-adaptive middleware for wireless sensor networks." *In Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 1315-1322. 2016.
- **P47**: Javed, Asad, Jérémy Robert, Keijo Heljanko, and Kary Främling. "IoTEF: A Federated Edge-Cloud Architecture for Fault-Tolerant IoT Applications." *Journal of Grid Computing (2020)*: 1-24.
- **P48**: Baresi, Luciano, Sam Guinea, and Adnan Shahzada. "SeSaMe: towards a semantic self adaptive middleware for smart spaces." *In International Workshop on Engineering Multi-Agent Systems*, pp. 1-18. Springer, Berlin, Heidelberg, 2013.
- **P49**: Weyns, Danny. "Software engineering of self-adaptive systems." *In Handbook of Software Engineering*, pp. 399-443. Springer, Cham, 2019.
- **P50**: Bouloukakis, Georgios, Nikolaos Georgantas, Patient Ntumba, and Valerie Issarny. "Automated synthesis of mediators for middleware-layer protocol interoperability in the IoT." *Future Generation Computer Systems* 101 (2019): 1271-1294.
- **P51**: Benayache, Ayoub, Azeddine Bilami, Sami Barkat, Pascal Lorenz, and Hafnaoui Taleb. "MsM: A microservice middleware for smart WSN-based IoT application." *Journal of Network and Computer Applications 144* (2019): 138-154.
- **P52**: Rafique, Ansar, Dimitri Van Landuyt, Eddy Truyen, Vincent Reniers, and Wouter Joosen. "SCOPE: self-adaptive and policy-based data management middleware for federated clouds." *Journal of Internet Services and Applications 10*, no. 1 (2019): 2.
- **P53**: Pease, Sarogini Grace, Paul P. Conway, and Andrew A. West. "Hybrid ToF and RSSI real-time semantic tracking with an adaptive industrial internet of things architecture." *Journal of Network and Computer Applications 99* (2017): 98-109.
- **P54**: Colin, Aygalinc, Eva Gerbert-Gaillard, German Vega, Philippe Lalanda, and Stéphanie Chollet. "Autonomic service-oriented context for pervasive applications." *In 2016 IEEE International Conference on Services Computing (SCC)*, pp. 491-498. IEEE, 2016.
- **P55**: Donohoe, Michael, Brendan Jennings, and Sasitharan Balasubramaniam. "Context-awareness and the smart grid: Requirements and challenges." *Computer Networks 79* (2015): 263-282.
- **P56**: Zhao, Mengxuan, Gilles Privat, Eric Rutten, and Hassane Alla. "Discrete control for smart environments through a generic finite-state-models-based infrastructure." *In European Conference on Ambient Intelligence*, pp. 174-190. Springer, Cham, 2014.
- **P57**: de Brito, Mathias Santos, Saiful Hoque, Ronald Steinke, Alexander Willner, and Thomas Magedanz. "Application of the fog computing paradigm to smart factories and cyber-physical systems." *Transactions on Emerging Telecommunications Technologies 29*, no. 4 (2018): e3184.
- **P58**: Koziolek, Heiko, Andreas Burger, Marie Platenius-Mohr, Julius Rückert, Francisco Mendoza, and Roland Braun. "Automated industrial IoT-device integration using the OpenPnP reference architecture." *Software: Practice and Experience* (2019).
- **P59**: de Souza Cimino, Leonardo, José Estevão Eugênio de Resende, Lucas Henrique Moreira Silva, Samuel Queiroz Souza Rocha, Matheus de Oliveira Correia, Guilherme Souza Monteiro, Gabriel Natã de Souza Fernandes et al. "A middleware solution for integrating and exploring IoT and HPC capabilities." *Software: Practice and Experience 49*, no. 4 (2019): 584-616.
- **P60**: Seiger, Ronny, Steffen Huber, Peter Heisig, and Uwe Aßmann. "Toward a framework for self-adaptive workflows in cyber-physical systems." *Software & Systems Modeling 18*, no. 2 (2019): 1117-1134.
- **P61**: Casado-Vara, Roberto, Pablo Chamoso, Fernando De la Prieta, Javier Prieto, and Juan M. Corchado. "Non-linear adaptive closed-loop control system for improved efficiency in IoT-blockchain management." *Information Fusion 49* (2019): 227-239.
- **P62**: Portocarrero, Jesús MT, Flavia C. Delicato, Paulo F. Pires, Bruno Costa, Wei Li, Weisheng Si, and Albert Y. Zomaya. "Ramses: a new reference architecture for self-adaptive middleware in wireless sensor networks." *Ad Hoc Networks 55* (2017): 3-27.