The Journal of Systems & Software 203 (2023) 111716

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

A systematic review on security and safety of self-adaptive systems☆

Irdin Pekaric [a,c,*], Raffaela Groner [b], Thomas Witte [b], Jubril Gbolahan Adigun [a], Alexander Raschke [b], Michael Felderer [a,d,e], Matthias Tichy [b]

[a] University of Innsbruck, Department of Computer Science, Technikerstraße 21a, A-6020 Innsbruck, Austria
[b] Ulm University, Institute of Software Engineering and Programming Languages, James-Franck-Ring 9, 89081 Ulm, Germany
[c] University of Liechtenstein, Department of Information Systems and Computer Science, Fürst-Franz-Josef-Strasse, 9490 Vaduz, Liechtenstein
[d] German Aerospace Center (DLR), Institute for Software Technology, Linder Hoehe, 51147 Cologne, Germany
[e] University of Cologne, Department of Mathematics and Computer Science, Albertus-Magnus-Platz, 50923 Cologne, Germany

ARTICLE INFO

Article history:
Received 25 January 2022
Received in revised form 28 February 2023
Accepted 16 April 2023
Available online 4 May 2023

Keywords:
Self-adaptive system
MAPE-K
Security and safety
Attack mechanisms
Safety hazards

ABSTRACT

Context: Cyber–physical systems (CPS) are increasingly self-adaptive, i.e. they have the ability to introspect and change their behavior. This self-adaptation process must be considered when modeling the safety and security aspects of the system.
Objective: This study collects and compares security attacks and safety hazards on self-adaptive systems (SAS) described in the literature. In addition, mitigation and treatment strategies, as well as the modeling and analysis approaches, are investigated.
Method: We conducted a systematic literature review on 21 selected papers. The selection process included a database search on four scientific databases using a common search string (1430 papers), forward and backward snowballing (1402 papers), and filtering the results based on predefined inclusion and exclusion criteria. The coding scheme to analyze the content of the papers was obtained through research questions, existing domain-specific taxonomies, and open coding.
Results: Safety and security are not jointly modeled in the context of self-adaptive systems. The adaptation process is often not considered in the attack and hazard analysis due to naïve assumptions and modeling. The proposed approaches are mostly verified and validated through simulation often using simple use cases and scenarios.
Conclusion: A thorough and joint modeling approach for safety and security in self-adaptive systems is still an open challenge that needs to be addressed. Further work is needed to address the gap between safety and security modeling in self-adaptive systems.

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

1. Introduction

Historically, safety and security were mostly studied in separation. Both domains independently developed techniques and modeling approaches for conducting different types of analyses. Safety-critical systems increasingly expose high software complexity and connectivity that warrant security concerns such as e.g., in the automotive (Amorim et al., 2017), medical devices (Johnson and Kelly, 2019), and robotics domain Sesame Project (2022). A joint analysis of safety and security aspects in cyber–physical systems (CPS) is therefore necessary, particularly to expose security implications on system safety and vice versa. Several mainly methodological approaches to safety and security co-engineering have been proposed recently (Kondeva et al., 2019; Chockalingam et al., 2017).

Self-adaptive systems (SAS), i.e., systems that can adapt their behavior or architecture at runtime, offer a promising and biologically inspired approach to manage or mitigate risks (Macías-Escrivá et al., 2013). Such adaptation mechanisms are already computationally feasible and in use in safety and security-critical domains, e.g. robotics, critical infrastructure, and production systems or transportation. Additional software components to provide the necessary introspection and adaptation capabilities for self-adaptation increase the software complexity significantly. This makes the adaptation mechanism itself a target for security attacks or the source of software failures and it has, therefore, to be included in the joint safety and security analysis.

☆ Editor: Christoph Treude.

* Corresponding author at: University of Liechtenstein, Department of Information Systems and Computer Science, Fürst-Franz-Josef-Strasse, 9490 Vaduz, Liechtenstein.
E-mail address: irdin.pekaric@uni.li (I. Pekaric).

https://doi.org/10.1016/j.jss.2023.111716

Currently, a comprehensive overview of existing work on the intersection of joint safety and security analysis in the context of self-adaptive systems is missing. The increasing computational power available to cyber–physical systems and their steadily increasing complexity makes self-adaptation more and more viable in the future. Safety and security-critical robotic systems like autonomous and connected cars, home assistance or transportation, and logistic robots will become ubiquitous and pervasive in the near future and will require runtime validation and certification for applications such as platooning or integration into robot fleet managers.

We conducted a systematic literature review, finally selecting 21 out of 1430 papers identified using database search and 1402 papers found via snowballing. In the literature, we identified attacks and hazards for self-adaptive systems and determined how safety and security are interrelated, jointly modeled, analyzed, and mitigated. Open challenges found in these papers hint at unsolved problems and possible future research directions. Relevant papers were collected using a predefined search string on multiple scientific databases and by applying an additional forward and backward snowballing iteration, manually filtering the results based on predefined inclusion and exclusion criteria after each step. We coded, collected, and extracted information on the treatment of security attacks and safety hazards. In addition, architectures that are used by the system or the adaptation process were investigated. This was achieved using a codebook, which contains all codes used and their documentation (MacQueen et al., 1998). The codebook was partially predefined using existing taxonomies and adapted during coding using an open coding approach. Open coding is an iterative process in which data are categorized based on concepts and themes identified by the researcher based on the data being analyzed (Williams and Moser, 2019).

The 21 selected papers are recent and most are published in the year 2015 or later. These are mainly from the automotive domain and utilize the MAPE-K architecture to implement the self-adaptation process. The identified attacks and safety hazards show a strong relationship in these systems, e.g. an attack leading to a safety hazard. We could not identify a common or standardized approach that is used to model and analyze this interrelationship between safety and security in self-adaptive systems. To mitigate safety hazards or as a reaction to security attacks, most strategies try to prevent damage to the system. Adaptations that use redundant systems, reduce capabilities, or stop the operation of the system altogether are commonly used. According to the results, most current systems integrate safety and security only loosely without considering self-adaptation capabilities as an integral part of the system. The self-adaptation process is merely an afterthought to resolve emerging safety and security problems. The self-adaptation process is often modeled with unrealistic, implicit assumptions such as that it is always succeeding in zero-time, or is not modeled at all.

The examined papers use a wide variety of analytical approaches for both security and safety analysis. In regards to these approaches, simulation and dynamic analysis at runtime are well suited due to the dynamic nature of self-adaptive systems.

Extracted open challenges as well as our results show the need for further research to jointly model and analyze safety and security in self-adaptive systems to adequately address and analyze safety hazards and security vulnerabilities impacting each other. As the self-adaptation mechanism is an integral part of the system, it must be taken into consideration during modeling and analysis.

The remainder of this paper will introduce significant background concepts in Section 2 and give an overview of related studies in Section 3. We describe our methodology and criteria for paper selection and analysis in Section 4, before presenting and discussing our results and findings in Sections 5 and 6. Finally, we conclude the paper in Section 7.

2. Background

In this section, the background information on self-adaptive systems such as their definition and common architectures is presented. In addition, the security and safety aspects that are used in the classification scheme presented in Section 4.3, are introduced.

2.1. Self-adaptive system

Birthed by a challenge posed by IBM in 2003 (IBM, 2005) leading to the founding of the International Conference on Autonomic Computing (ICAC), self-adaptive systems have grown in popularity over the last two decades. With the notion of self-management in autonomic computing that allows a computing system to dynamically modify its configuration in response to a change in the system while adhering to business goals, *Self-adaptive Systems (SAS)* refer to systems that are able to respond to changes in the operating environment thereby affecting the system's structure, behavior or even the system's logic. In some studies, self-adaptive systems are referred to as Dynamically Adaptive Systems (DAS) (dos Santos et al., 2021). All these systems exhibit certain characteristics such as self-healing, self-optimization, and self-configuration (Wong et al., 2022). To gain a common understanding, in the following, we describe briefly some of the terms used in SAS-related studies.

2.1.1. Adaptation

Adaptation is what gives SASs the ability to modify their behavior in response to changes within and outside the system. The type of adaptation possible is usually described in terms of self-* properties. Because adaptation can manifest itself in different ways, studies explore these various adaptation strategies employed by self-adaptive systems. At the primitive level of SAS, adaptivity is achieved through a system's ability to perceive its environment *(self-awareness or context-awareness)* (Salehie and Tahvildari, 2009). In addition to self-awareness, several other self-* properties can be defined of which the following are frequently encountered:

- *Self-protection* describes a group of self-managing systems capable of detecting and mitigating security threats at runtime (Yuan et al., 2014).
- *Self-healing* is the ability of a system to return to a functional state after being compromised by an anomalous agent. Self-healing systems endeavor to *heal* themselves in a similar fashion as a biological system heals from a wound from faults and recovers into normalcy (Ghosh et al., 2007).
- *Self-optimization* allows for systems to find the optimal solution such as configuration or output.
- *Self-configuration* is the feature of a system to dynamically and automatically reconfigure itself in response to changes. It allows for the adaptation of a new component or new execution environment within a system with little or no human intervention (Khan et al., 2008).

2.1.2. General architecture

In general, a self-adaptive system consists of two distinct parts, wherein the first part (self-awareness element) interacts with the environment, while the second part cooperates with the first part and deals with any adaptation concerns (Weyns, 2017). The self-awareness element requires that SAS is able to get information about its state through a process by which it *monitors* itself and its environment, *detects* changes, *decides* how to react, and *acts* based on the decisions made. This is known as the MAPE cycle (MAPE-K loop) (Kephart and Chess, 2003), which

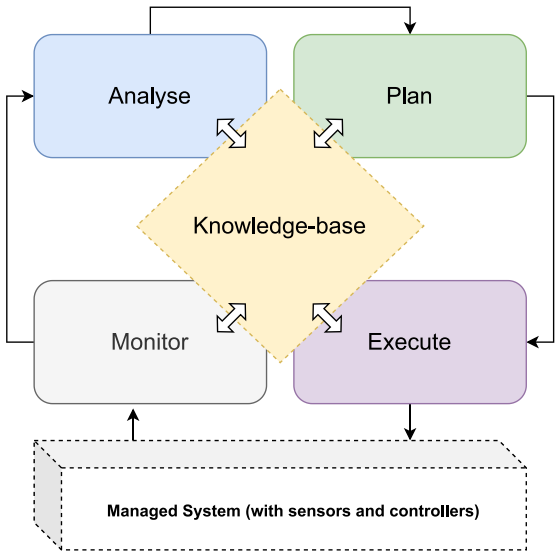

Fig. 1. General MAPE-K Architecture.

relies on an adaptation logic. M stands for Monitoring (system and environment), A for Analysis (what change should happen?), P for Planning (decision(s) to be made), E for Execution (of the action decided), and K for Knowledge base (a shared space that maintains data of the system and its environment). Fig. 1 shows a general MAPE-K architecture. The eventual detail of each object in the architecture depends on the application domain.

For example, in Carré et al. (2018), the MAPE-K architecture is applied to the safety domain to create a scalable and evolvable framework that decouples the different concerns into microservices that perform safety assurance and validation.

2.2. Adaptation properties

In the assurance of SASs, both functional (i.e. those relating to the function of the system) and non-functional properties (i.e. those relating to the runtime qualities of the system) must be taken into account. The fulfillment of these requirements may be impacted by the self-adaptation of the system. Even though the managed system or operational environment may pose uncertainties, the properties of the system specifications should hold *before, during* and *after* the whole adaptation process. The term *adaptation properties* refers to the characteristics (non-functional operational properties) of the system that need to be maintained to ascertain the normal operating condition of such a system. Examples of the qualities are: availability, efficiency, performance, reliability, robustness, security, and stability (Cheng et al., 2014). Some of these qualities that resulted from open coding are presented and described further in Section 5.4.2.

2.3. Security and safety

Due to the widespread use and ubiquity of CPS in our daily life today, malfunctions of these software-intensive systems can become a danger to life. In addition to physical harm, economic, ecological, and emotional risks can also occur. The IEC standard 61508 on functional safety defines safety as "freedom from unacceptable risks" (IEC 61508 2010, 2010). Minimizing these risks is a major field of research in the software engineering context.

In order to secure a system, it is not sufficient to consider only existing attacks, but also the functionalities of the system. The increasing interconnection of systems means that security is also increasingly influencing the functional safety of a complex system. The possibility of manipulating a system-internal message using an exploit or by making a component fail, can lead to the system reaching a critical state. This only holds if the aforementioned cases have not been taken into account from the beginning of the development.

Safety and security requirements can sometimes contradict each other, e.g., the encryption and displaying of a warning message, which can make the reaction time longer. It is therefore important that these two aspects are considered together right from the beginning in a software development process (Lisova et al., 2019).

The focus of this study lies in the common consideration of safety and security in the context of SASs. In the following, some typical terms of these areas are briefly introduced.

2.3.1. CAPEC mechanisms

Sometimes when a SAS experiences a change, it is usually because of a vulnerability within the system itself that is exploited by an external attacker. The Common Attack Pattern Enumeration and Classification (CAPEC)[1] scheme postulated by the MITRE Corporation demonstrates a hierarchical framework of different ways by which systems are attacked by a foreign entity. These techniques include *Deceptive Interactions, Abuse of Existing Functionality of the System, Data Structure Manipulation, System Resources Manipulation, Injection of Unexpected Items, Employing Probabilistic Techniques, Manipulation of System Timing and State, Collect and Analyze Information*, and *Circumventing or Subverting Access Control.* Each of these techniques is further categorized under different headings (which may also be further sub-categorized) that describe individual attack approaches employed by a perpetrator. For instance, when an attacker employs deceptive means, they could either decide to go by spoofing an identity, content, resource location, action, or manipulate human behavior. Similarly, when the approach is to subvert laid-down access control, this can be done by physically stealing access items, bypassing physical control, using known domain credentials, or abusing and bypassing authentication.

2.3.2. CIA triad

The CIA triad represents the most commonly used and applied information security model. It is used to provide various organizations and types of systems with knowledge on how to keep data secure. The model consists of the following three aspects: (1) confidentiality — prevention of unauthorized disclosure or use of information assets, (2) integrity — prevention of unauthorized modification of information assets, and (3) availability — ensuring authorized access of information assets when required (Oscarson, 2003). The CIA triad is applied in order to better understand which of the aforementioned aspects are affected by attacks on self-adaptive systems as well as to understand the attacker's goals.

2.3.3. Hazard source and cause

The overall objective of safety engineering is to make sure that the unacceptable risk does not transpire (International Organization for Standardization, 2011). The term risk represents the joint probabilities of occurrence of various harms and their respective severities (International Organization for Standardization, 2011). The direct assessment of such harms is extremely difficult. Thus, this needs to be achieved by identifying hazards, which are considered to be potential sources of harm. The identification of hazards is especially complex in the SAS domain because reconfigurations are conducted automatically and usually there is no

[1] https://capec.mitre.org/data/definitions/1000.html

human expert that can evaluate the safety of a system (Trapp and Schneider, 2014). As a result, the system has to assure its safety. In this regard, hazard causes and sources are investigated as a part of the conducted SLR. Hazard causes represent the potential reasons behind why the safety of a SAS was affected. According to Allouch et al. (2019), this can occur due to the following reasons: *collision with another object*, *environmental conditions*, *hardware failure*, *loss of control*, *pilot error* and *external attack*. On the other hand, hazard sources can be classified as *internal* and *external*. For example, an internal factor can be mechanical or software related, while external ones can relate to human and environment interaction (Allouch et al., 2019).

2.3.4. Safety quality factors

In the case that a hazard occurs, it is important to know what factors were affected. This includes *health*, *property* and *environmental* factors (Allouch et al., 2019). For example, in the case of a vehicle crash, it is possible that a person within the car is injured, which would affect health. Another example of the impact on health is when a drone falls to the ground and hits civilians or causes a mid-air collision with an aircraft that carries passengers. On the other hand, vehicle crashes can also cause damage to the vehicle itself, which would be an effect on the property. Finally, it is also possible that the vehicle damages any surrounding environmental object, which would be an effect on the environment. An additional effect could be an environmental catastrophe in case a SAS is transporting some sensitive chemicals or crashes into gas tanks causing a devastating explosion.

3. Related work

Different studies have gone into understanding and developing self-adaptive systems at various levels, which includes the adaptation approach. Accordingly, these studies cover individual aspects of self-adaptive systems or a combination of aspects (e.g. planning for decision-making (Pandey et al., 2016), models and verification at runtime (Blair et al., 2009; Calinescu et al., 2018) respectively, model-driven engineering (Vogel and Giese, 2014) etc.). At the same time, several literature reviews have been conducted to collect the body of knowledge in this area together (e.g. SAS evaluation (Gerostathopoulos et al., 2021)).

3.1. Overview

In a recent study that explored self-adaptive systems over the 30 years before the year 2021, the authors established a review across different categories and domains of self-adaptive systems (Wong et al., 2022). The Internet-of-Things (IoT) account for the largest application domains in the last five years. There has been an increasing amount of effort invested in self-adaptive technologies within the same period in web services and no change in the study category during the first half of the last ten years. The authors presented a matrix that distributed the selected papers (293 in total) according to the following categories: analytical, empirical, technological, methodological, and perspectives. Over the 30 years, studies focused mostly on self-adaptive technologies correlating to 109 papers out of the total 293.

