IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 23, 2020, accepted November 9, 2020, date of publication December 9, 2020, date of current version January 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3043440

Self-Adaptive Software Systems in Contested and Resource-Constrained Environments: Overview and Challenges

CLAUDIA SZABO[1], (Member, IEEE), BRENDAN SIMS[2], THOMAS MCATEE[1], RILEY LODGE[2], AND ROBERT HUNJET[2]

[1]School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia
[2]Defence Science and Technology Group, Edinburgh, SA 5111, Australia

Corresponding author: Claudia Szabo (claudia.szabo@adelaide.edu.au)

ABSTRACT Self-adaptive approaches are a promising to address the dynamic and uncertain nature of the environments where today's complex systems operate. In particular, systems operating in military environments, during crises or under unexpected conditions need to address critical concerns related to resource sparsity and the unstable and uncertain nature of exchanged information. Despite a plethora of self-adaptive and autonomic approaches proposed in the last decade, very few have been designed for or evaluated in *contested environments*, where adversarial action in the communications domain leads to stale or incomplete information, or in *resource-constrained environments*, where resources are either limited or required by a large number of components. These conditions are where self-adaptability to aid human operators is needed the most. To better understand self-adaptation in contested and resource-constrained environments, we conducted a systematic literature review of publications over the last decade. We conduct our review through the lens of a military environment, where contention, both physical and from a resource, be it computational or communication based is at its peak. We followed the systematic literature review methodology and analysed 238 primary studies. We identified that the most frequent application domains are those where failures are frequent and costly, namely, cloud computing, web services and applications, and servers. Despite this, less than 3% of the papers considered constrained resources *and* stale or incomplete information and a significant focus was on developing centralised solutions instead of distributed ones throughout all papers. Very few papers (4.6%) considered environments where the information about the system components was not readily available. No papers evaluated the systems running in contested and resource-constrained environments. We present an analysis of the self-adaptive systems that consider incomplete or stale information and constrained resources, discuss their limitations and identify future areas of research. Critical research gaps include the lack of evaluation of self-adaptive approaches, including the lack of standards or formalisms to allow for the comparison of various approaches. In addition, there is a need to consider more self-* properties and non-functional requirements in order to make sure the designed system is resilient. This paper presents our review findings in detail, examines how self-adaptation happens in contested and resource-constrained environments, and discusses the identified research gaps.

INDEX TERMS Adaptability, contested environments, self-adaptive systems, self-organization.

I. INTRODUCTION

Self-adaptation is recognised as one of the most effective approaches to manage and improve current large-scale dynamic software systems [1]–[3] that have increasingly complex requirements. These include resilience, dependability, and energy efficiency among others, as dictated by the specific environments in which they operate. Despite the prevalence of self-adaptive solutions in domains such as cloud computing [4], [5], web servers [6], [7] and wireless networks [8], [9], where faults and failures are ubiquitous and costly [10]–[12], few existing solutions operate in environments with constrained resources and/or where complete system information is not current and reliable. Constrained

The associate editor coordinating the review of this manuscript and approving it for publication was Feiqi Deng.

resources and unreliable information sources can be found in military land-based operations, where the potential benefits of using self-adaptive systems are significant [13], [14].

In the last decade, military land vehicles have transitioned to state-of-the-art digital platforms that include advanced communication, situational awareness and battle management mission systems [15]–[18]. The development of open electronic architecture approaches for these vehicles, such as the NATO Generic Vehicle Architecture [19], has facilitated mission system networking and integration. These systems perform several functions such as the sensing of vehicle and environmental status and the integration and dissemination of battlefield intelligence and commands. This allows for extensive tactical coordination before and during battles, with increased operational effectiveness and improved survivability. However, achieving these functions involves a consistent and continuous exchange of information amongst mission systems within and between vehicles and requires reliable system operation. Without a capability to manage integrated mission systems, operators may be required to process this information and reconfigure and repair on-vehicle mission systems while making critical battle decisions, which might negatively affect mission outcomes.

Military land vehicles operate in highly contested and resource-constrained environments, where the sensed information critical to their functionality is not correct and complete. This can be because the vehicle is under attack (either physically or electronically, e.g. through jamming) or because of the nature of its operating environment, e.g., a desert or a disaster area without specific communications infrastructure. In addition, the number of communicating systems as well as the large amount of information shared may pose a significant load on the limited physical computing infrastructure and on the bandwidth-limited network used for status and operational communication [20], [21]. Moreover, in future land deployments it is likely that specific resources or services can be accessed across vehicles; as such, resource and service provisioning decisions will not be made in a centralised manner and will require coordination and consensus many vehicles. This results in a critical need for mission systems to be able to operate reliably on information ranging from complete and timely to incomplete and outdated, and to manage constrained resources without service degradation.

Self-adaptive and autonomic approaches. Reference [22] are appealing to address these challenges. Automation can improve mission system management, and self-adaptation can address the highly contested and resource-constrained nature of the operational environment, the distributed nature of servicing decisions, and the highly dynamic and unpredictable nature of battle.

Advances have been made in adaptability within military subsystems to perform within very stringent policies. For example, a radio may change its data rate depending on received signal strength, or an unmanned aerial vehicle (UAV) might change its wing configuration upon turbulence. However, very few existing self-adaptive approaches applied to military systems embrace adaptability at high levels of abstraction, e.g. allowing a UAV to re-task itself in the event of having observed a new surveillance target.

The lack of self-adaptation in military applications has been acknowledged by Australian Defence and was a key driver in the instantiation of Australia's first Defence-led Collaborative Research Centre: the DCRC in Trusted Autonomous Systems [23]; this is a stand-alone company which has been created through a $50 million investment from the Australian Government. The centre was created to address the issue that, despite the promise of autonomous systems,[1] no autonomous system fielded within the Australian Defence Force (ADF) exist [24]. The realisation of contextually-aware self-adaptation is a mechanism to address this issue and to enhance the resilience and intelligence of machines.

Existing self-adaptive and autonomic approaches may inform the development of self-adaptive software systems for military applications, but the majority of literature on this topic does not consider the military domain explicitly. Existing works either consider perfect environments [25] or lack evaluation of key system properties [26] such as scalability [13], [14], [27], [28] or security [29], [30] among others. However, several approaches have been proposed in crisis management situations, where scenarios may include similar environmental complexity such as unreliable communication performance [25], and limited battery performance [31]. Issues relating to resource contention and a lack of complete information about system components are also ubiquitously found in many large scale distributed systems.

The environment type and its associated constraints, and how these are addressed by self-adaptive systems is not considered by existing surveys and systematic literature reviews on self-adaptive systems [32], [33]. To address this gap, in this paper, we conduct our own investigation into the current state of self-adaptive systems within the literature. This is achieved through a systematic literature review of self-adaptive and autonomic approaches published since 2008 to identify approaches that have been designed and evaluated in contested and/or resource-constrained environments. We analyse evaluation metrics, employed resource congestion mitigation algorithms, as well as mechanisms for handling incomplete or stale information. Our analysis also identifies the challenges and limitations of using autonomic and/or self-adaptive approaches in such environments, and existing research gaps. Through this investigation and analysis we elicit a better understanding of self-adaptation in contested and resource-constrained environments to inform the development and evaluation of self-adaptive approaches for military applications. Our contribution is twofold:

- A systematic literature review of 238 papers and an extensive analysis of their application domain, resource

[1] Autonomous systems are defined by the DCRC in Trusted Autonomous Systems as contextually-aware self-adaptive systems to ameliorate ambiguity associated with remotely piloted systems, which exhibit no ability to act without human intervention and are therefore not autonomous.

congestion mitigation algorithms, self-* properties and stated limitations among others.
- An analysis of the challenges and limitations of the limited number of approaches that consider both resource-constrained and contested environments, including an identification and discussion of current research gaps.