Also, Muccini et al. (2016) provided an overview of self-adaptation in cyber–physical systems (CPSs) with a focus on their architecture. The study provided answers on how adaptation is applied in CPSs, how self-adaptive approaches in CPSs address adaptation concerns, the different assurance strategies employed in providing evidence for adaptation, and the strengths and limitations of such approaches. They also provided a generalized three-layered adaptation model divided into physical, context management, and application layers.

3.2. Adaptation

Salehie and Tahvildari (2009) provided a taxonomy for the development and concerns of adaptation. This was achieved by answering the questions that relate to *what (system properties), how (supporting infrastructure), when (temporal properties), and where (system component changed)* in the self-adaptive software domain. The authors identified challenges in self-* properties, adaptation processes, engineering concerns, and system interaction. Furthermore, they investigated the different disciplines that contribute to the development of SASs, among which are *Software Engineering, Artificial Intelligence, Machine Learning, and Soft Computing and Network and Distributed Computing*. According to the authors, testing and assurance were perhaps the least focused aspects of self-adaptive software. With appropriate yardsticks, test-beds, and suitable case studies, the authors identified that one can evaluate and compare different adaptation solutions.

Gheibi et al. (2021) accentuated the growing application of learning approaches for understanding and analyzing the SAS environment and configuration spaces to facilitate adaptation. In their study, Geibi et al. investigate problems tackled by applying ML in SASs as well as the key engineering considerations for applying ML in this domain. The problems identified tackled adaptation and learning while cloud applications appeared to be the most popular. In addition, the authors identified learning methods open for future research including unsupervised learning, active learning, and adversarial learning. The authors also postulated an "Other learning methods" category for models that can detect novelty in the environment and synchronize the execution paths in complex settings.

Furthermore, D´Angelo et al. (2019) discussed the need for equipping individual components of a collective self-adaptive system (CSAS) with learning capabilities for the system to be able to dynamically handle uncertainty in operational environments. In this work, Reinforcement Learning was found to be the most popular learning technique representing about 60% of the reviewed papers. They carried out an extensive analysis of learning-enabled CSASs and introduced a 3D framework for illustrating the learning aspects of CSASs. The dimensions were *autonomy*, *knowledge access*, and *behavior* with each dimension being orthogonal to the others to form a continuous 3D space. Also, they highlighted various application examples to identify what open challenges there were and prescribe that there is a need for collaboration between resilience and privacy-awareness for CSASs.

3.3. General modeling dimensions and assurance

Cheng et al. (2009) and Andersson et al. (2009) provided modeling dimensions for self-adaptive systems. The authors identified dimensions for modeling and describing various facets of self-adaptation. They grouped the dimensions into four groups that include Goals, Change, Mechanisms, and Effects which were also partly adopted and modified in this publication. This was a study conducted to collect the areas of divergence in the development of self-adaptive systems having gone through various software engineering texts with the guiding insight that any development of a self-adaptive system should be according to a conceived model, regardless of the tools or underlying technologies deployed. The authors' objective was to demonstrate a baseline for key aspects of SAS. With their established modeling dimensions, they were able to showcase their application in two case studies — Traffic Jam Monitoring System (Self-healing) and Embedded Mobile System (Self-Adapting). In Cheng et al. (2009), Cheng et al. further proposed an assurance framework — a V&V model of the system going through a series of operational modes as the system context changes.

On the other end, Weyns et al. (2012) carried out a survey of formal methods used in SAS. It was posited that formal methods were mostly used for modeling and analysis with few studies of formal methods for verification of self-adaptive systems. The authors identified regular algebra as the most popular modeling language among studies that employ the use of tools for property specification. Whereas the use of formal methods was mostly applied concerning efficiency/performance of self-adaptation, the paper did not address the security and safety aspects of SAS.

Additionally, Cheng et al. (2014) explored the contemporary works that use models@runtime (M@RT) to address the assurance of self-adaptive software systems. They identified what information could be captured by M@RT bearing in mind their fit for assurance purposes, accompanying challenges, and the different application domains wherein M@RT could be useful. Additionally, the authors characterized assurance methods based on techniques and non-functional attributes that the assurance methods address.

3.4. Applications and challenges

Altawy and Youssef in Altawy and Youssef (2016) identified the safety and security challenges with respect to cyber–physical threats and physical vulnerabilities in civilian unmanned aerial vehicles (UAVs) while reiterating the different application areas of UAVs as well as privacy concerns with civilian drones, but no aspects of self-adaptation or its architecture were mentioned.

Similar to the work of Salehie and Tahvildari (2009), Macías-Escrivá et al. (2013) also studied the applications and challenges in SAS. The authors provided four common definitions for self-adaptivity and highlighted the different approaches employed in the development of SASs, which are both nature-inspired and engineered. Likewise, they identified global and specific tools and methods that help to achieve self-adaptivity e.g. feedback loop control used in monitoring the system and the process of decision-making after the analysis of data. Similarly, Alcaraz and Zeadally (2015) also studied security challenges in various critical systems but did not tackle the notion of self-adaptation. The focus was placed on organizational and operational security standards and measures with a brief mention of safety.

In comparison to the paper by Muccini et al. (2016), which focuses more on adaptation at the architecture level, this study goes further to explore attack surfaces, modeling approaches that combine security and safety, and V&V approaches. With respect to the number of papers, Muccini et al. studied a final set of 42 papers out of an initial set of 1103 using a combination of manual and automatic searches. This publication on the other hand resulted in 21 papers out of an initial set of 1430 using database searches as well as forward and backward snowballing.

In summary, most studies identified in this section provide a good overview of SASs, while some present ideas related to the two key notions of safety and security e.g. Altawy and Youssef (2016). However, these two features are studied in isolation even though studies have shown that they are related. In this work, combined modeling of security and safety is further explored in the context of self-adaptive systems as there is no study that does this concretely. Again, verification and validation techniques are used to test and evaluate system behaviors and detect unintended interactions that may occur. More so, with dynamic runtime behaviors of self-adaptive systems, it is crucial to be able to provide assurance for these systems throughout their life cycle, both at development time and at runtime. However, since the adaptive nature of SASs only precipitates at runtime, studies have been carried out to address runtime testing of SASs. dos Santos et al. (2021), studied the runtime testing of context-aware variability of adaptive systems by identifying possible failures that occur as a result of the system adaptation and verifying its behavioral properties with an approach dubbed as RETAkE. They evaluated the approach using two DAS examples and a mutation testing technique. Therefore, this paper aims to explore several aspects of the aforementioned related works to develop a comprehensive classification scheme that combines the safety and security elements of self-adaptive systems. In addition, the V&V of SASs are further investigated as they take up more than half of the design and development efforts. This is significant because it extends the coverage of the various components of SAS that have been studied by previous works. In this aspect, the characteristics and properties of SASs that are used in assuring the behavior of SASs are further investigated. This is achieved by introducing a classification dimension that helps to view how certain properties impact the security and safety of SASs.

4. Methodology

This paper aims to answer the following four research questions:

- (RQ1) Which security attacks and safety hazards in the context of self-adaptive systems are described in the literature?
- (RQ2) How are security attacks and safety hazards related and which safety mitigation and treatment strategies exist?
- (RQ3) How are models used in the context of security, safety, and self-adaptive systems, and which analysis approaches are utilized?
- (RQ4) What open challenges are described in the literature in relation to self-adaptive systems?

In order to address these research questions, a systematic literature review was carried out, in which both keyword-based database search (Kitchenham et al., 2009) and snowballing (Wohlin, 2014) methodologies were applied (see Section 4.1). The resulting set of papers contains publications that focus on the security and safety of self-adaptive systems (see Section 4.2). These publications were systematically classified according to a developed classification scheme that directly relates to the four research questions. The classification scheme is described in Section 4.3.

4.1. Search strategy

The goal of the search strategy was to retrieve a representative set of scientific works dealing with safety and security aspects in the context of self-adaptive systems. Therefore, a systematic literature review was conducted by employing a keyword-based database search that was complemented by applying single forward (cf. identification of new publications that cite the one being examined) and single backward (cf. examining the references of a publication being studied) snowballing iteration. According to Badampudi et al. (2015), the efficiency of additional snowballing stages is very low. By employing the two aforementioned methodologies together, sufficient literature coverage should be achieved and threats mitigated, especially since our initial set is quite extensive.

4.2. Search process

The search process consists of the following main phases: *keyword identification*, *database search*, *application of selection criteria*, and *snowballing*. This is illustrated in Fig. 2 together with the different amounts of publications added or excluded in the different phases/steps.

Keyword identification. In order to develop the search string, the terms “‘self-adaptive’”, “‘system’”, “‘safety’”, and “‘security’”

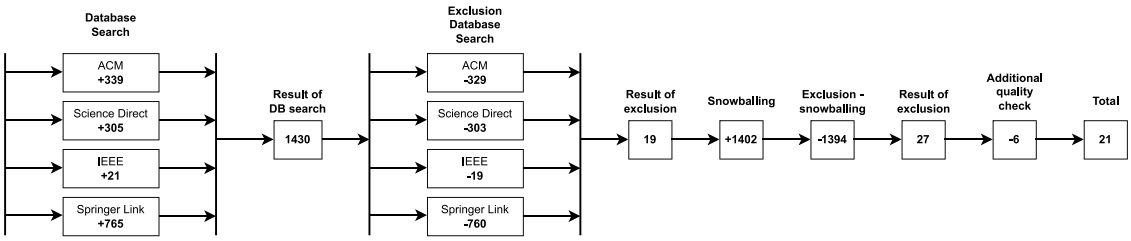

Fig. 2. The search process stages.

were combined. Since safety, security, and system are widely used terms in software engineering, we decided to only search them within abstracts of publications. It is expected that publications that focus on these topics mention them already in the abstract. As we are interested in the combination of safety and security, we require that both terms are mentioned in the abstract. In addition, we strictly focus on the term "'self-adaptive'" – to include self-adaptive systems as well as self-adaptive architecture – and do not consider broader, adjacent terms due to a very large number of results (over 10000 papers), which cannot be checked in a realistic timeline. We expect relevant papers containing synonymous terms to be found through the additional back- and forward snowballing. Additionally, we aim to keep the search string as simple and as concise as possible to avoid over-optimizing it and thereby introducing bias. All of these considerations lead to the formation of the following search string:

```
security[:abstract] AND safety[:abstract] AND
self-adaptive[:full-text] AND system[:abstract].
```

Database search. The keyword search was conducted using the following publication databases: ACM, Science Direct, IEEE Xplore, and Springer Link. This search strategy was applied because they are widely used databases that are utilized within the domains of software engineering, hardware safety and security, and electronic engineering. This allowed us to obtain a good starting set of papers that is highly relevant for the study.

The search resulted in a set of 1430 publications, consisting of 339 papers found through ACM, 305 papers found through Science Direct, 21 papers found through IEEE Xplore, and 765 papers found through Springer Link.[2] However, this set contained a large number of papers that were not relevant for the study, which were removed by applying the following inclusion and exclusion criteria.

Inclusion and Exclusion Criteria. The applied inclusion and exclusion criteria are listed in Table 1. The goal was to obtain a set of high-quality publications and therefore only peer-reviewed articles were considered. This included academic publications, journal papers, conference proceedings, books, and standards. On the other hand, gray and white literature was excluded, which consists of: technical/vendor reports, preprints, news/press, articles, work in progress, unpublished results, expert opinions/experiences based on theory, blog entries, and tweets. In addition to the gray literature, secondary studies such as other literature reviews were also excluded.

Furthermore, only publications that were accessible in the full text were considered in order to avoid any incomplete information for the data extraction phase. The time period from which publications were selected ranges from the year 2000 to the year 2020, which allowed us to obtain a set of recent publications. Regarding the language in which publications were written, only English papers were selected. Moreover, the final set of papers consisted only of papers that address explicitly self-adaptation or self-adaptive architecture such as MAPE-K. These papers also had to address security (describe an attack, security issue, vulnerability) and safety (safety scenario, safety approach). In addition, papers presenting verification and validation approaches combined with self-adaptive aspects of a system were included in the final selection. Finally, all the duplicate papers were excluded in order to avoid any inconsistencies. This involved removal of seven papers during the database search and the removal of 14 papers after the snowballing iterations. In total, this resulted in the elimination of 21 publications.

These inclusion and exclusion criteria were checked in three iterations that involved checking the title, abstract and full text of each publication. This was done by splitting papers equally into four overlapping subsets, wherein each paper was checked by at least two authors, which resulted in an overlap of 25%. Regarding the set for full-text reading, it was checked and discussed by four authors. As a result, a set of 19 papers was obtained, which was used for snowballing iterations.

Snowballing. The snowballing methodology was executed on the starting set obtained through the database search. Forward and backward snowballing iterations resulted in 1402 additional papers (998 backward and 404 forward snowballing), which were evaluated by four authors using the aforementioned overlapping sets procedure. This set was reduced to a total of 27 papers by applying the same inclusion and exclusion criteria as for the database search phase. Besides the application of the inclusion/exclusion criteria, an additional quality check was performed by reading the full text of each of these 27 papers making sure that they strongly relate to the research questions. This also assured that non-relevant publications are excluded from the final set. The check was conducted independently by each of the four authors, in which each of 27 papers was evaluated. In case of conflicts, these were thoroughly discussed and solved. If authors thought that a certain publication did not relate strictly to research questions, the publication would be removed by unanimous decision. As a result, six publications were removed from the set, which resulted in a final set of 21 papers. The confidence in the resulting number was obtained because of the rigorous cross-readings done by the team members. Adding additional variants of the search keywords would return more results, but the experience from preliminary readings and the initial papers that we iterated through, shows that most papers only superficially mention the buzz words such as self-organizing and self-awareness without demonstrating any implementation or example. In addition, it is evident that this seems to be a common trend in the SAS domain. For instance, Gheibi et al. (2021) and D´Angelo et al. (2019) eventually obtained 109 out of 6709 papers and 52 out 6147 papers from their initial automatic searches respectively.

4.3. Development of the classification scheme

The information included in the 21 publications of the final set obtained by the database search and snowballing processes,

[2] Limitations in the search syntax lead to a higher number of results

Table 1
Inclusion and exclusion criteria.

Inclusion	Exclusion
Peer-reviewed articles	Gray and white literature
Accessible in full text	Non-English articles
Published between year 2000 and 2020	Secondary studies
Discusses explicitly self-adaptation, self-healing or relates to a self-adaptive architecture	
Addresses security and safety	
Addresses verification and validation	

was systematically extracted by applying a classification scheme or a codebook. This codebook was developed in accordance with the methodology proposed by Usman et al. (2017). The following five phases were executed: (a) Planning, (b) Identification and Extraction, (c) Design, and Construction, (d) Validation, and (e) Refinement.

Phase (a) included the planning process, where four of the authors proposed ideas for the dimensions. These dimensions represented building blocks for the classification scheme. Phase (b) involved the actual development of the classification dimensions. Some of the dimensions were drawn from the literature and existing standards (e. g. CAPEC metrics The MITRE Corporation, 2021). This basic set of dimensions was extended and refined progressively by the authors while reading publications. For example, whenever at least two papers that belong to a new domain were identified, a new category for this domain was created. After this identification phase, each of the proposed dimensions was discussed and agreed upon among all the authors. Phase (c) and Phase (d) involved combining all the dimensions into the final classification scheme and validating these by classifying the identified papers. Finally, in the last phase (cf. Phase (e)) some dimensions were further refined after attempting to classify several papers. This was done in order to improve the quality of the dimensions and provide a better codebook, in which categories are directly related to the research questions.

Fig. 3 shows the proposed classification scheme which is split into five main categories (*System and its properties*, *Integration*, *Modeling Approach*, *Challenges*, and *Treatment*). These main categories are subdivided into several subcategories that are summarized in the following and described in more detail in the corresponding subsections of Section 5.

Besides the domain in which the described system is applied to (e. g. automotive, IoT, robotics, . . .), the category *System and its properties* focuses on the general characteristics of a self-adaptive system such as the used architecture (see Section 2.1.2). This part of the coding book was developed in an open coding process as described in Gibbs (2018).

The *adaptation strategy* describes concrete measures taken by the self-adaptation process, clustered to common high-level strategies, e.g. redundancy, in an open coding process. *Adaptation realization* similarly classifies how the adaptation is realized and which components or part of the adaptation process is detailed in a paper. The presented *limitations* are also coded as possible attacks on the self-adaptation process. Lastly, an adaptation is classified by its *Degree of Automation* in case this was described in a paper.

The subcategories that relate to the *Integration* of security aspects of the self-adaptive system are mainly derived from the literature along the categorizations presented in Section 2.3. They provide answers to research question RQ1 and include the impact of an attack to a system regarding its confidentiality, integrity, or availability (*CIA* triad, see Section 2.3.2). In addition, the *attack surfaces* and *attack mechanisms* that relate to CAPEC (Section 2.3.1) are considered in the classification scheme. Finally, the used *type of security data*, the *affected part of the adaptation*, and *how the severity of an attack is measured* in the investigated approaches is taken into account in the classification scheme. Regarding the safety aspects, the described *hazard sources and causes* are categorized as well as the *safety quality factors* (see Sections 2.3.3 and 2.3.4). However, *hazard causes* were not classified according to the classification presented by Allouch et al. (2019). This is due to the classification being biased and considering specific hazards that only relate to a specific type of system such as UAVs. As a result, *hazard causes* dimensions were developed using open coding in order to provide the classification of hazards that would be applicable to any domain.