II. RELATED WORK

Despite extensive research in the area of self-adaptive systems, few systematic literature reviews exist [32]–[35], and within these, to the best of our knowledge, no reviews focus on self-adaptive systems specifically designed for or working on environments with the characteristics outlined above, namely, with congested resources and where information about the system is incomplete or stale. We discuss these as well as other literature reviews in the below.

A study by Patikirikorala *et al.* [32] looks at research papers published in several venues between 2000 and 2011 and classifies them based on their specific application domain and performance analysis variables, identifying an extensive list of performance metrics that can be used to evaluate the performance of a self-adaptive system. The paper is focused only on approaches that use control engineering in their design and the authors employ a manual search from specific venues to identify relevant papers. The authors propose a taxonomy that captures the target system (its application domain, performance variables, and dimension), the control system (model, type, scheme), and the validation of the approach (simulation or case study). We use a similar taxonomy in our literature review.

A more recent systematic literature review is the 2016 survey of Muccini *et al.* [33], who focus on self-adaptation in cyber-physical systems and survey an extensive list of papers from the ACM DL, SpringerLink and Web of Science. They identify 42 primary studies and find that adaptation in cyber-physical systems is a cross-layer concern, where solutions combine different adaptation mechanisms within and across layers such as application, middleware, communication and service. They also find that a large percentage of the studies (36%) combine different mechanisms to realise adaptation, either combining Monitor-Analyse-Plan-Execute (MAPE) adaptation mechanisms with agents, reflection with self-organisation, agents with self-organisation, agents with reflection and self-organisation, and lastly MAPE function with self-organisation. This result shows that there is no one-size-fits-all approach to self-adaptive design, especially in the context of complex dynamic environments.

Dynamic and uncertain environments [36] are a key driver for trade-offs in self-adaptive systems as identified by Salama *et al.* [34]. They perform a systematic mapping study aimed at understanding how various trade-offs are managed in self-adaptive systems, and consider papers with explicit techniques or architectures that manage trade-offs. The study includes twenty primary studies to classify software paradigms, quality attributes considered, and the self-* properties that drive trade-offs management. Their analysis identifies the need for a framework to manage trade-offs considering a variety of quality attributes, dynamic and uncertain environments, as well as the complex challenges of modern, ultra-large scale systems. A similar study is that of Mahdavi-Hezavehi *et al.* [35], which focuses on analysing methods that handle multiple quality attributes in architecture-based self-adaptive systems. Focusing only on one of the self-* properties, Yuan *et al.* [37] conduct a systematic survey of self-protecting software systems.

Other reviews and surveys lack the methodology of a systematic study and may miss significant work, however they are nevertheless able to identify research challenges and knowledge gaps [38]–[40]. Weyns *et al.* [39] present a survey on the use of formal methods in self-adaptive systems, focusing on how specific properties could be verified. Parunak and Brueckner [40] identify the challenges of using software engineering methods in the design, implementation, and evaluation of self-adaptive systems, while Weyns *et al.* [38] discuss the claims and supporting evidence of self-adaptive systems, highlighting the importance of sound evaluation methods. This is further strengthened by the work of de Lemos *et al.* [36], who focus on the critical need to provide quality assurances as self-adaptive systems become ubiquitous. A potential evaluation framework is presented in [41], where metrics to evaluate both the controller and the managed systems are discussed, both from a performance and a quality perspective.

III. SELF-ADAPTATION IN CONTESTED AND RESOURCE CONSTRAINED ENVIRONMENTS

Within the domains of military and crisis situations, and in particular in the context of military land vehicles, the ability to self-adapt is a relatively new concept. Mission systems on these vehicles are designed to support soldiers, but the unpredictable nature of battle can often impact their performance and functionality. This may range from the degradation of sensor systems, to system faults and to significant outages in communication. The realisation of self-adaptability in this context offers a means by which performance and functionality can be sustained, recovered, protected or enhanced, and new ways to deliver required functions can be discovered. Furthermore, it provides a means to support mission system operation in dynamic environments, in which requirements and objectives may frequently change potentially requiring reconfiguration of systems. Overall, the resilience of mission systems is likely to improve, which is highly valuable to soldiers as they are reliant on these systems to communicate, operate and survive. Without self-adaptability, the responsibility to reconfigure and repair mission systems falls upon the soldier. This presents an added task requiring their attention in addition to their operational duties. Given mission systems are becoming increasingly complex, such a responsibility presents a significant cognitive burden on soldiers.

When consider self-adaptability to refer to the ability of a system to independently identify the need to adapt, plan

an appropriate course of action, and undertake changes to address the adaptation requirement. This may include reconfiguring computer systems running on military vehicles, reallocating resources, establishing new links between systems and services, and restarting and repurposing systems. A constraint faced by military land vehicles is that additional resources are typically unavailable and any adaptation must work with existing systems. In addition, power sources are scarce and as such any self-adaptive system running on such vehicles must resolve resource contention among others.

An example of basic self-adaptability in the case of military land vehicles is where particular mission system applications rely on specific algorithms to achieve their core functionality, e.g., parsing and integrating sensor data from a set of cameras and position sensors. A self-adaptive approach in this example could offer the ability to leverage a suite of algorithms and switch between them in response to changes in the current context, such as changes in network topology for distributed applications or changes in resource availability. Policy-based rule-sets could then be applied to modify the configuration of a vehicle's mission system and required algorithms and necessary data streams could be activated in response to configuration changes.

Although it is anticipated that the introduction of self-adaptive approaches will be beneficial to soldiers, it is not immediately apparent how such benefit should be quantified. Further effort is required to establish metrics and approaches that allow self-adaptive software systems in contested and resource-constrained environments to be understood and their impact to be measured. Moreover, when more than one self-adaptive approach is available, there is a need to analyse their benefits and disadvantages. To the best of our knowledge, to date there has been no analysis of self-adaptive systems from the perspective of contested and resource-constrained environments and their evaluation, and our study aims to address this gap.

IV. METHODOLOGY

Our survey of existing work has identified that, while there are a number of systematic literature surveys, none, to the best of our knowledge, focus on the application of self-adaptive systems in contested and resource-constrained environments, where information about the system may not be available, resources may be highly congested, and faults as well as attacks are frequent. To address the critical need for understanding self-adaptation and its evaluation in this context, we formulated the following research questions:

- **RQ1** What are the self-adaptive software systems approaches that consider contested and resource-constrained environments in their design and evaluation?
- **RQ2** What are the appropriate metrics for the evaluation of self-adaptive software approaches in contested and resource-constrained environments?
- **RQ3** What are the challenges and limitations of using self-adaptive software systems in contested and resource-constrained environments?

We conducted a systematic literature review to answer the first research question, following the systematic literature review guidelines [42]. A systematic literature review (SLR) gathers and evaluates all the research results on a selected research topic [43], [44]. It is closely related to a more shallow type of secondary study of systematic mapping studies (SMS), which aim at the classification and thematic analysis of earlier research [42], [45], [46]. Kitchenham and Charters [42] present the best practices of both for the field of software engineering and also compare the two. SMS is more general in search terms and aims at classifying and structuring the field of research, while the target of SLR is to summarise and evaluate research results. Kitchenham and Charters [42] also discuss the applications and observe that SMS can be especially suitable if only a few literature reviews have been done on the topic and there is a need to get a general overview of the field of interest. Both kinds of studies can be used to identify research gaps in the current state of research.

In this study, we first identified and selected the research studies from relevant databases following our search string. We then assessed the inclusion of the studies in our pool based on the selection criteria. We read each paper and performed data extraction of relevant data items. We discuss our process and results below.

A. IDENTIFICATION OF RESEARCH

We first performed a trial search to calibrate both our understanding of the selection/exclusion criteria but also the search strings. The group searched manually within the Software Engineering for Adaptive and Self-Managing Systems (SEAMS) symposiums and the ACM Transactions on Adaptive and Autonomous Systems in the last three years, analysing the title, abstract, and paper content and refining the search string throughout the process. This resulted in 21 papers. We also determined the search string that maximised precision (only relevant papers were included) and recall (all relevant papers were included), as discussed below.

In the final iteration of the research identification process, the search string was then used on ACM DL, IEEExplore, SpringerLink and ScienceDirect. The resulting papers were then read to determine whether they passed the inclusion/exclusion criteria (see Section IV-B), and removed from the list if they did not. The search using the search string defined below resulted in a total of 8,952 papers. After removing duplicates, resulting in 8,557 papers, we read the abstract of the papers to determine whether the papers were to be included/excluded from the study. The Zotero [47] tool was used in this process. A total of 770 papers remained after the application of the inclusion/exclusion criteria only to the abstract. The papers were then split equally among the authors and the process of reading and data item extraction began. The process included weekly calibration meetings, where researchers read the same ten papers, extracted data,

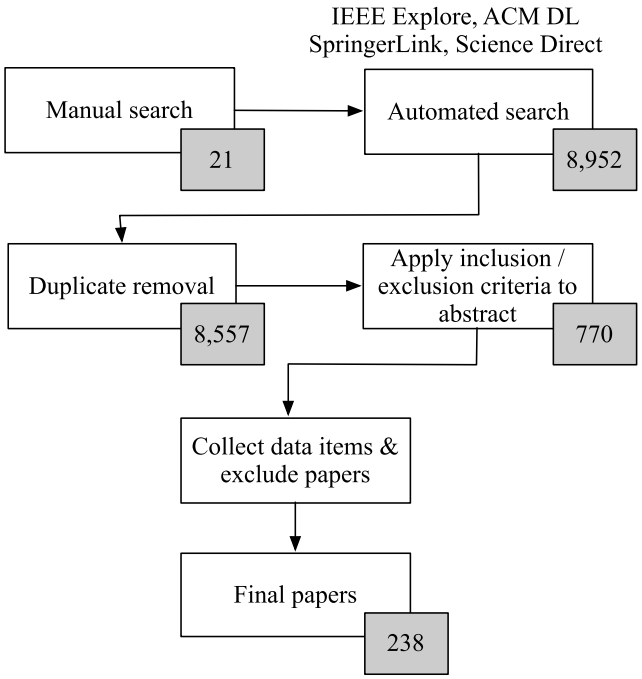


FIGURE 1. Systematic literature review process overview.

and discussed their results. In the first week, the process was repeated on different papers until an inter-rater reliability of 100% was obtained. Subsequent weeks showed an inter-rater reliability of 100%. Papers could be also be eliminated in this step, once a more in-depth read of the paper showed that it did not match the inclusion criteria. At the end of the data item extraction phase, a total of 238 papers remained,[2] as shown in Figure 1.

B. SEARCH STRING AND INCLUSION/EXCLUSION CRITERIA

The search string used boolean operators to refine the search and was: `"('self-*' OR 'autonomic') AND ('arch*' OR 'system' OR 'design')"` and was adapted to the specific database. Only papers published since 2008 were considered. The criteria for inclusion/exclusion were:

- *On topic* - papers were presenting self-adaptive or autonomic systems and their evaluation. Surveys as well as vision and position papers were excluded.
- *Length* >= 5 *pages* - short papers were excluded.
- *Language* - only English papers were included.

Beyond survey papers and vision and position papers, we also exclude papers that do not present any evaluation. We consider 'evaluation' to mean that the approach presented in the paper has undergone some form of analysis, either through a theoretical or practical example, through extensive experiments, proof of concepts, analytical analysis, or formal verification among others. Papers where an approach is presented but not evaluated are excluded.

[2]A link to all the 238 papers can be found here [48].

C. DATA ITEMS

Table 1 presents an overview of the data items extracted from the papers.

TABLE 1. Extracted data items.

ID	Data Item	Values
F1	Title	
F2	Year	
F3	Venue	
Architectural Decisions		
F4	Nature of managing system	Centralised/Decentralised [49]
F5	Nature of managed system	Centralised/Distributed [49]
F6	Sensors	
F7	Information type	Complete/Incomplete/Stale
F8	Information type mechanism	
F9	Congested resource	
F10	Congested resource mechanism	
F11	Runtime	Realtime/Stepped
Evaluation		
F12	Type of approach	Theoretical/Developed/ Deployed
F13	Type of analysis	Simulation/Case study/ Proof/Qualitative
F14	Performance measures	
F15	Quality measures	
F16	Deployed	Yes/No
F17	Repeatable	Yes/No
F18	Self-*	
Domain Specific Properties		
F19	Domain	
F20	Scalability	Yes/No
F21	Fault Tolerance	Yes/No
F22	Security	Yes/No
F23	Evolvability	Yes/No
F24	Tools	
F25	Programming language	
F26	Stated limitations	

1) NATURE OF MANAGING/MANAGED SYSTEM

A self-adaptive software system is comprised of a managing system and managed system. As defined in [49], the *managed system* comprises the application logic that provides the system's domain functionality, while the *managing system* oversees the managed system and comprises the adaptation logic that deals with various concerns.

2) INFORMATION TYPE

If all components in the system are aware of each other, and can observe each other's state, or if the system is always operating under the assumption that the information monitored/received is perfect, then the system is operating with *Complete* information. When components cannot observe the internal state of other components and may not be aware of other components in the system, the system is operating with *Incomplete* information. When components assume that specific information might become out of date, and a refresh process is discussed, then the system is operating with *Stale* information. The **information type mechanism** data item captures the mechanism employed by the system when dealing with incomplete or stale information. This data item also captures whether the system works in a contested environment.

3) CONGESTED RESOURCE

This data item and its pair (F10) captures whether the aim of the system is to improve a congested resource (and the description of the congested resource if that is the case) and the mechanism used to do so. This data item captures whether the system works in a resource constrained environment.

4) RUNTIME

This data item captures whether the system is managed in real-time or its operation needs to be stopped in order for (manual) re-configuration to occur (i.e. stepped).

5) TYPE OF APPROACH

This data item captures whether the proposed approach is *theoretical*, a prototype has been *developed*, or the approach has been *deployed* in a real-life, potentially commercial, environment.

6) QUALITY MEASURES

This data item captures any quality measures that the authors use to evaluate their approach, such as usability and feasibility among others.

7) DOMAIN SPECIFIC PROPERTIES

Data items F20-23 capture properties that are significant to most contested and resource constrained environments. Systems operating in these environments would need to be *fault tolerant*, have a *security* module or a security focused architecture, but would also need to be *scalable* and be able to *evolve* when faced with dynamic constraints and configurations.

8) STATED LIMITATIONS

This data item captures the approach limitations as identified by the paper authors.

D. QUALITATIVE ANALYSIS

To answer our research questions using most of the extracted data items from Table 1, a quantitative analysis was the straightforward approach. However, to better understand the application domains, the types of constrained resources and the algorithms employed to mitigate resource congestion, as well as to better understand the limitations of the papers, we performed a qualitative analysis guided by grounded theory, which is a state-of-the-art approach for extracting information from open-ended text. Grounded theory involves the establishment of a coding framework and analysis environment derived from the data itself rather than themes from research literature [50]. There are significant advantages in the adoption of a grounded theory approach, in contrast to directed content analysis with an established coding framework, including removing the potential to force fit observations into existing categories and misclassification. We commenced using bottom-up open coding [50] by reading through the collected data item values (F8-10, F14-15, F18-19, F24-26) and tagging blocks of text representing a concept related to application domains, constrained resources, algorithms, and limitations respectively. Once this step was complete, we commenced a focused coding process and mapped the identified blocks of text to specific categories. We discuss this grouping together with examples in each of the relevant sections below.