Finally, any approach focusing on the *integration of security and safety* aspects in the context of self-adaptive systems was coded separately. These codes together with the codes under the category *Treatment* address the research question RQ2.

Research question RQ3 is answered by analyzing the codes of the main category *Modeling approach*. These categories were derived in an open coding process and they comprise different *contexts* in which a model is used such as illustration purposes or behavior description, the *objectives* of the analyses performed on the presented models, and the *checked properties* (see Section 2.2). The information about the usage of *models at runtime* and any described *verification and validation techniques* were also collected.

The last research question, RQ4, was answered using the analysis results for the code *Open* which is a subcategory of *Challenges*. The subcategories used emerged during the analysis through open coding. These reflect the mentioned origins of open challenges and encompass: the *System* itself, e.g., through its distribution; the often not fully known *Environment* of a system; the missing joined consideration of *Safety and Security* as well as challenges resulting from the *Adaptation* of a system. Additionally, there exist *Other* origins that cannot be further categorized.

4.4. Classification and analysis

Once the classification scheme was completed, the classification of the final set of publications was conducted according to the following procedure:

1. Coding of each publication according to the proposed classification scheme by one of the authors. The coding was done using the MAXQDA 2020 tool.
2. In the verification process, the final set of papers was divided so that the coding of each publication was verified by at least three other authors. Their comments and results were recorded in the same coding document.
3. Finally, each discrepancy that was identified was discussed by all the authors, until an agreement was reached.

The application of the aforementioned procedure guaranteed that multiple authors were involved in the classification of each paper. The results were documented in a repository,[3] which were used for further analysis, as well as answering the research questions. The obtained results are presented in the following section.

[3] https://zenodo.org/record/6821442

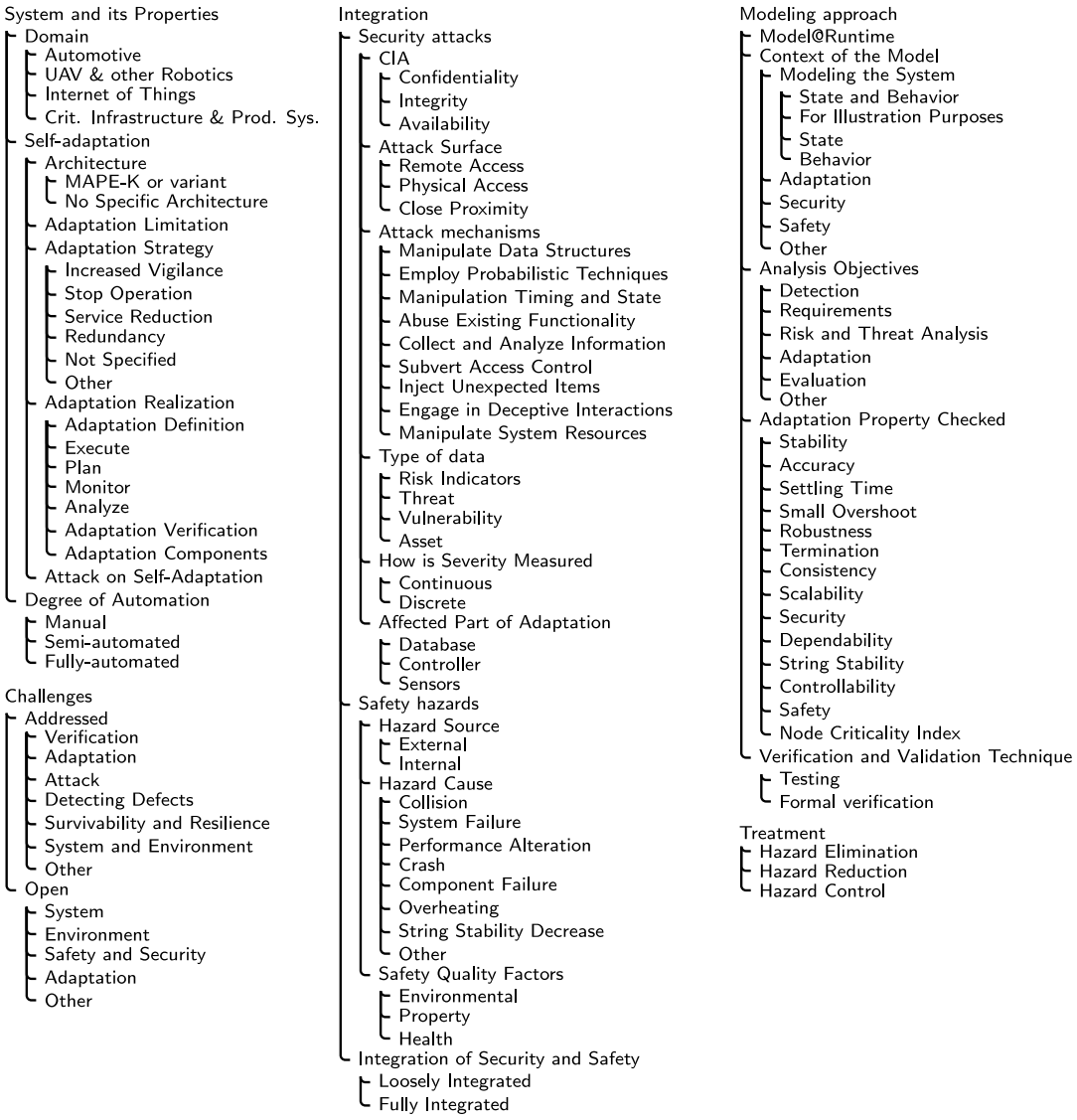

Fig. 3. The proposed classification scheme.

5. Results

The codes applied to the different publications of the final set were analyzed in a qualitative and quantitative way. The results of these analyses are summarized in this section. First, an overview of the domains of the included papers is given (Section 5.1). In Sections 5.2 to 5.5 the analyses of the categories related to the different research questions are presented. At the end of each of these subsections, an answer to the corresponding research question is formulated.

5.1. Overview of included papers

Most papers are from the automotive domain, especially covering autonomous cars and platooning (Table 2). Automotive applications need to focus on safety and increased security to keep the passengers safe (more immediate safety risks than other robotics). Car platoons are dynamic systems of systems (SoS) that can form and dissolve, which immediately implies some relevant characteristics of self-adaptive systems (the cars in the platoon are monitored for misbehavior; the platoon controller reacts to misbehavior by changing the platoon. This forms a meta-control cycle of a self-adaptive system).

Other robotic applications include unmanned aerial vehicles (UAV) (Ferrão et al., 2020; Yoon et al., 2017) as well as cooperative robotics (Beltrame et al., 2018; Causevic et al., 2019). Adaptive safety and security-critical applications also exist in the critical infrastructure and critical production systems domain Settanni et al. (2018a), Liang et al. (2018), Knight and Strunk (2004). These systems often consist of multiple subsystems – possibly deployed in multiple locations – that are connected to an industrial control system (e.g. SCADA). In a similar domain, Settanni et al. (2018b) focuses on workplace safety and security.

Sensor networks using IoT technology to form ad-hoc networks and exchange data adapt the network topology to ensure safe and secure operation in a changing environment (Veledar et al., 2019; Potteiger et al., 2020; Teimourikia and Fugini, 2017; Settanni et al., 2018a). Domain-independent papers in most cases cannot easily focus on safety *and* security in a way to fulfill our inclusion criteria. The motivation to require both safety and security and sufficiently concrete examples showcasing threats or attacks ties the paper to specific domains. Most papers focusing on broader, domain-independent aspects, therefore, did not fulfill the inclusion criteria for safety and security and were filtered out in the process. The two selected, domain-independent papers focus on runtime trust assurance, a narrow, cross-cutting concern, and translating existing ideas inspired by the mammalian

Table 2
Papers per domain (multiple selections possible).

Domain	#Papers	Papers
Automotive	12	Liu et al. (2017), Le and Maple (2019), Veledar et al. (2019), Potteiger et al. (2020), DeBruhl et al. (2015), Zhu et al. (2020), Liang et al. (2018), Petit and Shladover (2015), Monteuuis et al. (2018), Amoozadeh et al. (2015), Bolovinou et al. (2019), Causevic et al. (2019)
IoT	4	Veledar et al. (2019), Potteiger et al. (2020), Teimourikia and Fugini (2017), Settanni et al. (2018a)
Critical Infrastructure & Production Systems	4	Settanni et al. (2018a), Liang et al. (2018), Knight and Strunk (2004), Settanni et al. (2018b)
UAV & other robotics	4	Ferrão et al. (2020), Yoon et al. (2017), Beltrame et al. (2018), Causevic et al. (2019)
No specific domain	2	Schneider et al. (2011), Polack (2010)

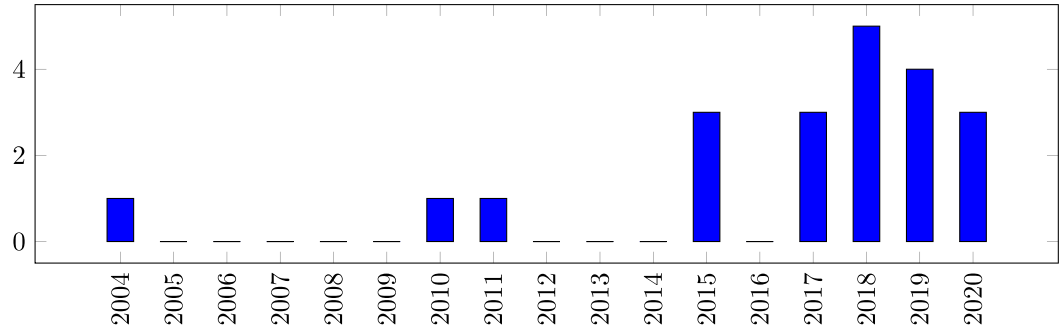

Fig. 4. Selected papers by year of publication.

Table 3
Type of venue for publication of the selected papers.

Type	#Papers	Papers
Conference	8	Le and Maple (2019), Veledar et al. (2019), DeBruhl et al. (2015), Zhu et al. (2020), Liang et al. (2018), Causevic et al. (2019), Settanni et al. (2018a), Yoon et al. (2017)
Journal	5	Potteiger et al. (2020), Petit and Shladover (2015), Amoozadeh et al. (2015), Teimourikia and Fugini (2017), Settanni et al. (2018b)
Workshop	4	Liu et al. (2017), Monteuuis et al. (2018), Beltrame et al. (2018), Polack (2010)
Symposium	3	Ferrão et al. (2020), Bolovinou et al. (2019), Schneider et al. (2011)
Book Chapter	1	Knight and Strunk (2004)

immune response from swarm robotics to other domains *and* providing multiple example use cases from multiple domains.

The intersection of safety, security, and self-adaptation is a recent and active research area. Although we searched for papers published in the past 20 years, most of the selected papers were published in 2015 or later as shown in Fig. 4.

The venues or journals, where these papers were published or presented, correlate with the domains of the papers, covering the respective research areas. The selected papers are mostly conference publications, possibly due to the very recent and novel research area (see Table 3).

5.2. Security, safety and self-adaptation (RQ1)

System adaptation can occur on different levels: from a simple parameter or configuration change to architectural reconfiguration at runtime that modifies the behavior of the system. In the SLR, the focus was on behavioral changes driven by architectural reconfiguration or state changes in the software that modify the behavior. A specialized adaptation architecture is required to trigger, execute and supervise the adaptation.

5.2.1. Adaptation architecture

Most of the 13 papers that specifically mention their self-adaptation mechanism or architecture use *MAPE-K* or a variant of it (see Table 4). Some papers (focusing specifically on platooning) do not mention how the self-adaptation is done or which self-adaptation architecture is used but only the trigger and the goal of the adaptation. Other adaptation architectures that are mentioned are directly tailored toward a very specific application and they are not generally applicable to all self-adaptive systems. Some of these can also be considered unnamed and cannot be identified as MAPE-K. For example, the virtualization architecture in Yoon et al. (2017) is specifically tailored towards UAVs or similar autonomous systems and enables adaptations to defend against specific attacks. Papers that are classified as not mentioning a specific architecture mention an adaptation process or adaptation goal but contain no information on how this goal is reached.

Table 4
Adaptation architectures used in papers.

Architecture	#Papers	Papers
MAPE-K or variant	7	Causevic et al. (2019), Settanni et al. (2018a), Teimourikia and Fugini (2017), Veledar et al. (2019), Settanni et al. (2018b), Knight and Strunk (2004), Le and Maple (2019)
Other architecture	6	Schneider et al. (2011), Beltrame et al. (2018), Zhu et al. (2020), Ferrão et al. (2020), Yoon et al. (2017), Potteiger et al. (2020)
No specific architecture mentioned	8	Liu et al. (2017), DeBruhl et al. (2015), Liang et al. (2018), Petit and Shladover (2015), Monteuuis et al. (2018), Amoozadeh et al. (2015), Bolovinou et al. (2019), Polack (2010)

Table 5
Adaptation strategies (multiple selection possible).

Strategy	#Papers	Papers
Redundancy	8	Monteuuis et al. (2018), Settanni et al. (2018a), Ferrão et al. (2020), Amoozadeh et al. (2015), Le and Maple (2019), Zhu et al. (2020), Potteiger et al. (2020), Knight and Strunk (2004)
Stop Operation	7	Settanni et al. (2018a), Yoon et al. (2017), Liu et al. (2017), DeBruhl et al. (2015), Amoozadeh et al. (2015), Le and Maple (2019), Teimourikia and Fugini (2017)
Reduce Service	5	Polack (2010), Yoon et al. (2017), Zhu et al. (2020), Amoozadeh et al. (2015), Knight and Strunk (2004)
Increased Vigilance	4	Knight and Strunk (2004), DeBruhl et al. (2015), Settanni et al. (2018a), Yoon et al. (2017)
Other	7	Knight and Strunk (2004), Teimourikia and Fugini (2017), Settanni et al. (2018a), Monteuuis et al. (2018), Causevic et al. (2019), Yoon et al. (2017), Potteiger et al. (2020)

5.2.2. Adaptation strategies

While MAPE-K was the only architecture for self-adaptation mentioned and used in multiple papers, the adaptation strategies and goals were more diverse. Four main adaptation strategies are mentioned: *redundancy* (redundant controllers (4), redundant sensors (1), model-based state estimation (3), unspecified (2)), *increasing vigilance* (higher safety margins (1), stricter security policies (1), blocking communication (2)), *reducing service* (degeneracy (1), saving energy (1), removing components (3), minimal operation mode (1)) and *stopping operation* (return to base (1), end cooperation (3), restarting the system (2), fail-stop (1)) (see Table 5). Redundancy is a common safety strategy to detect and recover from failing components. Redundancy can also increase security by adding plausibility checks to detect malicious or manipulated data and components. Redundant sensor measurements can be acquired from similar sensors (Amoozadeh et al., 2015) or by predicting the state using a model (Ferrão et al., 2020; Amoozadeh et al., 2015; Le and Maple, 2019). Multiple controllers are used, for example, to switch to a safe but worse-performing controller if another better-performing but unreliable control strategy (e.g. using trained neural networks) fails (Knight and Strunk, 2004; Potteiger et al., 2020).

Increasing the security or safety level can be another effective adaptation strategy. If an attack or malicious actor is detected, blocking communication and tightening security policies (e.g. by installing additional network traffic monitors (Knight and Strunk, 2004)) can prevent the attacker from compromising other parts of the system. Raising the safety margins can help compensate for longer reaction times due to less (trusted) information and higher uncertainty (DeBruhl et al., 2015).

These adaptations often include service reduction as an intentional or unintentional side effect: the service reduction due to worse-performing redundant sensors is unintentional while blocking communication is an intentional service reduction. Other forms of service reduction such as removing malicious or compromised actors from a vehicle platoon (Amoozadeh et al., 2015) can help stabilize the remainder of the system.

If the system is unable to ensure a secure and safe state or cannot safely recover from a fault or attack, stopping the service is often the best solution. The system can then restart or reform: e.g. after ending cooperation and dissolving a car platoon (DeBruhl et al., 2015; Liu et al., 2017), the platoon can be re-established. Robots may return to base (Yoon et al., 2017), remain stationary (Le and Maple, 2019), or shut down to avoid their loss or destruction.

Apart from these general strategies, adaptations can have very diverse goals. In some cases, instead of reducing the capabilities and access rights of system components at risk, the inverse strategy is used: granting these components additional privileges to be able to recover (Yoon et al., 2017), or scheduling an inspection and informing users of the risk (Teimourikia and Fugini, 2017). In addition to increasing the security level, proposed strategies include the anonymization of communication and messages (Monteuuis et al., 2018), e.g. to prevent the leaking of confidential and personal data. Decryption and checking the integrity of application components before loading them (Potteiger et al., 2020) can ensure, that after resetting the software, a known and secure state is reached. Instead of or in addition to blocking normal communication, establishing a hidden communication channel (Yoon et al., 2017) can help keep or regain control of the system in case of an attack. Concerning the platooning use case, route re-planning (Causevic et al., 2019) instead of dissolving the platoon might be an adequate mitigation strategy for external attacks or avoiding hazardous locations.