V. RESULTS

Our analysis identified over 30 application domains, as shown in Table 2, where we list under 'Other' application domains that appear only once, namely, electronics, simultaneous multi-threading processors, task scheduling, data processing, sandboxing, spacecraft and weapons systems. The most frequent application domain is cloud computing (representing 13% of the papers), followed by web services and applications (12.6% of papers), and servers (9.2%).

TABLE 2. Application domains (N = 238).

Domain	Frequency Count
Cloud Computing	31
Web Services and Applications	30
Servers	22
Wireless Networks	16
Intelligent Environments	**16**
** Smart Buildings	4
** Ubiquitous Computing	4
** Ambient Intelligence	2
** Assistive Applications	2
** Internet of Things	4
Security	14
Media and Data Streaming	12
Service-oriented Systems	12
Networks	11
Distributed Technologies	10
Autonomous Vehicles	9
Communication Systems	8
Data Centres	8
Crisis Management	6
Grid Computing	6
Mobile Systems	6
Transportation	6
Computer Vision	4
Electrical Power Systems	4
Simulation	4
Parallel Computing	3
Process Control	3
Systems-on-Chips	3
Surveillance	2
Automated Utilities	2
Middleware	2
Multimedia	2
Robotics	2
Other	7

Nature of the Managed/Managing systems: Figure 2 presents an overview of the spread of the managed and managing systems (data items F4 and F5). As it can be seen, most managed systems are distributed, representing 76.5% of all managed systems. In contrast, the majority of the managing systems (61.8%) are centralised. This might be due to the ease of implementation of a centralised managing system, as distributed systems issues such as synchronisation, coordination, and agreement do not have to be addressed.

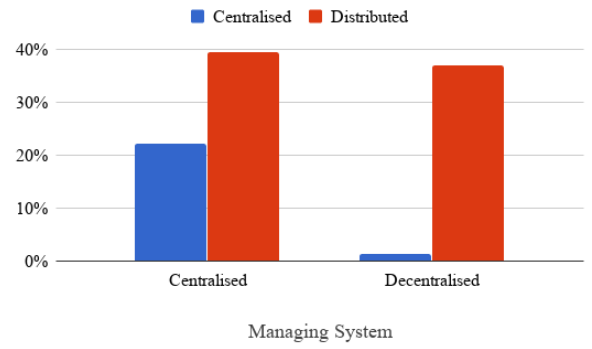


FIGURE 2. Overview of managed and managing systems (N = 238).

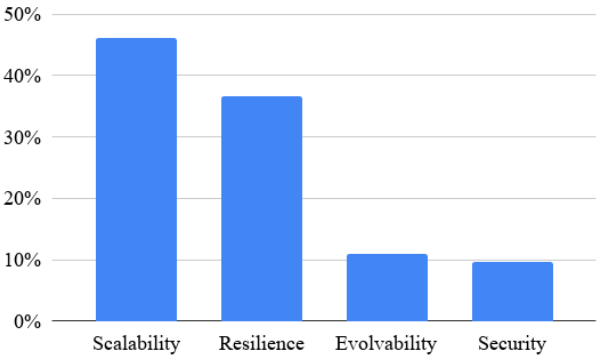


FIGURE 3. Overview of domain specific considerations (N = 238).

Our analysis of data item F11 identifies that the majority of systems (97.9%) are managed in real-time, with only 6 systems managed in a timestepped manner, that is, requiring the managed system to be stopped and restarted at every adaptation.

Nature of system properties: Our data collection in data items F20-F23 considers system properties that were present in the system evaluation, specifically scalability, fault tolerance, security and evolvability. As discussed above, we consider these properties as they would be significant in most contested and resource constrained environments. A summary of the results is shown in Figure 3. As it can be seen, the most frequently evaluated property is scalability (46% of papers) followed by resilience (37.3%) while 11% of papers evaluated evolvability. Despite cloud computing and web services and applications being the most common application domains, security is the least evaluated property (9.7% of papers). 16.8% of papers did not evaluate any of the properties on our list.

We also collected in data item F18 the type of Self-CHOP properties identified and evaluated in the included papers. Self-configuring, healing, optimising and protecting (CHOP) are important characteristics of self-adaptive systems and are defined in [51] and [22]. A *self-configuring* system is one that is able to automatically reconfigure itself in response to changing environments. A *self-healing* system is able to discover, diagnose, and correct system malfunctions without disruption to its operating environment. A *self-optimising* system automatically monitors and tunes resources to optimally meet end-user or business requirements. Finally, a *self-protecting* system anticipates, detects, identifies and protects against threats to the system.

Our analysis showed that 7.9% of papers did not have any self-CHOP properties. In Figure 4 we present the properties as they appear together, with the maximum present set shown. The most common property set identified is `C`, representing 13% of papers, followed by `CO` and `H` representing 11% and 8% of included papers respectively. As expected from our security analysis in Figure 3, only 7% of papers identified protection (either individually or as a set). Furthermore, only 2% of papers consider the entire `CHOP` property set in their evaluation. This indicates that consideration of all of these properties together in a self-adaptive software system is uncommon despite the large number of citations of the autonomic computing vision paper [51].

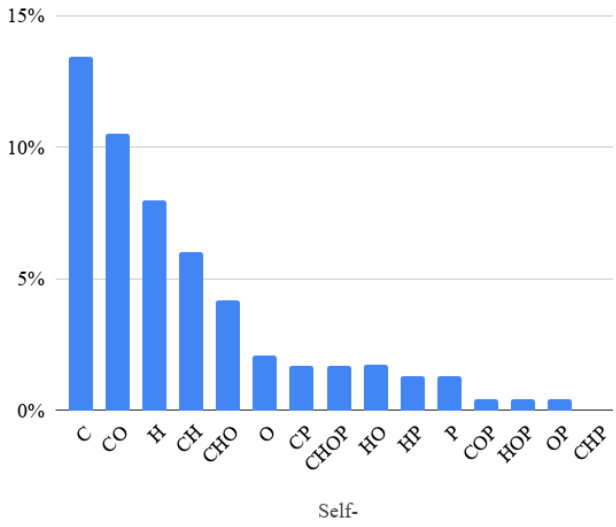


FIGURE 4. Self-CHOP properties (N = 238).

We extend this analysis and look at all self-* properties as identified and discussed within each paper and present our results in Figure 5. We can observe that the three most popular self-* properties were self-configuring, self-optimising, and self-healing. Self-organisation, a critical property in achieving adaptability, is considered only by 14% of the papers. An explicit consideration of self-organisation would allow approaches to move towards decentralised solutions thus providing a significant increase in fault tolerance and resilience, particularly in contested and resource constrained environments.

Evaluation type: We identified that the most common form of evaluation (data item F13) is through simulation, with 74.1% evaluations being performed through simulation, 21% through case studies, 2.2% through descriptive qualitative analysis, and 2.6% being evaluated through proofs. The majority of the approaches had some form of developed prototype (75.9%), and 21.9% of the approaches

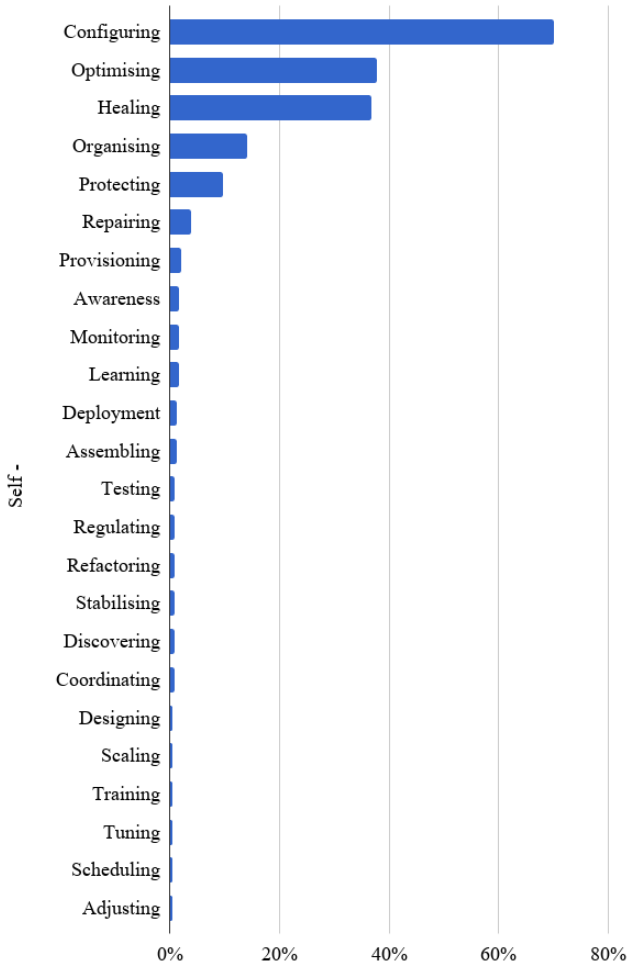


FIGURE 5. Overview of self-* properties (N = 238).

were theoretical. Only 2.2% of the approaches were deployed in a real-life or commercial setting. This highlights a gap between published research efforts and the evaluations of self-adaptive systems in real-life scenarios.

Evaluation metrics: Our analysis identified 509 individual metrics, with an average of 2.1 metrics per paper. All identified metrics are shown in Table 3.

A number of metrics have been grouped in this analysis as they capture a variety of single-instance metrics that do not fit in the presented categories. The "Other" category is the largest of these with 33 metrics. We identify 17 metrics in the "Utility" category, which are explicitly presented as utility specific to the problem being solved in their respective papers, but often contain a combination of domain-specific variables. Metrics based on scores are also grouped under "Utility". The "Time (Other)" category contains a collection of metrics relating to time such as travel time and waiting time while "Overhead (Other)" captures various overhead-related metrics including virtual machine provisioning overhead. "Reliability" metrics relate to faults (e.g. failure probability, time to failure) hence they have been grouped under "Fault handling". The "Accuracy" category contains a collection of accuracy-related metrics from different domains such as accuracy metrics for workload prediction and positioning.

As it can be seen, the most frequent metrics relate to resource utilisation (28.2% of papers), overhead (24.3%) and fault handling (23.1%). When ignoring the grouping discussed above, the most common individual metric is response time (18.1% of papers) followed by communication overhead (11.3%), execution time (11.8%), energy/power consumption (9.2%) and throughput (9.2%). It is important to note that four papers (1.7%) consider and evaluate metrics specific

TABLE 3. Frequency of performance metrics (N = 238).

Performance Metric	Count
Resource utilisation	**67**
** Energy/power consumption	22
** CPU utilisation	18
** Cloud resource utilisation	12
** Memory usage	9
** Server resource utilisation	6
Overhead	**58**
** Communication overhead	27
** Computation overhead	11
** Memory overhead	8
** Monitoring overhead	2
** Overhead (other)	10
Fault handling	**55**
** Recovery time	20
** Fault detection	14
** Fault recovery performance	11
** Reliability	10
Response time	43
Execution time	28
Throughput	22
Delay/Latency	19
Utility	17
Time (other)	12
Financial cost	12
Packet delivery performance	12
Attack detection	11
SLA violations	9
Load balancing performance	9
Quality of service	7
Revenue/Profit	7
Configuration time	7
Accuracy	7
Number of nodes/agents	7
Number of virtual machines	6
Number of routing hops	6
Task completion	5
Configuration performance	5
Initialisation time	4
Video quality	4
Number of admitted sessions	4
Convergence time	3
User experience	3
Service availability	3
Number of replicas	3
Time in valid state	2
Downtime	2
Service discovery time	2
Computing performance	2
Adaptation probability	2
Adaptation quality (time, goals)	2
Number of notifications	2
Other	33
Not specified	7

to the adaptation process, namely, adaptation runtime and the number of adaptation goals achieved. As expected, some papers, 2.9%, do not explicitly specify metrics, due to them being qualitative or case study papers.

Table 3 shows similarities between the most common metrics and the most frequent domains as presented in Table 2. Many of the metrics relate to performance and overhead in computing applications and networked systems, which is reflective of the domains considered in our paper set.

A. RQ1: APPROACHES EVALUATED IN CONTESTED AND RESOURCE-CONSTRAINED ENVIRONMENTS

In this analysis, we consider papers that discuss systems that had some form of *resource constraints* or systems where the information necessary to make a decision is not perfect, that is, it is either *incomplete* or *stale*. As discussed above, *incomplete* information means that components cannot observe the state of all other system components or sensors. This may be due to an attack, to the presence of system failures or to the nature of the operating environment (e.g., a data centre with frequent power failures). In the case of *stale* information, we consider papers where the systems discussed have an understanding that the information employed might become out of date and employ some mechanism to receive or construct updated information. We found a total of 132 papers that meet this criteria, as shown in Table 4. Since these papers form the main focus of this systematic literature review, we refer to them as the *target subset* henceforth.

TABLE 4. Systems working in contested & resource-constrained environments in target subset (N = 132).

Contested	Resource-Constrained	Non Resource-Constrained
Not contested	96	-
Incomplete	4	22
Stale	2	8

As it can be seen in Table 4, the majority of the papers (72.7% of the target subset) consider that complete information from system sensors as well as other system components is always available to make a decision. Only six systems consider both resource-constrained and contested environments, representing 4.5% of the target subset. This low number could be a consequence of the application domain and specific goals of each particular system, in that failures and security concerns are not specifically considered, as shown also in Figure 3. However, failures are prevalent in most application domains and their consequences are significant [10]–[12] and, as such, there is a need for self-adaptive systems to consider this aspect as well. We discuss the papers that consider resource-constrained and contested environments, referred to as the Contested and Resource-Constrained set (CRC), in detail in Section V-C.

An overview of the application domains discussed in the target subset is given in Figure 6. As expected from our analysis in Table 2, the most frequent application domain is cloud computing. Other application domains that traditionally have resource congestion or are subject to attacks can also be found on this list, namely, media and data streaming, communication, autonomous vehicles, crisis management, data centres and parallel computing. As above, we list under ''Other'' application domains that appear only once, namely, Automated Services, Spacecraft and Aircraft, Systems-on-Chips, Electrical Power Systems, and Surveillance.

1) UNDERSTANDING RESOURCE CONGESTION

To better understand the specific constrained resources considered in the 102 papers captured in the first column of Table 4, we perform a grounded theory analysis as discussed in Section IV-D. In the open coding step, we tagged all congested resources listed in the F9 data item. In most cases, the tag was the same as the F9 data item itself, for example where ''cloud resources'' was used. In other cases, a list of congested resources was tagged with a common term to describe them all. For example, ''CPU, memory, disk space'' was tagged using the term ''computational resources'', whereas the data item ''CPU'' was tagged using the term ''CPU'' (and the paper was examined to ensure that it only referred to CPU as a congested resource). The tags were then refined to group similarities, e.g., ''cloud computing'', ''virtual machine'', ''cloud resources'' were then re-tagged as ''virtualised resources''. An overview of the congested resources is found in Table 5.

TABLE 5. Frequency of congested resources in target subset (N = 132).