It is possible to select one of the multiple strategies depending on the criticality, e.g. switching to or using additional redundant sensors as a first countermeasure and escalating to stopping or reducing service if the critical state is not resolved. Stopping operation is mostly used as a last resort or catch-all in papers

that do not describe the adaptation strategy in detail (Amoozadeh et al., 2015; DeBruhl et al., 2015; Liu et al., 2017; Settanni et al., 2018a; Le and Maple, 2019).

5.2.3. Limitations to self-adaptation

Only a few papers mention conditions under which the success of the self-adaptation cannot be guaranteed (Liu et al., 2017; Potteiger et al., 2020; Zhu et al., 2020; Knight and Strunk, 2004), or checks on the self-adaptation (Causevic et al., 2019). Especially if no specific adaptation architecture is mentioned, the adaptation is implicitly seen as atomic, instantaneous, and guaranteed to succeed, then showing that after the successful adaptation a safe and secure state can be reached.

In Liu et al. (2017), it is mentioned, that the switch from cooperative adaptive cruise control (CACC) to adaptive cruise control (ACC) might not succeed. No method or mechanism to detect or recover from such failure is described. Runtime verification of safety and cyber-security requirements is used in Causevic et al. (2019) to guarantee dependability after and during a reconfiguration. In Knight and Strunk (2004), the authors advise against an ad-hoc implementation of the adaptation mechanism, as it might reduce system dependability even compared to no adaptation at all due to failures during the adaptation process. The operating system and the reconfiguration mechanism must be secure and isolated from potentially compromised components for the adaptation to succeed (Potteiger et al., 2020). Virtualization technology can be used to isolate components from the supervising system (Yoon et al., 2017), but failures in these components or attacks on this isolation are not considered in the papers. Even if succeeding and working correctly, frequent and intermittent reconfigurations (such as switching between different, redundant controllers) can lead to undesired behavior and reduced performance (Zhu et al., 2020). Methods to limit or prevent intentionally induced frequent adaptations are not reported in the papers.

5.2.4. Implementation of the adaptation

Even though the same MAPE-K adaptation architecture is used in most papers, the implementation varies widely or is sometimes only described on a high level. Noteworthy, in Causevic et al. (2019) runtime monitors are used during both normal operation and self-adaptation to assure optimal scheduling of computational resources, as during adaptation additional computational load might occur. Runtime verification during adaptation is used to ensure the dependability of services.

The MAPE-K components run often in a privileged mode compared to the rest of the system. This provides isolation between the system and the MAPE-K components to restrict potentially erroneous or malicious dataflow between the system and the adaptation system (Potteiger et al., 2020; Yoon et al., 2017). The privileged role of the adaptation component allows it to give additional access or rights to other parts of the system as part of the reconfiguration, e.g. creating hidden channels (Yoon et al., 2017), giving some components additional capabilities (Teimourikia and Fugini, 2017; Yoon et al., 2017) or restart (parts) of the system.

In large systems, multiple MAPE loops can be used in a decentralized manner. The MAPE functions can then communicate directly or indirectly via the knowledge base (Settanni et al., 2018b,a). Even in papers that do not specifically use MAPE-K, the self-adaptation mechanism is similarly distributed over multiple components. This helps separate concerns like monitoring, context knowledge, threat analysis, and mitigation assessment (Le and Maple, 2019) or control objectives, adaptation controller, and context monitoring. Computational resources are conserved by triggering and executing later stages of the self-adaptation only if context changes are detected.

5.2.5. Attack mechanisms

In order to identify potential threats to self-adaptive systems, we investigated various attacks that are described in the literature. These attacks were classified according to the Mechanisms of Attack: CAPEC-1000 (The MITRE Corporation, 2021) (see Section 2.3.1), where each attack vector was mapped to a specific category. As a result, a classification of attacks related to self-adaptive systems was developed. This mapping is illustrated in Table 6, in which papers are assigned to specific categories of attack mechanisms. The large majority of attacks belong to *Manipulate System Resources* (56), *Engage in Deceptive Interactions* (34), and *Inject Unexpected Items* (20). In addition, we identified attacks that belong to *Subvert Access Control* (8), *Collect and Analyze Information* (6), and *Abuse Existing Functionality* (2). Finally, there were no attack vectors that *Manipulate Timing and State*, *Employ Probabilistic Techniques*, or *Manipulate Data Structures*.

A *manipulation of system resources* includes attacks in which attackers aim to manipulate single or multiple resources to achieve a certain goal. This is achieved by degrading the availability of critical sensors (Yoon et al., 2017), GPS jamming (Petit and Shladover, 2015) and message replay (Liu et al., 2017) attacks. Attack mechanisms that focus on malicious interactions towards a target with a goal to deceive it by falsifying the content or identity are part of the *Engage in Deceptive Interactions* category. These attacks include sensor and camera spoofing (Le and Maple, 2019), sound interference (Petit and Shladover, 2015), and invisible object (Petit and Shladover, 2015) attacks. The *Inject Unexpected Items* category focuses on attacks that aim to control or disrupt the target, which includes attacks such as message falsification (Amoozadeh et al., 2015), fault injection (Zhu et al., 2020) and miss-report (Liu et al., 2017) attacks. Attacks that try to exploit the weakness that leads to access to the system and various resources can be referred to as *Subvert Access Control* attacks. An example of such attacks include unauthorized access to privacy information (Veledar et al., 2019), personation (Polack, 2010) and man in the middle (Settanni et al., 2018b) attacks. The*Collect and Analyze Information* category deals with attacks that aim to collect and steal various information. Examples of such attacks are location tracking (Petit and Shladover, 2015), information disclosure (Monteuuis et al., 2018), and eavesdropping sensors (Petit and Shladover, 2015) attacks. Finally, in the *Abuse Existing Functionality* an attacker tries to manipulate the functionalities of an application to achieve a malicious goal. Side-channel (Yoon et al., 2017) and blocking pseudonym change (Petit and Shladover, 2015) are examples of such attacks.

In relation to attack mechanisms that specifically focus on the reconfiguration itself, Yoon et al. (2017) discusses that an attacker might forge special, privileged commands that override control of a drone returning to the base. These commands, normally designed to prevent losing control over the drone, can therefore be used against the drone if not carefully chosen and secured.

5.2.6. Attack surfaces and CIA

Aspects that were also addressed from a security point of view include attack surfaces and the CIA triad (see Section 2.3.2). This was done to understand how are the attacks executed as well as to perceive the focus of the attacks. In regards to attack surfaces, attacks can be classified based on the initial entry point, which includes the following three types of access: *physical access* (direct access to wires and control boxes), *close proximity* (includes the attacks that focus on communication with the system such as sensor, audio, and dedicated short-range communication attacks) and *remote access* (involves attacks that are implemented over large distances over the network that utilize GPS, radio, and internet) (Parkinson et al., 2017). As shown in Table 7, a large majority of attacks on self-adaptive systems are

Table 6
Classification of attack vectors using Mechanisms of Attack: CAPEC-1000.

Attack Mechanisms	#Attacks	Papers
Manipulate System Resources	56	Liu et al. (2017), Le and Maple (2019), Knight and Strunk (2004), Settanni et al. (2018b), Zhu et al. (2020), Petit and Shladover (2015), Settanni et al. (2018a), Monteuuis et al. (2018), Amoozadeh et al. (2015), Bolovinou et al. (2019), Yoon et al. (2017)
Engage in Deceptive Interactions	34	Liu et al. (2017), Le and Maple (2019), DeBruhl et al. (2015), Liang et al. (2018), Petit and Shladover (2015), Monteuuis et al. (2018), Amoozadeh et al. (2015), Ferrão et al. (2020), Bolovinou et al. (2019), Yoon et al. (2017), Potteiger et al. (2020)
Inject Unexpected Items	20	Liu et al. (2017), Settanni et al. (2018b), Potteiger et al. (2020), DeBruhl et al. (2015), Zhu et al. (2020), Petit and Shladover (2015), Settanni et al. (2018a), Amoozadeh et al. (2015), Bolovinou et al. (2019), Yoon et al. (2017)
Subvert Access Control	8	Settanni et al. (2018b), Veledar et al. (2019), Petit and Shladover (2015), Settanni et al. (2018a), Monteuuis et al. (2018), Polack (2010)
Collect and Analyze Information	6	Veledar et al. (2019), Petit and Shladover (2015), Monteuuis et al. (2018), Bolovinou et al. (2019)
Abuse Existing Functionality	2	Yoon et al. (2017), Petit and Shladover (2015)
Manipulate Timing and State	0	
Employ Probabilistic Techniques	0	
Manipulate Data Structures	0	

Table 7
Attack Surfaces.

Attack Surfaces	#Papers	Papers
Remote Access	17	Liu et al. (2017), Le and Maple (2019), Knight and Strunk (2004), Settanni et al. (2018b), Veledar et al. (2019), Potteiger et al. (2020), DeBruhl et al. (2015), Zhu et al. (2020), Liang et al. (2018), Petit and Shladover (2015), Settanni et al. (2018a), Monteuuis et al. (2018), Amoozadeh et al. (2015), Polack (2010), Ferrão et al. (2020), Bolovinou et al. (2019), Yoon et al. (2017)
Close Proximity	3	Le and Maple (2019), Petit and Shladover (2015), Amoozadeh et al. (2015)
Physical Access	1	Petit and Shladover (2015)

conducted using *remote access* (17). This is due to multiple attacks that involve Camera/RADAR/LiDAR spoofing/tampering/jamming (e.g., Petit and Shladover (2015), Yoon et al. (2017), Liu et al. (2017)). On the other hand, only a small number of attacks are executed via *close proximity* (3) such as camera tampering (Le and Maple, 2019), blind vision (Petit and Shladover, 2015) and radio jamming (Amoozadeh et al., 2015). Only a single paper addressed attacks that require *physical access* such as removing flash firmware and thermal attacks (Petit and Shladover, 2015). Regarding the information security principles that are affected, *integrity* (16) and *availability* (14) are influenced to the highest degree (see Table 8). At the same time, *confidentiality* (12) is impacted slightly less.

5.2.7. Type of data

The *type of data* dimension (see Table 9) is significant because it provides information regarding what security data is being used in attacks that were described in the investigated set of papers. This makes it possible to better understand the attacker's means and targets. The highest addressed type of security data is *asset* (13). This is due to many approaches describing directly significant assets that are targeted by adversaries. For example, an asset can be a vehicle, infrastructure, or a specific component (e.g., Monteuuis et al. (2018)). On the other hand, *risk* (6) and *threat data* (5) are described to a lesser extent. The possible risks were usually only stated qualitatively (e.g., Amoozadeh et al. (2015)). The biggest surprise is that there is a small number of approaches utilizing *vulnerability* (3) (Amoozadeh et al., 2015; Potteiger et al., 2020; Settanni et al., 2018b) and *exploit* (0) data. It would be expected that this type of data would be included together with the identified attacks to describe how the attack was conducted.

5.2.8. Affected part of adaptation

Another aspect that was investigated is the *affected part of adaptation* (see Table 10). This dimension investigates the components that are affected by attacks, which interact with the adaptation mechanisms. The self-adaptive architectures such as MAPE-K include or interplay with various components such as controllers, sensors, and database. As a result, it is important to identify which of these components are affected by attacks because this can affect the overall adaptation process and cause serious issues. The components that are most often targeted include *sensors* (7 - e.g., Veledar et al. (2019), Liang et al. (2018)) and *controllers* (7 - e.g., Yoon et al. (2017), DeBruhl et al. (2015)). On the other hand, the self-adaptive *database* (1) (Veledar et al.,

Table 8
CIA Triad.

CIA	#Papers	Papers
Integrity	16	Liu et al. (2017), Le and Maple (2019), Schneider et al. (2011), Settanni et al. (2018b), Veledar et al. (2019), DeBruhl et al. (2015), Zhu et al. (2020), Liang et al. (2018), Petit and Shladover (2015), Settanni et al. (2018a), Monteuuis et al. (2018), Polack (2010), Ferrão et al. (2020), Bolovinou et al. (2019), Yoon et al. (2017)
Availability	14	Liu et al. (2017), Le and Maple (2019), Knight and Strunk (2004), Schneider et al. (2011), Settanni et al. (2018b), Veledar et al. (2019), Potteiger et al. (2020), Petit and Shladover (2015), Settanni et al. (2018a), Monteuuis et al. (2018), Amoozadeh et al. (2015), Ferrão et al. (2020), Bolovinou et al. (2019), Yoon et al. (2017)
Confidentiality	12	Liu et al. (2017), Le and Maple (2019), Settanni et al. (2018b), Veledar et al. (2019), Potteiger et al. (2020), Petit and Shladover (2015), Settanni et al. (2018a), Monteuuis et al. (2018), Amoozadeh et al. (2015), Ferrão et al. (2020), Bolovinou et al. (2019), Yoon et al. (2017)

Table 9
Type of Data.

Type of Data	#Papers	Papers
Asset	13	Liu et al. (2017), Le and Maple (2019), Settanni et al. (2018b), Potteiger et al. (2020), DeBruhl et al. (2015), Zhu et al. (2020), Liang et al. (2018), Petit and Shladover (2015), Monteuuis et al. (2018), Amoozadeh et al. (2015), Ferrão et al. (2020), Bolovinou et al. (2019), Yoon et al. (2017)
Risk (indicators)	6	Liu et al. (2017), Settanni et al. (2018b), Veledar et al. (2019), Teimourikia and Fugini (2017), Monteuuis et al. (2018), Amoozadeh et al. (2015)
Threat	5	Le and Maple (2019), Settanni et al. (2018b), Veledar et al. (2019), Petit and Shladover (2015), Monteuuis et al. (2018)
Vulnerability	3	Settanni et al. (2018b), Potteiger et al. (2020), Amoozadeh et al. (2015)
Exploit	0	

Table 10
Affected Part of Adaptation.

Affected Part of Adaptation	#Papers	Papers
Controller	7	Liu et al. (2017), Settanni et al. (2018b), Potteiger et al. (2020), DeBruhl et al. (2015), Zhu et al. (2020), Monteuuis et al. (2018), Yoon et al. (2017)
Sensors	7	Veledar et al. (2019), Le and Maple (2019), Liang et al. (2018), Petit and Shladover (2015), Monteuuis et al. (2018), Bolovinou et al. (2019), Yoon et al. (2017)
Database	1	Veledar et al. (2019)

2019), in which different configurations are stored, presents the least likely target for attackers.

5.2.9. Hazard source and cause

In relation to the safety aspects that were investigated, *hazard source* and *hazard cause* (see Section 2.3.3) play a significant role (see Table 11). According to Allouch et al. (2019), hazard source can be classified as *internal* or *external*. For example, an internal factor can be mechanical or software related, while external can relate to human and environment interaction. Most of the identified hazards take place due to *external* factors (16), while a low number of hazards are realized by *internal* factors (7). External factors mostly include adversaries that aim to cause damage to a system (e.g., Le and Maple (2019), Settanni et al. (2018a), Yoon et al. (2017)), while internal factors are usually related to mechanical or hardware problems (e.g., Teimourikia and Fugini (2017), Ferrão et al. (2020), DeBruhl et al. (2015)). When it comes to the *hazard cause* (see Table 12), this dimension investigates the potential reasons behind why the safety property was affected. It was identified that *collisions* (6 — e.g., Liu et al. (2017), DeBruhl et al. (2015), Liang et al. (2018)) and *system failures* (6 — e.g., Petit and Shladover (2015), Monteuuis et al. (2018), Bolovinou et al. (2019), Settanni et al. (2018a)) play the biggest role because these are the hazards that can potentially create the biggest impact. As a result, they were investigated to the highest degree. Other significant hazard causes include *performance alterations* (5 — e.g., Liu et al. (2017), Le and Maple (2019), Veledar et al. (2019)), *crashes* (4 — e.g., Potteiger et al. (2020), Liang et al. (2018), Petit and Shladover (2015)), *component failures* (3 — (Veledar et al., 2019; Petit and Shladover, 2015; Yoon et al., 2017)), *overheating* (2 — (Settanni et al., 2018a; DeBruhl et al., 2015)), and *string stability decrease* (2 — (Liu et al., 2017; Settanni et al., 2018b)). All the hazard causes that occurred only a single time were placed into the *other* category (5). These included the following: unwanted behavior (Potteiger et al., 2020), faulty navigation, Beltrame et al. (2018), failing to turn (Petit and Shladover, 2015), erroneous response (Teimourikia and Fugini, 2017) and engine failure (Beltrame et al., 2018).

Table 11
Hazard Source.