Congested Resource	Count
Computational resources	30
Virtualised resources	25
Network link capacity	14
Bandwidth	13
CPU	13
Cell radio resources	8
Battery	2
FPGA	2
Software entity utilisation	2
Cell agents	1
Conveyor belt	1
Load injection machines	1
Power network capacity	1
Quality of Service (QoS)	1
Road usability	1
SMS network load	1
Storage	1
Submarine transmission capacity	1

In a similar manner, we analysed the types of resource congestion mitigation mechanisms employed by the selected approaches encoded in data item F10 and present our results in Table 6.

Resource provisioning mechanisms include those mechanisms where new computational, virtualised or other resources are made available to relieve the congested resource. They are by far the most frequently used mechanisms. *Load balancing* mechanisms include mechanisms where load is re-distributed across existing resources, and no new resources are used. *Reconfiguration* mechanisms are those where the system adapts by changing some or all of its configuration parameters, e.g., by changing some threshold

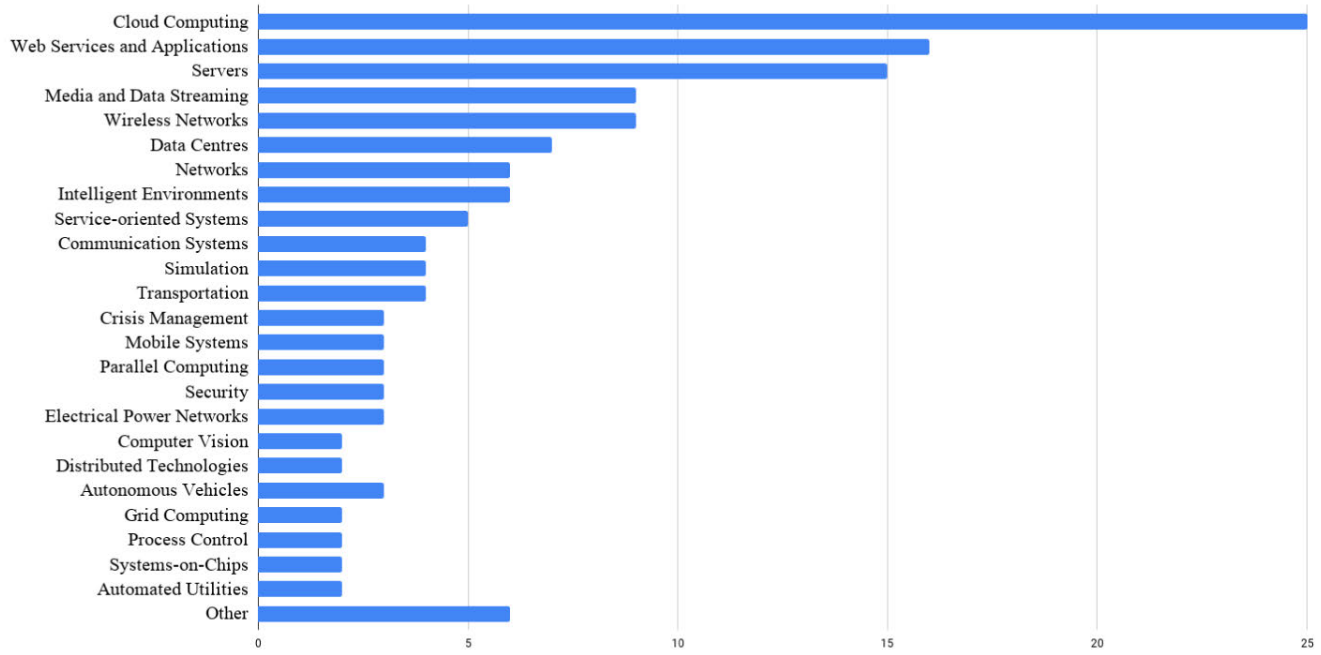


FIGURE 6. Frequency of application domains in the target subset (N = 132).

TABLE 6. Frequency of congestion mitigation mechanisms in target subset (N = 132).

Congestion Mitigation Mechanism	Count
Resource provisioning	36
Reconfiguration	22
Load balancing	19
Optimisation & machine learning	13
Dropping quality	10
Migration	5
Deactivation	3
Adaptive scheduling	3
Spot pricing	1
Hardware virtualisaton	1
Dynamic routing tables	1
Auctioning	1

parameters. *Optimisation & machine learning* mechanisms are mechanisms where a particular optimisation or machine learning algorithm is used to determine the best configuration setting, for example an evolutionary algorithm used to determine the best resources to use for provisioning, a heuristic employed in load balancing, or a machine learning algorithm that determines when resources should be provisioned [52]. This category is subsumed by resource provisioning, load balancing and reconfiguration, however, only in 13 papers (9.8% of the target subset) were optimisation techniques explicitly mentioned. The majority of the papers considering optimisation techniques used evolutionary or swarm-based techniques to optimise their objective, and none of the papers used machine learning techniques to learn to adapt to change, highlighting future avenues of research. *Dropping quality* is a mechanism whereby the system makes a decision to reduce the quality of its services, e.g. either by displaying text and no images on a website page or by reducing the quality of the video streamed [53].

B. RQ2: METRICS USED FOR THE EVALUATION IN CONTESTED AND RESOURCE-CONSTRAINED ENVIRONMENTS

Table 7 shows the performance metrics identified for the papers in the target subset.

Similarly to the results in Table 3, we see that resource utilisation, overhead and fault handling are the most common performance metrics. In addition, we observe a lack of metrics focused on analysing the suitability of the employed solution in the context of incomplete or stale information. This could take the form of confidence values or intervals and other information quality metrics.

C. RQ3: LIMITATIONS OF SELF-ADAPTIVE SYSTEMS RUNNING IN CONTESTED AND RESOURCE-CONSTRAINED ENVIRONMENTS

To provide insight into the limitations of existing systems operating in resource-constrained and contested environments, we list, in Table 8, the stated limitations identified in the target subset (N = 132). We then present a more in-depth analysis of the six papers that consider systems that operate in resource-constrained *and* contested environments (the CRC set).

Table 8 lists the limitations as identified by each paper's authors. Similar to the above, we perform a qualitative analysis of the stated limitations and use grounded theory to develop groups - for example the **Limited Analysis** group is an aggregation of several distinct types of analytical issues

TABLE 7. Frequency of performance metrics in target subset (N = 132).

Performance Metric	Count
Resource utilisation	**49**
** Energy/power consumption	16
** CPU utilisation	12
** Cloud resource utilisation	11
** Memory usage	6
** Server resource utilisation	4
Overhead	**36**
** Communication overhead	17
** Computation overhead	5
** Memory overhead	2
** Monitoring overhead	2
** Overhead (other)	10
Response time	31
Fault handling	**23**
** Recovery time	11
** Fault detection	6
** Fault recovery performance	3
** Reliability	3
Throughput	18
Delay/Latency	15
Execution time	13
Financial cost	11
Utility	11
SLA violations	9
Time (other)	9
Packet delivery performance	7
Load balancing performance	6
Number of virtual machines	6
Attack detection	5
Configuration performance	5
Quality of service	4
Revenue/Profit	4
Accuracy	4
Number of routing hops	4
Configuration time	3
Number of nodes/agents	3
Number of admitted sessions	3
Initialisation time	2
Video quality	2
Service availability	2
Number of replicas	2
Downtime	2
Computing performance	2
Number of notifications	2
Other	19
Not specified	2
Adaptation runtime	1
Adaptation goals satisfied	1

present in 20 of the target subset papers. Similarly, the **Not implemented** category includes approaches that are both completely theoretical (such as in Chainbi *et al.* [54]) and partially implemented (such as in Li *et al.* [55]). The **Lack of artificial intelligence** category refers to approaches where authors considered that the approach could be improved through the application of artificial intelligence techniques, such as [56], where the inclusion of neural-net based behavioural analysis would augment the proposed system. The frequencies of each reported limitation are identified in Table 8, where the *Other* limitation category captures limitations that could not be aggregated into abstract classes. They tended be specific or idiosyncratic to their respective approaches.

TABLE 8. Stated limitations in the target set (N = 132).

Limitation	Count
Limited analysis	20
Lack of scalability	15
Not implemented	8
Expert knowledge required	6
Unreliable performance	6
Lacking support for heterogeneous platforms	4
Centralised	4
Lack of fault tolerance	3
Poor performance	3
Lack of stability	3
Validation	3
Suboptimal agent performance	2
Limited experiment scale	2
Deployment	2
Inefficient optimisation	2
High overhead	2
Slow start	2
Domain-specific training data required	2
Suboptimal adaptation	2
Lack of artificial intelligence	2
Lacks Formalisation	2
Lacks Verification	2
Lacks Sensor Sensitivity	2
Lacks Security	2
Lacks Healing	2
Other	22

70% of the papers inside the target subset discussed their limitations, with the most frequent limitations being the need for more extensive analysis (which occurred in 20, or 15.15%, of the papers inside the target subset) and the need to consider and develop for scalability (which occurred in 15, or 11.4%, of the papers inside the target subset). The third most frequent limitation corresponded to systems that are yet to be implemented in practice, which occurred in 8 papers (or 6.1% of the target subset). Altogether 125 individual limitations were identified by authors across the target subset.

The *Limited Analysis* category from Table 8 includes papers missing the evaluation of key performance metrics [28], [57], papers that contained limited comparisons of system performance to contemporary solutions in the same problem space [58], papers that did not consider alternative design decisions [59], [60] and papers where the performance of the system was difficult to measure [7], [8]. The large number of papers that reported *Limited Analysis* as a limitation is concerning, as it may indicate that systems operating in or designed for contested or resource-constrained environments could be better evaluated. The performance of these systems is especially critical for scenarios involving human well-being [7], [61]. This may also indicate that the research space may require formalisation in terms of the analysis that is expected to accompany studies on self-adaptive systems. This will also allow systems to evaluate that they have not met their adaptation goal, identify potential causes in the environment, information uncertainty, and employed self-adaptation methods, and devise strategies to alleviate this.

This is a multi-faceted problem in itself, as within the system there must be an understanding of the impact on uncertainty itself, as well as an understanding of the effect that the

uncertainty mitigation strategies will have on the goals of the self-adaptive system. This requires the self-adaptive system to have an internal model of uncertainty and also to be able to reason over the benefit of an uncertainty reduction and evaluate its application performance gain. Moreno *et al.* [62] propose that a self-adaptive system or system component identify when uncertainty is present, and apply several uncertainty reduction tactics to reduce it. The uncertainty reduction tactics are defined based on several identified uncertainty causes, such as simplifying assumptions, noise, model drift, context, human in the loop and decentralisation. However, the impact of each reduction tactic needs to be properly understood before it is applied.

Table 9 shows the limitations of the papers in the CRC set.

TABLE 9. Stated limitations for papers in the CRC set (N = 6).

Limitation	Count	Papers
Limited Analysis	2	[60], [63]
No/Limited Scalability	2	[52], [64]
Limited Experiment Scale	1	[63]
Race Conditions Present	1	[5]
Slow-Start	1	[52]
No/Limited Fault-Tolerance	1	[65]

D. UNDERSTANDING RESOURCE-CONSTRAINED AND CONTESTED ENVIRONMENTS

We present in the following an overview of the approaches from the CRC set, with a focus on understanding how they operate in contested and resource-constrained environments.

1) STALE INFORMATION

[60] Reference propose to address the problem of video streaming in MANET networks, which is slowed down by the necessary use of cryptography. To address this, they propose a QoS aware adaptive security scheme that counters the effects of delay overhead by adapting cryptography and specific streaming properties. The scheme defines four adaptive mechanisms, that are combinations of allowing varying frames per second, requesting the encryption of specific types of video frames, and defining only specific frames as mandatory to transmit. The schemes are adapted based on stale feedback information received from the receiver, which is regularly updated as feedback packets are received. The approach is promising and the evaluation extensive, however the evaluation considers a small network (n = 36 nodes) in a single configuration and does not show the adaptation process as it switches between schemes. Instead, the evaluation presents scenarios where a specific adaptation scheme is employed. In addition, the considered MANET is stationary and thus mobility is not considered in the evaluation. This is critical when information is stale, as the adaptation process might not be instantaneous and thus the end result might present an adaptation for network conditions or topologies that no longer exist due to node mobility.

To address the challenges of provisioning QoS over wireless sensor networks, Berrayana *et al.* [64] propose XLEngine, a cross-layer autonomic architecture that makes optimisations based on local and network-wide knowledge. The approach employs local information obtained about a node's neighbours to construct a model of the network information. Only links that have had a heartbeat recently are considered. The analysis shows the benefits to end-to-end delay and throughput in a simulated network of between 10 and 60 nodes. The analysis does not consider larger networks. Moreover, the evaluation does not consider frequent node failures or nodes joining or leaving the wireless network (high churn rate), which is a common phenomenon that can have significant effects on the quality of information that is constructed.

2) INCOMPLETE INFORMATION

An approach that addresses high churn rate is proposed in [5], where cooperative autonomic components self-organise into a dynamically created overlay network. Information is shared locally with neighbours, ensuring that eventually each component gain access to global system information. The autonomic managers manage a set of resources by defining an overlay network of autonomic managers (AMs) that changes as the AMs join and leave the system, as in a peer-to-peer system. The approach is resilient to churn through dynamic resource claiming and dynamic neighbourhood building. Dynamic resource claiming happens once an AM detects failures among its AM neighbours; it informs the domain registry and reclaims resources that have become unmonitored because of the failure. Dynamic neighbourhood building happens when an AM fails or gracefully leaves, and the remaining neighbouring AMs link themselves to the leaving AMs neighbours. The proposed approach is lightweight and is evaluated both through an analytical model and a simulation. However, the analysis is presented under simple network conditions with only one massive churn event, where a large number of nodes leave the system. Furthermore, different network topologies are not evaluated.

Another approach that looks at streaming applications in sensor networks [63] similarly uses local information from neighbours to build global information about the network. The approach proposes to use the Lotka-Volterra competition model to control the rate of traffic flow from each node. The rate of every flow is regulated in order to prevent congestion in the entire wireless network. Each node is in charge of self-regulating and self-adapting the rate of its flow, and all flows compete for available buffer capacity at their one-hop away receiving node. Similar to the approaches discussed above, only a small number of nodes (20) are used to evaluate scalability and churn is not considered.

Ranjan and Zhao [65] address service provisioning in cloud computing infrastructures by proposing an integrated framework that facilitates peer-to-peer management of cloud system components for providing end users with fault tolerant and reliable services. Their approach uses Distributed Hash Tables (DHTs) that provide a peer-to-peer routing structure for interconnection of cloud system components. A DHT

overlay is extended with cloud service monitoring, discovery and load-balancing capabilities, and a heartbeat mechanism is used to determine the status of the various resources. The approach evaluation is extensive, and its benefits highlighted under various network conditions and system parameters. A high churn scenario is however not evaluated, but the reliance on a DHT should alleviate this.

In another cloud computing approach, Jamshidi *et al.* [52] propose FQL4KE, an online fuzzy self-learning mechanism that adjusts and improves autoscaling policies at runtime. The approach is implemented and evaluated both in OpenStack and Microsoft Azure platforms and analysis shows its sensitivity to the reinforcement signal, which takes a long time to arrive as there is a discrepancy between the scaling decision, which takes up to 10 seconds, and the scaling action, which takes up to 10 minutes. Moreover, the analysis only considers a limited set of possible scaling actions.