Hazard Source	#Papers	Papers
External	16	Liu et al. (2017), Le and Maple (2019), Knight and Strunk (2004), Settanni et al. (2018b), Veledar et al. (2019), Potteiger et al. (2020), DeBruhl et al. (2015), Zhu et al. (2020), Liang et al. (2018), Teimourikia and Fugini (2017), Petit and Shladover (2015), Settanni et al. (2018a), Monteuuis et al. (2018), Amoozadeh et al. (2015), Ferrão et al. (2020), Yoon et al. (2017)
Internal	7	DeBruhl et al. (2015), Teimourikia and Fugini (2017), Petit and Shladover (2015), Monteuuis et al. (2018), Amoozadeh et al. (2015), Ferrão et al. (2020), Bolovinou et al. (2019)

5.2.10. Safety quality factors

The last safety dimension that was addressed is *safety quality factors* (see Table 13). This was investigated in order to check what is affected by different hazards that relate to self-adaptive systems (see Section 2.3.4). The two highest affected factors are *property* (11 — e.g., DeBruhl et al. (2015), Bolovinou et al. (2019)) and *health* (10 — e.g., Ferrão et al. (2020), Le and Maple (2019)). On the other hand, the impact on *environmental* factors was described in only three approaches (Settanni et al., 2018b; Teimourikia and Fugini, 2017; Ferrão et al., 2020).

Answer to RQ1: A substantial number of security attacks that target self-adaptive systems was identified. These were investigated and classified according to the Mechanisms of Attack: CAPEC-1000, which resulted in the following mapping of categories and specific attacks: Manipulate System Resources (56), Engage in Deceptive Interactions (34), and Inject Unexpected Items (20), Subvert Access Control (8), Collect and Analyze Information (6), and Abuse Existing Functionality (2). In regards to the specific attacks, these include sensor degradation (Yoon et al., 2017), GPS jamming (Petit and Shladover, 2015), message replay (Liu et al., 2017), sensor and camera spoofing (Le and Maple, 2019), sound interference (Petit and Shladover, 2015), invisible object (Petit and Shladover, 2015), message falsification (Amoozadeh et al., 2015), fault injection (Zhu et al., 2020), miss-report (Liu et al., 2017) attacks, unauthorized access to privacy information (Veledar et al., 2019), personation (Polack, 2010), man in the middle (Settanni et al., 2018b), location tracking (Petit and Shladover, 2015), information disclosure (Monteuuis et al., 2018), eavesdropping sensors (Petit and Shladover, 2015), side-channel (Yoon et al., 2017), and blocking pseudonym change (Petit and Shladover, 2015) attacks. The aforementioned attacks are in most cases executed remotely targeting each property of the CIA triad with a focus on sensors and controller of a self-adaptive system. In regards to the safety hazards, the sources of hazards are to a high degree external and are realized by external attacks targeting property and health quality factors. The results of hazards are in most cases collisions and system failures.

5.3. Combined security and safety approaches and their mitigation strategies (RQ2)

In the following, the approaches that integrate security and safety aspects, as well as their mitigation strategies are presented.

5.3.1. Approaches combining safety and security

A joint consideration of safety and security that also takes into account the interplay of these two aspects, is presented in six papers (Liu et al., 2017; Veledar et al., 2019; Beltrame et al., 2018; Bolovinou et al., 2019; Monteuuis et al., 2018; Yoon et al., 2017). The authors of Liu et al. (2017) present their engineering process which combines safety and security considerations during the development of a system. The process described consists of first defining security and safety goals and then determining which attacks threaten them. For example, it is indicated that collision induction attacks can cause severe accidents by broadcasting an acceleration message, which indicates that they are speeding up, while the attacker starts to brake heavily. Another example is the message falsification attack in which the preceding vehicle that drives with a low speed misreports that it is driving faster. This may lead to a possible collision and life endangerment. As a result, it is very important that a system is resilient to these attacks. Based on these findings, security requirements are then defined for the system to be developed.

The authors of Veledar et al. (2019) propose the usage of Digital Twins to improve the safety and security of IoT and CPS. The authors present a process for designing a corresponding Digital Twin, consisting of identifying the assets as well as the safety and security goals. Subsequently, metrics for the safety and security evaluation are defined and finally, among other things, threat modeling is performed. The authors of Beltrame et al. (2018) propose to extend Buzz code, which is a multi-robot scripting language, by including safety and security primitives. A Pattern Traversal Flow Analysis (PTFA) should then extract a Control Flow Graph from this extended code. The model extracted is then used for model checking. A threat model that calculates the impact of an attack based, among other factors, on the attack's impact on safety is presented in Bolovinou et al. (2019).

The authors of Monteuuis et al. (2018) discuss a risk assessment approach, in which both security and safety are jointly addressed. In addition, they mention possible attacks that could impact the safety of a vehicle. One such attack includes an attacker attempting to affect sensors connected to a vehicle by altering the calibration. This leads to a system error and possible system failure, which could cause a vehicle crash and serious safety issues. Similarly, sensor attacks are also described in Yoon et al. (2017), where an attacker launches an attack in order to degrade the availability of critical sensors impacting overall safety. Furthermore, an attack could also disable the flight controller during the flight, which would cause the system to be in an open-loop state leading to a crash.

5.3.2. Treatment of hazards

An important component of self-adaptive systems is the need to be sure that hazards created from changes in the system or as a result of an external influence have little or no negative impact on the system. To do this, hazards need to be handled in different ways demonstrated in Table 14. Hazard treatment refers to the various annulment approaches that are employed to ensure that derailment from the normalcy of a system does not result in the complete destruction of the system, property damage or total property loss, and human injury or even loss of life. In this classification, the *damage minimization* appears to be the least popular as it was not identified in a single paper.

Table 12
Hazard Cause.

Hazard Cause	#Papers	Papers
Collision	6	Liu et al. (2017), DeBruhl et al. (2015), Liang et al. (2018), Petit and Shladover (2015), Amoozadeh et al. (2015), Ferrão et al. (2020)
System Failure	6	Petit and Shladover (2015), Monteuuis et al. (2018), Bolovinou et al. (2019), Settanni et al. (2018a), Knight and Strunk (2004), Polack (2010)
Performance Alteration	5	Liu et al. (2017), Le and Maple (2019), Veledar et al. (2019), Potteiger et al. (2020), Amoozadeh et al. (2015)
Other	5	Potteiger et al. (2020), Petit and Shladover (2015), Teimourikia and Fugini (2017), Yoon et al. (2017), Beltrame et al. (2018)
Crash	4	Potteiger et al. (2020), Liang et al. (2018), Petit and Shladover (2015), Bolovinou et al. (2019)
Component Failure	3	Veledar et al. (2019), Petit and Shladover (2015), Yoon et al. (2017)
Overheating	2	Settanni et al. (2018a), DeBruhl et al. (2015)
String Stability Decrease	2	Liu et al. (2017), Settanni et al. (2018b)

Table 13
Safety Quality Factors.

Safety Quality Factors	#Papers	Papers
Property	11	Liu et al. (2017), Le and Maple (2019), Settanni et al. (2018b), Potteiger et al. (2020), DeBruhl et al. (2015), Liang et al. (2018), Teimourikia and Fugini (2017), Amoozadeh et al. (2015), Ferrão et al. (2020), Bolovinou et al. (2019), Yoon et al. (2017)
Health	10	Liu et al. (2017), Le and Maple (2019), Potteiger et al. (2020), DeBruhl et al. (2015), Teimourikia and Fugini (2017), Petit and Shladover (2015), Monteuuis et al. (2018), Amoozadeh et al. (2015), Ferrão et al. (2020), Bolovinou et al. (2019)
Environmental	3	Settanni et al. (2018b), Teimourikia and Fugini (2017), Ferrão et al. (2020)

Table 14
How hazards are handled.

Treatment and Mitigation	#Papers	Papers
Hazard Reduction	8	Liu et al. (2017), Le and Maple (2019), Knight and Strunk (2004), Schneider et al. (2011), Veledar et al. (2019), Settanni et al. (2018a), Monteuuis et al. (2018), Yoon et al. (2017)
Hazard Control	7	Le and Maple (2019), Veledar et al. (2019), Zhu et al. (2020), Amoozadeh et al. (2015), Settanni et al. (2018a), Ferrão et al. (2020), Yoon et al. (2017)
Hazard Elimination	6	Liu et al. (2017), Potteiger et al. (2020), DeBruhl et al. (2015), Teimourikia and Fugini (2017), Monteuuis et al. (2018), Settanni et al. (2018a)

The reason for the unpopularity of damage minimization seems to be obvious bearing that the cost of damage may be too dear to bear. Consequently, we see the corresponding highest number for hazard reduction as we aim to reduce how much impact the system gets upon attack from a foreign entity or changes from within the system. Here, a PLC reset action is triggered when the number of security events through a human–machine interface (HMI) exceeds two per minute. The three identified hazard treatment types follow similar concepts as what we have in fail-passive or fault-operational systems (*Hazard Control* — systems that will remain operational during system failure by passing control to another agent), fault-tolerant systems (*Hazard Reduction* — systems that continue to operate by detecting components that are at risk and replacing them before any hazard results) and fail-safe systems (*Hazard Elimination* — systems that switch to safe-mode when the system fails).

From Table 14, eight papers employed hazard reduction techniques to treat hazards. It is evident that most of these studies are in domains that are heavily safety-critical e.g. automotive and autonomous systems as in Liu et al. (2017), Le and Maple (2019), Veledar et al. (2019), Monteuuis et al. (2018), Yoon et al. (2017). In Yoon et al. (2017), a combination of both hazard control and hazard reduction techniques is used. To achieve this, the system switches control over to a secure controller to limit functionalities that are not trustworthy or reliable. Settanni et al. (2018a) studies cyber–physical production systems which are the backbone of modern-day and future industrial systems while (Knight and Strunk, 2004) and Schneider et al. (2011) are more focused on risk management and trust assurance addressing concerns related to survivability and dependability respectively. Both *hazard control* and *hazard elimination* follow closely with hazard reduction at a total of seven and six papers respectively.

Answer to RQ2: According to the results of the SLR, there is a significant relationship between security attacks and safety hazards in the context of self-adaptive systems. Mainly in the automotive domain, this interrelation of safety and security becomes apparent. For example, attacks such as message falsification attacks, message spoofing, message replay, and collision induction attack can lead to collisions and system failures (Liu et al., 2017; DeBruhl et al., 2015; Petit and Shladover, 2015). This is also similar to various types of jamming attacks, which can cause a self-adaptive system to stay or switch to a wrong trajectory leading to a safety hazard (Le and Maple, 2019). Additionally, security attacks such as GPS/radar/lidar spoofing and jamming can also develop traffic disturbance hazards creating a wide range of safety issues (Petit and Shladover, 2015). In order to mitigate hazards that arise as a result of attacks on SAS, it was identified that the most common strategy employed is hazard reduction whereby the ability of a foreign agent to impact the system is greatly reduced as much as possible. It was also noted that there exist instances of combination of approaches as in the case of Le and Maple (2019), Settanni et al. (2018a) and Yoon et al. (2017). First, in order to identify a change in behavior, systems have different adaptation properties (see Table 19) that they check and then are able to tell when a system's behavior has changed within the system. When the system's behavior defers from normalcy, the system is triggered to implement hazard management techniques to deal with potential hazards that may arise. As an example, (Le and Maple, 2019) propose that for a connected automated vehicle (CAV) system, a security profile of roads and their conditions can provide details about the roads' security threats and then exchange safety information with the system such as Basic Safety Messages (BSM).

5.4. Modeling and analysis (RQ3)

In the following, Section 5.4.1 presents results regarding the usage of models in the context of safety and security of self-adaptive systems, while Section 5.4.2 discusses the utilized analysis approaches. Models and analyses are considered jointly, as analyses based on models are increasingly common. This connection does not only exist in the field of model checking but is also used for other analyses. For example, Bucchiarone et al. (2009) used the critical pair analysis of AGG[4] to verify that for every reachable inconsistent configuration of a modeled self-repairing system, there is a repairing rule that leads to a valid configuration again.

5.4.1. Modeling

Through our analysis of the papers, it was identified that models are mainly used to model the system considering adaptation, security, and safety. Table 15 shows in detail the number of papers according to the context in which models are used. This classification was chosen in order to see if there are models that are used jointly in the context of safety, security, and adaptation. In total, out of the 21 publications considered, the three papers (Petit and Shladover, 2015; Ferrão et al., 2020; Yoon et al., 2017) do not use models at all.

One can see in Table 15 that in 12 papers models are used to model the *system* considered. In Potteiger et al. (2020) and Teimourikia and Fugini (2017) models of the system considered were only used for *illustration purposes*. The authors of Potteiger et al. (2020) use an informal model to represent the components of the system and Teimourikia and Fugini (2017) describe the security management process used in their tool using Business Process Model and Notation (BPMN) (Chinosi and Trombetta, 2012). The five papers (Liu et al., 2017; Settanni et al., 2018b; DeBruhl et al., 2015; Settanni et al., 2018a; Liang et al., 2018) present the mathematical models that the authors used to describe the *states and the behavior* of a system. These models are then used as the basis for simulations. The three papers (Amoozadeh et al., 2015; Potteiger et al., 2020; Zhu et al., 2020) also perform simulations, which means that they also have a corresponding mathematical model of their system and its behavior, even if this is not explicitly described in the publications. Besides mathematical models, models are also used to describe the state or behavior of a system.

The modeling of the system *state* was mentioned in Knight and Strunk (2004), Causevic et al. (2019). Each of the two papers mentions using the models at runtime and none of them mentioned a specific type of model used to describe the system state. For example, the authors of both papers only mention using "resource models". With respect to the usage of the models, the authors of Knight and Strunk (2004) describe that they use them to determine the adaptation, and in Causevic et al. (2019) the authors plan to use them for analyses.

Modeling the *behavior* of a system was reported in four papers. In Knight and Strunk (2004) finite state machines are used to detect erroneous states in the system by modeling the behavior of the system in case of a fault. It should be noted here that this is a different model than the previously mentioned model from (Knight and Strunk, 2004) that is used to model the system *state*. The difference is that the previously mentioned model was used to model the application state to determine possible adaptations. Embedded in their mathematical model, the authors of DeBruhl et al. (2015) use models at runtime of the behavior of vehicles in order to detect misbehavior. The authors of Beltrame et al. (2018) generate a control flow graph out of safety and security code primitives, which is later transformed and used as input for model checking. In Causevic et al. (2019), the authors only mention that they plan to develop "run-time behavioral models" for adaptive systems.

As shown in Table 15, two papers use models in the context of *adaptation*. One paper (Knight and Strunk, 2004) describes the use of a distributed workflow model that describes the intentions of an adaptation request, the temporal order needed for its execution, and its resource usage. The vision paper (Causevic et al., 2019) only states that the authors plan to develop models to take into account the adaptive characteristics of a system.

In the context of *security*, six papers use models. Veledar et al. (2019) uses attack trees and security metrics in their presented validation method. In addition to the aforementioned paper, Le and Maple (2019) utilizes attack trees as well. The authors model the attacker by taking into account the attacker's knowledge of the system, expertise, equipment, opportunities, and time. Compared to Le and Maple (2019), the authors of Monteuuis et al. (2018) apply attack trees to model an attacker only on the basis of the attacker's knowledge of the system, expertise, and equipment. In addition, they present their developed threat model (STRIDELC) and security goal model (AINCAAUT). Bolovinou et al. (2019) presents a developed threat model named TARA+. Here, attackers are modeled using the same properties as in Le and Maple (2019), but time is not taken into account. In DeBruhl et al. (2015), attacks are modeled by the values of the coefficients in the mathematical model of the system that was used. In order to detect coordinated security attacks that try to attack the system with multiple vulnerabilities, (Knight and Strunk, 2004) proposes the usage of a finite-state-machine hierarchy.

Table 15 shows that in three papers models are used in the context of *safety*. The authors of Teimourikia and Fugini (2017)

[4] https://www.user.tu-berlin.de/o.runge/agg/index.html

Table 15
Context in which models are used.

Context	#Papers	Papers
System	12	Potteiger et al. (2020), Teimourikia and Fugini (2017), Liu et al. (2017), Settanni et al. (2018b), DeBruhl et al. (2015), Settanni et al. (2018a), Liang et al. (2018), Amoozadeh et al. (2015), Zhu et al. (2020), Knight and Strunk (2004), Causevic et al. (2019), Beltrame et al. (2018)
Security	6	Veledar et al. (2019), Le and Maple (2019), Monteuuis et al. (2018), Bolovinou et al. (2019), DeBruhl et al. (2015), Knight and Strunk (2004)
Safety	3	Teimourikia and Fugini (2017), Schneider et al. (2011), Zhu et al. (2020)
Adaptation	2	Knight and Strunk (2004), Causevic et al. (2019)

Table 16
Objectives of the analyses.

Objective	#Papers	Papers
Evaluation	8	Liu et al. (2017), Settanni et al. (2018b), DeBruhl et al. (2015), Settanni et al. (2018a), Liang et al. (2018), Amoozadeh et al. (2015), Potteiger et al. (2020), Zhu et al. (2020)
Detection	5	Yoon et al. (2017), Settanni et al. (2018b), DeBruhl et al. (2015), Liang et al. (2018), Settanni et al. (2018a)
Risk & Threat	4	Teimourikia and Fugini (2017), Monteuuis et al. (2018), Veledar et al. (2019), Le and Maple (2019)
Requirements	4	Liu et al. (2017), Schneider et al. (2011), Yoon et al. (2017), Causevic et al. (2019)
Adaptation	1	Knight and Strunk (2004)
Other	3	Veledar et al. (2019), Beltrame et al. (2018), Zhu et al. (2020)

present a safety ontology that allows the creation of a safety model. This model is intended to capture safety expertise and enable risk management. (Schneider et al., 2011) presents ConSert, which describes a set of alternative safety guarantees in the form of boolean expressions. These expressions are then represented in the form of Binary Decision Diagrams (BDD) and used at runtime to realize dynamic negotiation of safety contracts. In order to specify an initial set of states, the authors of Zhu et al. (2020) use a directed graph in their analysis.

There were no identified papers in the context of joint modeling of safety and security of self-adaptive systems. This fact can also be seen in Table 15, as there is no paper that uses models jointly in the context of *safety*, *security*, and *adaptation*. The closest to an approach that provides joint modeling of safety and security is the threat model presented in Bolovinou et al. (2019). The authors suggest calculating the impact of an attack based on security metrics. In the steps mentioned in Veledar et al. (2019) to design a digital twin, safety and security goals and their modeling are also considered, but without further details. Also in the vision paper (Causevic et al., 2019), the authors mention models for joint consideration of safety and security at runtime. Joint consideration of these two aspects is currently only realized in the form of approaches that attempt to combine safety and security, such as in Liu et al. (2017), Veledar et al. (2019), Beltrame et al. (2018), Bolovinou et al. (2019) (cf. Section 5.3.1).

5.4.2. Analysis

Models are often used as a basis for analysis. In the following, we describe in more detail the identified analysis methods.

The analyses mentioned in the papers are very heterogeneous and use a wide variety of data as input, so we decided to classify the analyses according to their objective. This also allows us to see what analyses are used for in the selected set of papers.

Table 16 shows the number of papers in which an analysis is mentioned and categorized by the objectives of the analysis. One can see that analyses are mainly used for *evaluation* purposes, more precisely models of this category are used in the context of assessing an approach presented in a paper. All eight of the 21 included papers use simulations based on a mathematical model to evaluate their presented approaches, namely (Liu et al., 2017; Settanni et al., 2018b; DeBruhl et al., 2015; Settanni et al., 2018a; Liang et al., 2018; Amoozadeh et al., 2015; Potteiger et al., 2020; Zhu et al., 2020).

In terms of analysis related to *adaptation*, (Knight and Strunk, 2004) mentions that the state needs to be analyzed to determine to which nodes adaptation commands should be sent to.

As shown in Table 16, *risk and threat* analyses are mentioned by four papers, whereas risk assessment analyses are mentioned in three papers, namely (Teimourikia and Fugini, 2017; Monteuuis et al., 2018; Veledar et al., 2019). In addition to the risk assessment, (Veledar et al., 2019) also mentions threat modeling and forecasting. The authors of Veledar et al. (2019) describe the evaluation of risks and threats using metrics and security models and the authors of Le and Maple (2019) mention a threat agent analysis.

Four papers mention analyses that are used to check whether *requirements* are met. The papers (Liu et al., 2017; Schneider et al., 2011; Yoon et al., 2017) mention analyses that check whether safety requirements are met or not. The vision paper (Causevic et al., 2019) also mentions analyses to check safety and security requirements at design time and at runtime.

In addition to checking requirements, some papers also describe analyses to *detect* anomalies, e.g. caused by an attack. As shown in Table 16, five papers described such analysis (Yoon et al., 2017; Settanni et al., 2018b; DeBruhl et al., 2015; Liang et al., 2018; Settanni et al., 2018a). In Yoon et al. (2017), authors fetch the data from a trusted environment, which allows them to rely on the authenticity of the data during analysis. Based on that condition, the authors state that they can identify attacks that

Table 17
Overview of the models used and the related analyses.

Context	Model Language	Analysis Objectives	Paper
System	Math. formula	Evaluation	Potteiger et al. (2020)
System	Math. formula	Evaluation	Liu et al. (2017)
System	Math. formula	Evaluation	Zhu et al. (2020)
System	Math. formula	Evaluation	Amoozadeh et al. (2015)
System	Math. formula	Evaluation	Settanni et al. (2018a)
System	Math. formula	Evaluation	Settanni et al. (2018b)
System	Math. formula	Evaluation, Detection	DeBruhl et al. (2015)
System	Math. formula	Evaluation, Detection	Liang et al. (2018)
System - Illustration	None specific		Potteiger et al. (2020)
System - Illustration	BPMN		Teimourikia and Fugini (2017)
System - State	None specific (“resource models”)	Requirements	Causevic et al. (2019)
System - State	None specific (“resource models”)		Knight and Strunk (2004)
System - Behavior	Finite-state-machine	Adaptation	Knight and Strunk (2004)
System - Behavior	Math. formula	Evaluation, Detection	DeBruhl et al. (2015)
System - Behavior	Control Flow Graph	Other	Beltrame et al. (2018)
System - Behavior	None specific (“run-time behavioral models”)	Requirements	Causevic et al. (2019)
Adaptation	None specific	Requirements	Causevic et al. (2019)
Adaptation	None specific (“workflow model”)		Knight and Strunk (2004)
Safety	Directed graph	Other	Zhu et al. (2020)
Safety	BDD	Requirements	Schneider et al. (2011)
Safety	Own safety model	Risk and threat analysis	Teimourikia and Fugini (2017)
Security	Math. formula	Evaluation, Detection	DeBruhl et al. (2015)
Security	Attack Trees	Risk and threat analysis	Veledar et al. (2019)
Security	Attack Trees, own threat model and own security goal model	Risk and threat analysis	Monteuuis et al. (2018)
Security	Attack Trees	Risk and threat analysis	Le and Maple (2019)
Security	Finite-state-machine hierarchy		Knight and Strunk (2004)
Security	Own threat model		Bolovinou et al. (2019)

set e.g. wrong control parameters. The security threat detection analysis described in Settanni et al. (2018b) consists of analyzing monitored data and security metrics to identify system events that indicate possible threats. In addition, previously collected external security information is also taken into account during the analysis. In Settanni et al. (2018a), AMiner is used to check the collected log data for anomalies using a white list, and in Liang et al. (2018) the authors use the Principal Component Analysis, which adapts a Gaussian model to the data that represents normal behavior. An anomaly is then detected in case of deviations. In the analysis described in DeBruhl et al. (2015), each vehicle in a platoon models the expected behavior of its predecessor. If the actual measured behavior deviates from the modeled behavior, this is recognized as an anomaly.

Three papers mention further analysis types, in which (Veledar et al., 2019) uses asset modeling and Beltrame et al. (2018), Zhu et al. (2020) describe various analyses for their presented approach. The authors of Zhu et al. (2020) present an analysis to check whether a system enters pre-specified unsafe states. During the analysis, weakly-hard constraints are taken into account which are related to different functional properties, i.e. safety. The analysis described in Beltrame et al. (2018) is performed with the help of a Pattern Traversal Flow Analysis. This is achieved by extracting primitives in the program code based on security and safety aspects. The result is a Control Flow Graph which is then transformed and used as input for a model checker.

In Table 17, the different models are sorted by their modeling context and related to the objectives of the analyses which work on these models. One can clearly see that mostly mathematical formulas were used to model the systems and then used for evaluation. This is due to the fact that mainly simulations were used for evaluation, which requires a description of the system as a mathematical model. For the risk and threat analysis, as shown in Table 17, Attack Trees are used quite often, but overall the modeling languages used are very different, e.g. specially developed models, finite state machines, or Binary Decision Diagrams (BDD) are used. Often the authors make only very vague statements about the model languages used, so frequently no language is mentioned or only general terms, such as “resource model”, are used to describe the models used.

In the following, the analysis types of SASs with respect to the objective of checking conformance to requirements are further explored. This relates to V&V and has been described earlier in Sections 2 and 3. Even though models are common for this purpose, they are not the only assurance strategies especially when related to safety and security as concerned as in (Liu et al., 2017; Schneider et al., 2011; Yoon et al., 2017; Causevic et al., 2019). In the following paragraphs, this process is further described.

Table 18 shows the distribution of the different approaches collected from our studies.

There are two main V&V approaches, i.e. formal verification *(i.e. relying on the mathematical specification of systems using temporal logic and its variants or some other tools and frameworks)*, and testing *which largely relates to simulations, proof of concepts, use of testbeds and hardware-in-the-loop (HITL) testing*. For instance, for the latter, Veledar et al. (2019) uses a digital twin for validation in an autonomous driving setting. The study also uses a multi-metric security approach to validate both security and safety.

Testing is observed to be more common. One can see that 12 papers were identified to have used different testing techniques in verifying the performance of their proposed system. In particular, the most popular technique is the use of simulators in virtual environments as in most of the papers or frameworks e.g. a virtual drone framework demonstrated in Yoon et al. (2017) or HITL testing as previously mentioned in Veledar et al. (2019). At the same rate, in Amoozadeh et al. (2015), the authors make use of an integrated simulation platform called VENTOS, which combines SUMO (Simulation of Urban Mobility)[5] and OMNET++[6] which is used in wireless communication simulation.

At the same rate, across all 21 papers, a total of 13 properties were identified from open coding as well as from existing literature. It was observed that four papers check for security and safety respectively. Yoon et al. (2017) and Veledar et al. (2019) both check for security and safety while (Settanni et al., 2018a; Schneider et al., 2011) and (Liang et al., 2018; Zhu et al., 2020)

5 https://sumo.dlr.de/docs/index.html
6 https://omnetpp.org/

Table 18
Verification and Validation Approaches (overlapping papers may apply)

V & V Approach	#Papers	Papers
Testing	12	Liu et al. (2017), Settanni et al. (2018b), Veledar et al. (2019), Potteiger et al. (2020), DeBruhl et al. (2015), Zhu et al. (2020), Liang et al. (2018), Settanni et al. (2018a), Amoozadeh et al. (2015), Polack (2010), Ferrão et al. (2020), Yoon et al. (2017)
Formal Verification	4	Schneider et al. (2011), Beltrame et al. (2018), Zhu et al. (2020), Causevic et al. (2019)

Table 19
Different adaptation property checked.

Adaptation Property	#Papers	Papers
Security	4	Yoon et al. (2017), Veledar et al. (2019), Settanni et al. (2018a), Schneider et al. (2011)
Safety	4	Yoon et al. (2017), Veledar et al. (2019), Liang et al. (2018), Zhu et al. (2020)
Stability	2	Zhu et al. (2020), Amoozadeh et al. (2015)
String Stability	2	DeBruhl et al. (2015), Liu et al. (2017)
Controllability	2	Veledar et al. (2019), Monteuuis et al. (2018)
Robustness	1	Yoon et al. (2017)
Node Criticality Index	1	Ferrão et al. (2020)
Termination	1	Settanni et al. (2018a)
Consistency	1	Settanni et al. (2018a)
Dependability	1	Schneider et al. (2011)
Scalability	1	Monteuuis et al. (2018)
Accuracy	1	Zhu et al. (2020)
Settling time	1	Potteiger et al. (2020)

check separately for security and safety properties respectively. There were no indications of the properties checked in nine papers and those that were found to contain indications are reported in Table 19.

Properties that arose from open coding include *Node Criticality Index (NCI)*, *String Stability* and *Controllability*. *NCI* is a property that combines network quality of service, safety, and security and provides useful information for decision (Ferrão et al., 2020). Controllability, on the other hand, refers to how the system controls are configured per time, either at design time or run time. This behavior was observed in both (Veledar et al., 2019) and (Monteuuis et al., 2018). Furthermore, *scalability* is mentioned in Monteuuis et al. (2018) and describes how attack goals can be scaled across multiple vehicles e.g. single or multiple attacks. Another commonly identified property is *stability* in Zhu et al. (2020), Amoozadeh et al. (2015) which is not to be mistaken for string stability, in which a group of vehicles (in a platoon) tend to maintain a certain distance among each other, achieved through obeying a constant control law. *Stability* which may also be referred to as *Local stability* describes how the effect of perturbation reduces with time in the following vehicle behind another in a vehicle platoon. One property that was identified in previous studies is *small overshoot* but was not observed in any of our final paper selections. Suffice it to mention that *Dependability* is a notion of trust and combines many attributes such as availability, reliability, safety, integrity, and maintainability and may include confidentiality when security is in question (Avizienis et al., 2004). However, for the purpose of our work, we have decided to treat Dependability as a whole since we are unable to draw the isolated impact of each of the attributes on the system.

Answer to RQ3: The results show that mainly the system and its behavior are modeled using mathematical models. These models are then used for simulations, e.g. to analyze the response of a system under attack. Modeling in the context of adaptation is mentioned in only three publications, namely (Knight and Strunk, 2004; Veledar et al., 2019; Causevic et al., 2019). Two of these papers describe the usage of models at runtime for detection, reasoning, and adaptation.+ For security modeling mainly attack trees are used and for safety modeling, different models are used, depending on the purpose, e.g. Binary Decision Diagrams to model safety guarantees. As already mentioned, simulations are mainly performed for analysis. Otherwise, risk and threat analyses are performed in most cases, as well as analyses to detect anomalies or to check requirements.

5.5. Open challenges (RQ4)

In the following, the *open challenges* in the context of safety and security of self-adaptive systems mentioned in the selected set of papers are summarized and discussed.

Table 20
Number of papers mentioning open challenges categorized by the origin of the challenge.

Origin of open Challenges	#Papers	Papers
Adaptation	4	Settanni et al. (2018b), Causevic et al. (2019), Beltrame et al. (2018), Polack (2010)
Environment	3	Teimourikia and Fugini (2017), Causevic et al. (2019), Beltrame et al. (2018)
System	2	Causevic et al. (2019), Beltrame et al. (2018)
Safety & Security	2	Causevic et al. (2019), Teimourikia and Fugini (2017)
Other	1	Liang et al. (2018)

The open challenges mentioned in the publications can be divided into five classes: *adaptation*, *environment*, *system*, combined consideration of *safety and security*, and *other*. Each of these classes describes the origin of a challenge.

Table 20 shows for each of these categories how many papers belong to each class.

In total four papers mention challenges resulting from the *adaptation* of a system. The authors of Settanni et al. (2018b) mention, that a major issue for running self-adaptive systems, which have to meet certain safety requirements is that after any adaptation, the safety certification of the system becomes invalid. The paper (Causevic et al., 2019) notes that the currently used safety assurance or verification methods are not applicable to self-adaptive systems, since these methods require detailed knowledge about the system and its environment. Similar challenges are also mentioned in Beltrame et al. (2018) and Causevic et al. (2019) regarding the validation of the emergent behavior of a system at runtime. Other challenges include the required additional processing overhead and memory, the fragility to unexpected changes, and the complex design of a self-adaptive system, which are mentioned in Polack (2010).

The *system* itself also leads to new challenges. The increasing connectivity of self-adaptive systems results in more vulnerabilities, that can be exploited for attacks (Causevic et al., 2019). Additionally, systems must also be able to cope with the loss of messages. Challenges arising from swarm applications are the unknown global system state, which forces robots to rely only on local information, and the unpredictability of the resulting group behavior (Beltrame et al., 2018).

The three papers (Teimourikia and Fugini, 2017; Causevic et al., 2019; Beltrame et al., 2018) mention challenges that result from the *environment* of a system. The cooperation of people with complex systems, for example at a workplace, leads to new safety and security requirements that must be taken into account (Teimourikia and Fugini, 2017). In Beltrame et al. (2018) the authors mention not only the interaction of the system with the environment and its influence on the system but also the fact that the environment can only be partially observed. A continuous change of the environment is also possible, to which the system must adapt in terms of safety and security, even under uncertainty. This, in turn, can lead to the rise of new vulnerabilities that need to be taken into account (Causevic et al., 2019).

As seen in Table 20, two papers mention challenges in combining *safety and security*. Thus, the authors of Causevic et al. (2019) mention a combined consideration of these two aspects of a system as an open challenge. The paper (Teimourikia and Fugini, 2017) is a bit more detailed and states that a balanced consideration of both aspects of a system without compromising each other is a challenge.

Another challenge mentioned by the authors of Liang et al. (2018) is related to secure communication. The authors of Liang et al. (2018) mention the need for methods to prove the security of cryptographic constructions, for example, to ensure secure communication.

Answer to RQ4: The results of the analysis show that the adaptation of a system is the main cause of new challenges. In particular, the focus is on the assurance of safety requirements and the expected behavior of the system. The system environment is also mentioned as an origin of new challenges. Because of the increasing interaction of self-adaptive systems with their environment, more safety and security problems+ can occur that were not fully known during the design of the system. Further challenges occur due to the often distributed system design of self-adaptive systems, which results in more vulnerabilities. In addition, balanced joint consideration of safety and security aspects of self-adaptive systems is mentioned as an open challenge.

6. Discussion

In the following section, we present the key findings and possible threats to validity related to this study.

6.1. Key findings

The high number of papers from the automotive domain shows that applications like autonomous cars and platooning are in need of a joint analysis and modeling approach for their innate safety and security criticality. Emerging technologies, such as the increasing computerization of these systems, along with their growing complexity and connectivity, make a component-based and self-adaptive architecture necessary. Therefore, applications in the automotive domain are likely drivers for future standardization and full integration of safety, security, and self-adaptation. Current approaches, however, fall short: the adaptation process is seldom fully fleshed out and incomplete. Risks and vulnerabilities of the adaptation system are often not considered or left unaddressed due to naive assumptions such as reconfigurations always succeeding or happening instantaneously. As the lack of modeling and analysis of the adaptation itself shows (Tables 15, 16), the need for modeling the adaptation, accompanying modeling tools, the additional complexity, and the increased attack surface of a self-adaptive system are mostly ignored. While MAPE-K can be seen as the greatest common denominator of many of the examined approaches (7 papers mention using MAPE-K or clearly use a similar architecture; Table 4), the implementation of the adaptation system is currently not standardized and often incompletely described or formalized (eight papers do not mention a specific adaptation architecture used or do not describe it at all; Table 4). Only two papers use an adaptation model; Table 15).

Our findings also indicate that attacks that specifically target self-adaptive systems and their respective architectures are not addressed to a high degree. Most of the security attacks mentioned in Section 2.3 could also be applied to other systems. However, some attacks that specifically target self-adaptiveness were still identified. These attacks are almost always realized by

purposely targeting sensors in order to feed false data to the system. In most cases, the aim of these attacks is to affect the controller so the system would switch to the wrong configuration during the reconfiguration phase. This can have serious consequences because the device that uses the system could crash and cause serious damage to the surroundings. Moreover, these attacks mostly target the integrity and availability of a system, while confidentiality is affected to a smaller degree. This directly relates to the attacks that target self-adaptiveness because an attacker usually attempts to disrupt the system and not read or steal confidential data.

In regards to the combination of security and safety, results show that there is a lack of approaches that utilize both of these at the same time. This can be seen from the total number of papers that were collected, as well as that only six papers considered them jointly to a higher degree (cf. Section 5.3.1). This is very significant for self-adaptive systems because they are considered hazardous technology and are prone to both security and safety aspects (Ismail et al., 2021). In the past, the problem of differentiating between security and safety could be addressed by establishing the difference between a criminal offense and an accident (Bieder and Pettersen Gould, 2020). However, due to the digitalization and transformation of security and safety policies, as well as the increase in the complexity of risk management solutions, it is evident that joint consideration cannot be overlooked anymore (Plósz et al., 2017).

SAS increases the problem, not only do these systems often operate in a dynamic environment that is not fully known at the time of design (cf. Section 5.5). On the other hand, each adaptation of the system must be re-certified with respect to safety. This is aggravated by the fact that not all possible adaptations may be known at the design time of the system (cf. Section 5.5). Furthermore, a SAS offers new attack surfaces while it performs an adaptation. A combined modeling approach for the safety and security of self-adaptive systems, which also includes possible threats during adaptation and the resulting consequences, is missing (cf. Section 5.5). However, these points must be taken into account in order to ensure the safety and security of a SAS.

With respect to verification and validation, we have observed that not a lot of work has been done specifically focusing on a combined security and safety V&V (assurance) of self-adaptive systems. In our study, only (Ferrão et al., 2020) mentions the Node Criticality Index, which is a metric that combines the quality of service, security, and safety. By extension, the extrapolation of classification taxonomy in this regard remains a challenge. Most of the activities under V&V are carried out as part of the analysis of a system and for the purpose of evaluating the proposed system for conformance with requirements and certain goals. We have observed that the most common verification and validation approach is simulation testing. This includes the use of standalone simulation systems, model-in-the-loop (MIL) tests, software-in-the-loop (SITL) tests, and hardware-in-the-loop (HITL) tests.

6.2. Threats to validity

The possible threats to validity were taken into account while conducting the SLR. These threats were also minimized as much as possible. According to (Petersen and Gencel, 2013), we differentiated between descriptive validity, theoretical validity, generalizability, interpretive validity, and repeatability.

Descriptive validity: There is a threat that the classification scheme does not address the stated research goal. In order to tackle this issue, the research questions were developed as an initial step. This allowed the clear formulation of each dimension in the classification scheme and classification of all the papers from the final set. Additionally, many of the dimensions were extracted systematically from existing literature.

Theoretical validity: As an initial step in the publication search process, a keyword search was conducted. This included the search in multiple publication databases. As a result, a large set of relevant publications was obtained. However, there is always a chance that some key papers were missed using the aforementioned methodology. The vastly different numbers of search results between search engines might indicate that the search string excludes relevant papers in some search engines (e.g. through different semantics of operators or options in these search engines). This threat to validity was reduced by applying a forward and backward snowballing iteration to find these papers missed by the database search. However, this snowballing strongly depends on the connectivity of the citation graph and might fail to include papers not directly related to the papers found through the database search. Additionally, there was a chance that there could be a potential bias regarding the selection and extraction processes. This was countered by applying the cross-validation approach by which each paper was classified and checked by multiple authors. In case of a difference in opinion, each issue was thoroughly discussed and resolved among the authors. It is also significant to consider the researcher's bias during the classification of papers. This is strongly related to the understanding of self-adaptive architectures and related processes. We addressed this in the same way as the bias in the selection and extraction processes.

Generalizability: It is important to note that this study has a substantial focus on self-adaptive systems and their relationship with security and safety aspects. This means that the results are not applicable to any other domain. However, parts of the classification scheme could be used within other domains related to cyber–physical systems. Besides this, the study has a strong scientific focus. This means that only evaluated research work was considered, but for instance, gray literature was not.

Interpretive validity: In order to avoid results or part of the results being interpreted according to a personal point of view of a specific author, it was made sure that all results were interpreted by every author. This means that all the disagreements were profoundly discussed and solved. In addition, the MAXQDA 2020 tool was used, which allowed rigid tracking of findings, transparent classification, and thorough analysis of papers.

Repeatability: The replication of the conducted SLR represents another possible threat to validity. This is addressed by providing a detailed methodology, which was documented to the smallest detail. This is shown in the methodology section (see Section 4). Furthermore, we applied the existing guidelines according to Wohlin (2014), Usman et al. (2017) and Petersen and Gencel (2013). Similarly, the process of classifying publications can be executed by employing the proposed classification scheme. As a result, the replication of the study and its procedure should be possible.

7. Conclusion

We conducted a systematic literature review to analyze how well the safety and security of self-adaptive systems are considered in current research. For this purpose, we assessed 2.832 found publications using our inclusion and exclusion criteria and classified the resulting 21 remaining publications using our coding scheme.

The results show that the correlations between security attacks and safety hazards are already addressed in the literature. However, their interplay is not sufficiently considered in the design or analysis of a system. Some approaches attempted to combine safety and security, but there is a lack of a tighter

coupling of the two aspects in order to reliably model and analyze their interaction. In particular, the consideration of safety and security requirements during the adaptation of a system is completely neglected in the literature. Various investigated analyses focus on ensuring that the system resulting from an adaptation meets safety and/or security requirements. The fact that these requirements could also be violated during the adaptation is not considered. It is often (implicitly) assumed that adaptation is performed in zero-time and in the form of a transaction. However, this does not reflect reality, which means that severe vulnerabilities of a system could easily be missed.

As a part of future work, it is necessary to focus on reducing the gap between safety and security analysis. In particular, the focus needs to be on self-adaptation to be able to fulfill the necessary safety and/or security requirements during the adaptation process.

CRediT authorship contribution statement

Irdin Pekaric: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Raffaela Groner:** Conceptualization, Investigation, Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Thomas Witte:** Conceptualization, Investigation, Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Jubril Gbolahan Adigun:** Conceptualization, Investigation, Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Alexander Raschke:** Methodology, Validation, Writing – original draft, Writing – review & editing, Visualization. **Michael Felderer:** Conceptualization, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Matthias Tichy:** Conceptualization, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have added the link to Zenodo repository in the paper.

Acknowledgments

This work was partially supported by the Austrian Science Fund (FWF): I 4701-N and German Research Foundation (DFG): 435878599.

Appendix. Selected papers detailed table

To give an overview over the selected papers and their coding, we condensed the tables from the result section into the following overview table (Table A.21).

Table A.21
Comprehensive Table of selected papers and their coded properties.

	Liu et al. (2017)	Le and Maple (2019)	Veledar et al. (2019)	Potteiger et al. (2020)	DeBruhl et al. (2015)	Zhu et al. (2020)	Liang et al. (2018)	Petit and Shladover (2015)	Monteuuis et al. (2018)	Amoozadeh et al. (2015)	Bolovinou et al. (2019)	Causevic et al. (2019)	Teimourikia and Fugini (2017)	Settanni et al. (2018a)	Knight and Strunk (2004)	Settanni et al. (2018b)	Ferrão et al. (2020)	Yoon et al. (2017)	Beltrame et al. (2018)	Schneider et al. (2011)	Polack (2010)
Domain																					
Automotive	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓									
IoT			✓	✓									✓	✓							
C. Infrastructure & Prod. Systems							✓							✓	✓	✓					
UAV & other robotics												✓					✓	✓	✓		
No specific domain																				✓	✓
Adaptation Architecture																					
MAPE-K or variant		✓	✓									✓	✓	✓	✓	✓	✓	✓	✓		
Other architecture				✓		✓															
No specific architecture mentioned	✓				✓		✓	✓	✓	✓	✓										✓
Adaptation strategies																					
Redundancy		✓		✓		✓			✓	✓				✓	✓		✓				
Stop Operation	✓	✓			✓					✓			✓	✓				✓			
Reduce Service						✓				✓					✓			✓			✓
Increased Vigilance					✓									✓	✓			✓			
Other				✓					✓			✓	✓	✓	✓			✓			
Attacks by Attack Mechanisms																					
Manipulate System Resources	✓	✓				✓		✓	✓	✓	✓			✓	✓	✓		✓			
Engage in Deceptive Interactions	✓	✓		✓	✓		✓	✓	✓	✓	✓						✓	✓			
Inject Unexpected Items	✓			✓	✓	✓		✓		✓	✓			✓		✓		✓			
Subvert Access Control			✓					✓	✓					✓		✓					✓
Collect and Analyze Information			✓					✓	✓		✓										
Abuse Existing Functionality								✓										✓			
Attacks by Attack Surfaces																					
Remote Access	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓			✓
Close Proximity		✓						✓		✓											
Physical Access								✓													

(*continued on next page*)

Table A.21 (*continued*).

Attacks by CIA Triad																				
Integrity	✓	✓	✓		✓	✓	✓	✓	✓		✓			✓		✓	✓	✓		✓	✓
Availability	✓	✓	✓	✓				✓	✓	✓	✓			✓	✓	✓	✓	✓		✓	
Confidentiality	✓	✓	✓	✓				✓	✓	✓	✓			✓		✓	✓	✓			
Type of Data used in Attacks																					
Asset	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓					✓	✓	✓			
Risk (indicators)	✓		✓					✓	✓				✓			✓					
Threat		✓	✓					✓	✓							✓					
Vulnerability				✓						✓						✓					
Affected Part of Adaptation																					
Controller	✓			✓	✓	✓		✓								✓		✓			
Sensors		✓	✓				✓	✓	✓		✓							✓			
Database			✓																		
Hazard Source																					
External	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓			
Internal					✓			✓	✓	✓	✓		✓				✓				
Hazard Cause																					
String Stability Decrease	✓				✓																
Crash				✓			✓	✓			✓										
Component Failure			✓					✓										✓			
Overheating													✓			✓					
System Failure								✓	✓		✓		✓	✓							✓
Performance Alteration	✓	✓	✓	✓						✓											
Collision	✓				✓		✓	✓		✓							✓				
Other				✓				✓					✓					✓	✓		
Safety Quality Factors																					
Property	✓	✓		✓	✓		✓			✓	✓		✓			✓	✓	✓			
Health	✓	✓		✓	✓			✓	✓	✓	✓		✓				✓				
Environmental													✓			✓	✓				
Treatment and Mitigation																					
Hazard Reduction	✓	✓	✓					✓					✓	✓			✓	✓		✓	
Hazard Control		✓	✓			✓			✓				✓				✓	✓			
Hazard Elimination	✓			✓	✓			✓					✓	✓							
Modeling Context																					
System	✓			✓	✓	✓	✓			✓		✓	✓	✓	✓	✓			✓		
Security		✓	✓		✓			✓		✓					✓						
Safety						✓						✓								✓	
Adaptation											✓				✓						
Analysis Objectives																					
Evaluation	✓			✓	✓	✓	✓		✓					✓		✓					
Detection					✓		✓							✓		✓		✓			
Risk & Threat		✓	✓					✓				✓									
Requirements	✓										✓							✓		✓	
Adaptation															✓						
Other			✓			✓													✓		
V & V Approach																					
Testing	✓		✓	✓	✓	✓	✓		✓					✓		✓	✓	✓			✓
Formal Verification						✓					✓								✓	✓	
Adaptation Property																					
Security			✓										✓					✓		✓	
Safety			✓			✓	✓											✓			
Other	✓		✓	✓	✓	✓		✓					✓			✓	✓	✓		✓	
Open Challenges																					
Adaptation											✓				✓				✓		✓
Environment											✓	✓							✓		
System											✓								✓		
Safety & Security											✓	✓									
Other							✓														

References

Alcaraz, C., Zeadally, S., 2015. Critical infrastructure protection: Requirements and challenges for the 21st century. Int. J. Crit. Infrastruct. Prot. 8, 53–66. http://dx.doi.org/10.1016/j.ijcip.2014.12.002.

Allouch, A., Koubâa, A., Khalgui, M., Abbes, T., 2019. Qualitative and quantitative risk analysis and safety assessment of unmanned aerial vehicles missions over the internet. IEEE Access 7, 53392–53410. http://dx.doi.org/10.1109/ACCESS.2019.2911980.

Altawy, R., Youssef, A.M., 2016. Security, privacy, and safety aspects of civilian drones: A survey. ACM Trans. Cyber-Phys. Syst. 1 (2), 1–25. http://dx.doi.org/10.1145/3001836.

Amorim, T., Martin, H., Ma, Z., Schmittner, C., Schneider, D., Macher, G., Winkler, B., Krammer, M., Kreiner, C., 2017. Systematic pattern approach for safety and security co-engineering in the automotive domain. In: Tonetta, S., Schoitsch, E., Bitsch, F. (Eds.), Computer Safety, Reliability, and Security. Springer International Publishing, Cham, pp. 329–342.

Andersson, J., de Lemos, R., Malek, S., Weyns, D., 2009. Software Engineering for Self-Adaptive Systems. Springer, Berlin, Heidelberg, pp. 27–47. http://dx.doi.org/10.1007/978-3-642-02161-9_2.

Badampudi, D., Wohlin, C., Petersen, K., 2015. Experiences from using snowballing and database searches in systematic literature studies. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering. EASE '15, Association for Computing Machinery, New York, NY, USA, pp. 1–10. http://dx.doi.org/10.1145/2745802.2745818.

Blair, G., Bencomo, N., France, R.B., 2009. Models@run.time. Computer 42 (10), 22–27. http://dx.doi.org/10.1109/MC.2009.326.

Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U., Habli, I., Kelly, T., 2018. Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE Trans. Softw. Eng. 44 (11), 1039–1069. http://dx.doi.org/10.1109/TSE.2017.2738640.

Carré, M., Exposito, E., Ibanez-Guzman, J., 2018. Challenges for the self-safety in autonomous vehicles. In: 13th Annual Conference on System of Systems Engineering (SoSE). pp. 181–188. http://dx.doi.org/10.1109/SYSOSE.2018.8428718.

Cheng, B.H.C., Eder, K.I., Gogolla, M., Grunske, L., Litoiu, M., Müller, H.A., Pelliccione, P., Perini, A., Qureshi, N.A., Rumpe, B., Schneider, D., Trollmann, F., Villegas, N.M., 2014. Models@run.time: Foundations, Applications, and Roadmaps. Springer International Publishing, Cham, pp. 101–136. http://dx.doi.org/10.1007/978-3-319-08915-7_4.

Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Di Marzo Serugendo, G., Dustdar, S., Finkelstein, A., Gacek, C., Geihs, K., Grassi, V., Karsai, G., Kienle, H.M., Kramer, J., Litoiu, M., Malek, S., Mirandola, R., Müller, H.A., Park, S., Shaw, M., Tichy, M., Tivoli, M., Weyns, D., Whittle, J., 2009. Software Engineering for Self-Adaptive Systems. Springer, Berlin, Heidelberg, pp. 1–26. http://dx.doi.org/10.1007/978-3-642-02161-9_1.

Chockalingam, S., Hadžiosmanović, D., Pieters, W., Teixeira, A., van Gelder, P., 2017. Integrated safety and security risk assessment methods: A survey of key characteristics and applications. In: Havarneanu, G., Setola, R., Nassopoulos, H., Wolthusen, S. (Eds.), Critical Information Infrastructures Security. Springer International Publishing, Cham, pp. 50–62.

D´Angelo, M., Gerasimou, S., Ghahremani, S., Grohmann, J., Nunes, I., Pournaras, E., Tomforde, S., 2019. On learning in collective self-adaptive systems: State of practice and a 3D framework. In: IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, pp. 13–24. http://dx.doi.org/10.1109/SEAMS.2019.00012.

DeBruhl, B., Weerakkody, S., Sinopoli, B., Tague, P., 2015. Is your commute driving you crazy? A study of misbehavior in vehicular platoons. In: Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks. WiSec '15, ACM, New York, NY, USA, http://dx.doi.org/10.1145/2766498.2766505.

dos Santos, E.B., Andrade, R.M., de Sousa Santos, I., 2021. Runtime testing of context-aware variability in adaptive systems. Inf. Softw. Technol. 131, 106482. http://dx.doi.org/10.1016/j.infsof.2020.106482.

Gerostathopoulos, I., Vogel, T., Weyns, D., Lago, P., 2021. How do we evaluate self-adaptive software systems?: A ten-year perspective of SEAMS. In: 16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE, pp. 59–70. http://dx.doi.org/10.1109/SEAMS51251.2021.00018.

Gheibi, O., Weyns, D., Quin, F., 2021. Applying machine learning in self-adaptive systems: A systematic literature review. ACM Trans. Auton. Adapt. Syst. 15 (3), 1–37. http://dx.doi.org/10.1145/3469440.

Ghosh, D., Sharman, R., Raghav Rao, H., Upadhyaya, S., 2007. Self-healing systems — survey and synthesis. Decis. Support Syst. 42 (4), 2164–2185. http://dx.doi.org/10.1016/j.dss.2006.06.011, Decision Support Systems in Emerging Economies.

Gibbs, G., 2018. Analyzing Qualitative Data. In: Qualitative Research Kit, SAGE Publications.

IBM, 2005. An architectural blueprint for autonomic computing. In: IBM (Ed.), Tech. rep., IBM, URL https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf.

IEC 61508 2010, 2010. IEC 61508:2010: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. International Electrotechnical Comission.

International Organization for Standardization, 2011. ISO 26262 road vehicles—Functional safety.

Johnson, N., Kelly, T., 2019. Devil's in the detail: Through-life safety and security co-assurance using SSAF. In: Romanovsky, A., Troubitsyna, E., Bitsch, F. (Eds.), Computer Safety, Reliability, and Security. Springer International Publishing, Cham, pp. 299–314.

Kephart, J., Chess, D., 2003. The vision of autonomic computing. Computer 36 (1), 41–50. http://dx.doi.org/10.1109/MC.2003.1160055.

Khan, M.J., Awais, M.M., Shamail, S., 2008. Self-configuration in autonomic systems using clustered CBR approach. In: International Conference on Autonomic Computing. pp. 211–212. http://dx.doi.org/10.1109/ICAC.2008.10.

Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009. Systematic literature reviews in software engineering–a systematic literature review. Inf. Softw. Technol. 51 (1), 7–15.

Kondeva, A., Nigam, V., Ruess, H., Carlan, C., 2019. On computer-aided techniques for supporting safety and security co-engineering. In: 2019 IEEE International Symposium on Software Reliability Engineering Workshops. ISSREW, pp. 346–353. http://dx.doi.org/10.1109/ISSREW.2019.00095.

Le, A., Maple, C., 2019. A simplified approach for dynamic security risk management in connected and autonomous vehicles. In: Living in the Internet of Things (IoT 2019). pp. 1–8. http://dx.doi.org/10.1049/cp.2019.0140.

Liang, H., Jagielski, M., Zheng, B., Lin, C.-W., Kang, E., Shiraishi, S., Nita-Rotaru, C., Zhu, Q., 2018. Network and system level security in connected vehicle applications. In: IEEE/ACM International Conference on Computer-Aided Design. ICCAD, pp. 1–7. http://dx.doi.org/10.1145/3240765.3243488.

Lisova, E., Šljivo, I., Čaušević, A., 2019. Safety and security co-analyses: A systematic literature review. IEEE Syst. J. 13 (3), 2189–2200. http://dx.doi.org/10.1109/JSYST.2018.2881017.

Liu, J., Ma, D., Weimerskirch, A., Zhu, H., 2017. A functional co-design towards safe and secure vehicle platooning. In: Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security. CPSS '17, ACM, New York, NY, USA, pp. 81–90. http://dx.doi.org/10.1145/3055186.3055193.

Macías-Escrivá, F.D., Haber, R., del Toro, R., Hernandez, V., 2013. Self-adaptive systems: A survey of current approaches, research challenges and applications. Expert Syst. Appl. 40 (18), 7267–7279. http://dx.doi.org/10.1016/j.eswa.2013.07.033.

MacQueen, K.M., McLellan, E., Kay, K., Milstein, B., 1998. Codebook development for team-based qualitative analysis. Cam Journal 10 (2), 31–36.

Muccini, H., Sharaf, M., Weyns, D., 2016. Self-adaptation for cyber-physical systems: A systematic literature review. In: IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. SEAMS, pp. 75–81. http://dx.doi.org/10.1109/SEAMS.2016.016.

Oscarson, P., 2003. Information security fundamentals. In: Irvine, C., Armstrong, H. (Eds.), Security Education and Critical Infrastructures. Springer US, New York, NY, pp. 95–107.

Pandey, A., Moreno, G.A., Cámara, J., Garlan, D., 2016. Hybrid planning for decision making in self-adaptive systems. In: IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems. SASO, pp. 130–139. http://dx.doi.org/10.1109/SASO.2016.19.

Petit, J., Shladover, S.E., 2015. Potential cyberattacks on automated vehicles. IEEE Trans. Intell. Transp. Syst. 16 (2), 546–556. http://dx.doi.org/10.1109/TITS.2014.2342271.

Potteiger, B., Zhang, Z., Koutsoukos, X., 2020. Integrated moving target defense and control reconfiguration for securing Cyber-Physical systems. Microprocess. Microsyst. 73, 102954. http://dx.doi.org/10.1016/j.micpro.2019.102954.

Salehie, M., Tahvildari, L., 2009. Self-adaptive software: Landscape and research challenges. ACM Trans. Auton. Adapt. Syst. 4 (2), 1–42. http://dx.doi.org/10.1145/1516533.1516538.

Sesame Project, 2022. SESAME - secure and safe multi-robot systems. URL https://www.sesame-project.org, (Accessed: 2022-06-27).

The MITRE Corporation, 2021. CAPEC-3000: Domains of attack. URL https://capec.mitre.org/data/definitions/3000.html, (Accessed: 2021-11-15).

Trapp, M., Schneider, D., 2014. Safety assurance of open adaptive systems–a survey. In: Models@Run.Time. Springer, pp. 279–318.

Usman, M., Britto, R., Börstler, J., Mendes, E., 2017. Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method. Inf. Softw. Technol. 85, 43–59.

Veledar, O., Damjanovic-Behrendt, V., Macher, G., 2019. Digital twins for dependability improvement of autonomous driving. In: Walker, A., O'Connor, R.V., Messnarz, R. (Eds.), Systems, Software and Services Process Improvement. Springer International Publishing, Cham, pp. 415–426. http://dx.doi.org/10.1007/978-3-030-28005-5_32.

Vogel, T., Giese, H., 2014. Model-driven engineering of self-adaptive software with EUREMA. ACM Trans. Auton. Adapt. Syst. 8 (4), 1–33. http://dx.doi.org/10.1145/2555612.

Weyns, D., 2017. Software engineering of self-adaptive systems: an organised tour and future challenges. In: Chapter in Handbook of Software Engineering. Springer, p. 2.

Weyns, D., Iftikhar, M.U., de la Iglesia, D.G., Ahmad, T., 2012. A survey of formal methods in self-adaptive systems. In: Proceedings of the 5th International C* Conference on Computer Science and Software Engineering. In: C3S2E '12, ACM, New York, NY, USA, pp. 67–79. http://dx.doi.org/10.1145/2347583.2347592.

Williams, M., Moser, T., 2019. The art of coding and thematic exploration in qualitative research. Int. Manag. Rev. 15 (1), 45–55.

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

Wong, T., Wagner, M., Treude, C., 2022. Self-adaptive systems: A systematic literature review across categories and domains. Inf. Softw. Technol. 148, 106934. http://dx.doi.org/10.1016/j.infsof.2022.106934.

Yuan, E., Esfahani, N., Malek, S., 2014. A systematic survey of self-protecting software systems. ACM Trans. Auton. Adapt. Syst. 8 (4), 1–41. http://dx.doi.org/10.1145/2555611.

Zhu, Q., Li, W., Kim, H., Xiang, Y., Wardega, K., Wang, Z., Wang, Y., Liang, H., Huang, C., Fan, J., Choi, H., 2020. Know the unknowns: Addressing disturbances and uncertainties in autonomous systems. ICCAD '20, ACM, New York, NY, USA, http://dx.doi.org/10.1145/3400302.3415768.

Selected papers

Amoozadeh, M., Raghuramu, A., Chuah, C.-n., Ghosal, D., Zhang, H.M., Rowe, J., Levitt, K., 2015. Security vulnerabilities of connected vehicle streams and their impact on cooperative driving. IEEE Commun. Mag. 53 (6), 126–132. http://dx.doi.org/10.1109/MCOM.2015.7120028.

Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C., 2004. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. 1 (1), 11–33. http://dx.doi.org/10.1109/TDSC.2004.2.

Beltrame, G., Merlo, E., Panerati, J., Pinciroli, C., 2018. Engineering safety in swarm robotics. In: Proceedings of the 1st International Workshop on Robotics Software Engineering. RoSE '18, ACM, New York, NY, USA, pp. 36–39. http://dx.doi.org/10.1145/3196558.3196565.

Bieder, C., Pettersen Gould, K., 2020. The Coupling of Safety and Security: Exploring Interrelations in Theory and Practice. Springer Nature.

Bolovinou, A., Atmaca, U.-I., Sheik, A.T., Ur-Rehman, O., Wallraf, G., Amditis, A., 2019. TARA+: Controllability-aware threat analysis and risk assessment for L3 automated driving systems. In: 2019 IEEE Intelligent Vehicles Symposium. IV, pp. 8–13. http://dx.doi.org/10.1109/IVS.2019.8813999.

Bucchiarone, A., Pelliccione, P., Vattani, C., Runge, O., 2009. Self-repairing systems modeling and verification using AGG. In: Joint Working IEEE/IFIP Conference on Software Architecture European Conference on Software Architecture. IEEE, pp. 181–190. http://dx.doi.org/10.1109/WICSA.2009.5290804.

Causevic, A., Papadopoulos, A.V., Sirjani, M., 2019. Towards a framework for safe and secure adaptive collaborative systems. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference. COMPSAC, 2, pp. 165–170. http://dx.doi.org/10.1109/COMPSAC.2019.10201.

Chinosi, M., Trombetta, A., 2012. BPMN: An introduction to the standard. Comput. Stand. Interfaces 34 (1), 124–134. http://dx.doi.org/10.1016/j.csi.2011.06.002.

Ferrão, I.G., Pigatto, D.F., Fontes, J.V.C., Silva, N.B.F., Espes, D., Dezan, C., Branco, K.R.L.J.C., 2020. STUART: ReSilient archiTecture to dynamically manage Unmanned aeriAl vehicle networks under atTack. In: 2020 IEEE Symposium on Computers and Communications. ISCC, pp. 1–6. http://dx.doi.org/10.1109/ISCC50000.2020.9219689.

Ismail, S., Shah, K., Reza, H., Marsh, R., Grant, E., 2021. Toward management of uncertainty in self-adaptive software systems: IoT case study. Computers 10 (3), 27.

Knight, J.C., Strunk, E.A., 2004. Achieving critical system survivability through software architectures. In: de Lemos, R., Gacek, C., Romanovsky, A. (Eds.), Architecting Dependable Systems II. Springer, Berlin, Heidelberg, pp. 51–78. http://dx.doi.org/10.1007/978-3-540-25939-8_3.

Monteuuis, J.-P., Boudguiga, A., Zhang, J., Labiod, H., Servel, A., Urien, P., 2018. SARA: Security automotive risk analysis method. In: Proceedings of the 4th ACM Workshop on Cyber-Physical System Security. CPSS '18, ACM, New York, NY, USA, pp. 3–14. http://dx.doi.org/10.1145/3198458.3198465.

Parkinson, S., Ward, P., Wilson, K., Miller, J., 2017. Cyber threats facing autonomous and connected vehicles: Future challenges. IEEE Trans. Intell. Transp. Syst. 18 (11), 2898–2915.

Petersen, K., Gencel, C., 2013. Worldviews, research methods, and their relationship to validity in empirical software engineering research. In: Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement. IEEE, pp. 81–89.

Plósz, S., Schmittner, C., Varga, P., 2017. Combining safety and security analysis for industrial collaborative automation systems. In: International Conference on Computer Safety, Reliability, and Security. Springer, pp. 187–198.

Polack, F.A.C., 2010. Self-organisation for survival in complex computer architectures. In: Weyns, D., Malek, S., de Lemos, R., Andersson, J. (Eds.), Self-Organizing Architectures. Springer, Berlin, Heidelberg, pp. 66–83. http://dx.doi.org/10.1007/978-3-642-14412-7_4.

Schneider, D., Becker, M., Trapp, M., 2011. Approaching Runtime Trust Assurance in Open Adaptive Systems. SEAMS '11, ACM, New York, NY, USA, pp. 196–201. http://dx.doi.org/10.1145/1988008.1988036.

Settanni, G., Skopik, F., Karaj, A., Wurzenberger, M., Fiedler, R., 2018a. Protecting cyber physical production systems using anomaly detection to enable self-adaptation. In: 2018 IEEE Industrial Cyber-Physical Systems. ICPS, pp. 173–180. http://dx.doi.org/10.1109/ICPHYS.2018.8387655.

Settanni, G., Skopik, F., Wurzenberger, M., Fiedler, R., 2018b. Countering targeted cyber-physical attacks using anomaly detection in self-adaptive Industry 4.0 Systems. E & I Elektrotechnik Und Informationstechnik 135 (3), 278–285. http://dx.doi.org/10.1007/s00502-018-0615-6.

Teimourikia, M., Fugini, M., 2017. Ontology development for run-time safety management methodology in Smart Work Environments using ambient knowledge. Future Gener. Comput. Syst. 68, 428–441. http://dx.doi.org/10.1016/j.future.2016.07.003.

Yoon, M.-K., Liu, B., Hovakimyan, N., Sha, L., 2017. VirtualDrone: Virtual sensing, actuation, and communication for attack-resilient unmanned aerial systems. In: Proceedings of the 8th International Conference on Cyber-Physical Systems. ICCPS '17, ACM, New York, NY, USA, pp. 143–154. http://dx.doi.org/10.1145/3055004.3055010.

Irdin Pekaric is a postdoctoral researcher at the Department of Information Systems and Computer Science at the University of Liechtenstein. He obtained his Ph.D. degree at the Institute of Computer Science at the University of Innsbruck, Austria. His research areas lie in the field of information security. Specifically, he conducts research on attack model generation for integrated security and safety analysis, cross-analysis between software libraries and vulnerabilities, and mining of security data. His prior work focused on user anomaly detection on both the host and network levels.

Raffaela Groner is a Ph.D. student at Ulm University. Her research is focused on the performance of model transformations. Prior, she studied Computer Science at the Ulm University, where she received her M. Sc. in Computer Science.

Thomas Witte is a Ph.D. student at Ulm University. His research is mainly focused on software architectures for mobile robotics and data provenance tracking in component-based architectures. Prior, he studied. Computer Science at the Ulm University, where he received his M. Sc. in Computer Science.

Jubril Gbolahan Adigun is a researcher and Ph.D. candidate at the Quality Engineering Lab, Department of Computer Science, University of Innsbruck, Austria. Jubril holds a bachelor's degree in Electrical and Electronics Engineering from the University of Ilorin, Ilorin, Nigeria and a joint M.Sc. in engineering degree in Software Engineering from the University of Tartu and Tallinn University of Technology, Estonia. His Ph.D. focuses on the Assurance of AI systems under uncertainty where he combines his experiences in testing and working with simulators. Jubril is also actively teaching and supervising both bachelor's and master's students at UIBK.

Alexander Raschke has a permanent position as a post-doc at the Institute of Software Engineering and Programming Languages at Ulm University. His research interests are in the area of model-driven software engineering and formal methods. He wants to improve developer's experience through better modeling tools, but also through suitable expressive domain-specific languages and automated analysis on them.

Michael Felderer is the Director of the Institute for Software Technology within the German Aerospace Center (DLR) as well as a full professor of software technology at the University of Cologne, and an associate professor of software engineering at the University of Innsbruck. He holds a Ph.D. and a habilitation degree in computer science. His research interests in software and security engineering include software and security testing, empirical methods in software and security engineering, software and security processes, software analytics, risk management, requirements engineering, model engineering, and data-driven engineering. He works in close collaboration with industry and is a regular speaker at industrial conferences.

Matthias Tichy is a full professor for software engineering at Ulm University and director of the Institute of Software Engineering and Programming languages. His main research focuses on model-driven software engineering, particularly for cyber–physical systems. He works on requirements engineering, dependability, and validation and verification complemented by empirical research techniques. He is a regular member of program committees for conferences and workshops in the area of software engineering and model driven development.