VI. DISCUSSION

Our analysis looked at identifying self-adaptive and/or autonomic approaches that considered resource-constrained and contested environments and at understanding the trade-offs they perform as well as how they are evaluated. Contested and resource-constrained environments are found by military land vehicles in the battlefield or any system operating in unexpected conditions, such as a distributed system under unexpected load or a system working in a crisis scenario.

A. PERFORMANCE AND EFFECTIVENESS METRICS

The ability to evaluate system self-management in contested and resource-constrained environments is important because it provides a means to demonstrate its effect in supporting system objectives and it may be used as feedback by control and learning systems applied in this space.

Our analysis identified that the majority of metrics used focused on computing and network resource utilisation as well as computing resource overhead. While we identified metrics related to fault tolerance, we found no experiments that looked at system resilience in the presence of high and frequent fault rates. Despite availability being identified as an important quality for dynamic software systems [66], no metrics or evaluations focused on the availability of core system functionality. In addition, we found no metrics or analyses to evaluate the suitability of approaches running in environments where incomplete or stale information was a constraint. Experiments could focus on evaluating the accuracy of the global snapshot needed to make decisions or could evaluate the outcome given environments with varying levels of stale or incomplete information. This analysis was missing from all papers, including those few that considered incomplete information.

Adaptability is a desired system property yet none of the experimental analyses considered a comparison between proposed self-adaptive approaches or techniques. A critical open research gap remains in evaluating self-adaptive approaches and in comparing their benefits and trade-offs in achieving a specific outcome. Our findings confirm the findings from [41], which concluded that in their studied papers the evaluation of adaptive systems is generally not addressed explicitly in either the managed or managing system and adaptation metrics are generally not considered. The evaluation of self-adaptive systems in the 238 papers in our set tended to focus on aspects specific to the goals of the system where self-adaptability was being applied. This includes aspects such as relevant performance metrics and overheads within their context of operation. While this contributes to understanding the value of a self-adaptive system, it offers little in the way of providing an understanding of how adaptability itself might be evaluated. This presents as a significant gap in the literature, which if addressed would allow, for example, the comparison between approaches on the basis of adaptability and the identification of the extent of adaptability that various approaches would enable. An experimental framework to compare between different self-adaptive techniques is yet to be designed and implemented, resulting in a lack of understanding of mechanisms of adaptation and their suitability for different application domains and thus limiting the use of self-adaptive approaches in domains with critical requirements and long purchase cycles such as the military.

An important consideration relating to distributed and decentralised self-management is the ability to globally evaluate systems and their adaptability. This presents a complex problem as the evaluation of a single component may not be representative of the global situation due incomplete or stale information. In order to address this problem, metrics that are ergodic in nature are required. Similarly, mechanisms are needed by which local metrics can be aggregated and shared efficiently while considering bandwidth and other relevant constraints. Such an approach will be important to support any evaluation of the adaptability of distributed and decentralised self-managing systems.

B. CRITICAL SYSTEM PROPERTIES

The desired system properties for self-adaptive systems running in contested and resource constrained environments are adaptability, reliability, fault tolerance and security, yet few of the papers considered them in their evaluation, namely, in 17%, 4%, 13% and 4.8% of papers respectively. We also found that few papers (less than 3%) considered self-CHOP properties together. An explicit consideration of self-organisation, self-healing and self-protecting together would allow approaches to move towards decentralised solutions, thus providing a significant increase in fault tolerance and resilience. It is unclear from the design considerations and evaluations present in the papers whether this is due to specific trade-offs encountered in the design or to the specifics of the application domains. As the most popular domains include cloud computing, web services and applications and wireless networks, where faults are frequent and costly, it is more likely that specific trade-offs limited the design, however these are not documented nor evaluated.

Evolvability, defined as the ability to evolve in dynamic situations [67], is another important property for self-adaptive systems in changing and unpredictable environments. As opposed to adaptability, which may range from an immediate response to a fault to adjusting the goals of a system when faced with new information, evolvability enables a self-adaptive system to change its behaviour over time by adjusting its architecture and components and introducing novel functionality to meet operational needs and to excel in its environment. This is important for self-adaptive systems in environments such as the military domain due to the vast potential of operating contexts that cannot easily be accounted for during self-adaptive system design. A small number of papers (11%) considered this property in their evaluation, such as [68] where adaptation effectiveness of the considered approaches was evaluated in terms of key quality constraints for a particular application. In the future, a heightened focus on evolvability would support the development of highly complex self-adaptive systems for dynamic environments.

C. CONTESTED AND RESOURCE-CONSTRAINED ENVIRONMENTS

An aspect of functioning in contested and resource-constrained environments is that self-adaptive systems should consider managed systems that are distributed and managing systems that are decentralised in terms of computation and decision making. Papers that evaluate these types of systems provide a greater insight to relevant metrics and limitations, in line with operation in a military battlespace. As presented in Section V, the majority of managed systems in the reviewed papers were distributed, but managing systems were predominantly centralised. This highlights a significant research gap in the field, as centralised solutions are not suitable in environments where bandwidth is limited and failures are frequent. It should be noted that the majority (five) of the six CRC papers considered distributed managed systems and decentralised managing systems, however these solutions represent only 2% of our set.

Another relevant characteristic of the approaches considered in this review is how they address resource congestion. We found that the most common ways to address resource congestion are resource provisioning, reconfiguration, and load balancing. Only reconfiguration and load balancing are possible in systems where resources are finite (so no new resources can be allocated), such as in the majority of military systems or in application domains where resource purchasing cycles are static and lengthy. An exception is cognitive radio systems, in which radio modulation schemes can be changed dynamically, trading off resilience and capacity, thereby providing an avenue to generate additional resources (capacity) and provision accordingly. Very few papers (4.6%) considered environments where the information about the system components was not readily available. The mechanisms employed in these cases were heartbeat or refresh based. Only six papers considered resource-constrained *and* contested environments and most had limited analysis in terms of the metrics and scenarios considered. Furthermore, as environments are dynamic and their constraints evolve, there is a lack of understanding of the mechanisms of adaptation and their suitability to different environmental constraints, further highlighting the experimental analysis identified above.

The nature of the environment, where information might not reach its destination, implies nodes have an incomplete view of the entire system and thus have to make decisions based on local snapshots. The constrained nature of the computing platform also implies that adaptation goals might not always be entirely met. In this case, it is critical that the self-adaptive system is able to evaluate that the adaptation goal has not been met, identify reasons, and change its strategy. None of the works in the CRC set consider this perspective.

Another important consideration in the design of any self-adaptive system that operates in a contested and resource-constrained environment is how and whether adaptation is modelled from a theoretical and/or formal perspective. Having a theoretical or a formal understanding of how self-adaptation should happen offers increased confidence in the self-adaptive system as the system can be theoretically validated or formally verified. It also also allows for precise interventions as the system matter experts have a better understanding of the effect of specific input or environment changes on the self-adaptation outcome. Increased confidence in the system and the ability to quickly and precisely change the outcome or choice of self-adaptation is critical, in particular in systems that run in crisis situations or in military operations.

The work of Jamshidi *et al.* [52] enhances a MAPE-K loop with online learning and extensively models the dynamics of the fuzzy self-learning mechanism using an analytical approach. Similarly, Antoniou and Pitsillides [63] analytically model the Lotka-Volterra competition model employed. Reza and Barbeau [60] presents an extensive analytical model of adaptation both within the security focused adaptation and in the video focused adaptation. The authors of [65] present a theoretical evaluation of the time complexity of the adaptation process but do not model the adaptation process itself. Similarly, Xu *et al.* [5] extensively model the AM network using the conceptual framework and notations from complex network theory and evaluate the cost of communication and the cost of building the network. They also evaluate the cost of adaptation as local load adjustment. Berrayana *et al.* [64] do not formally or theoretically model adaptation.

As we can see, adaptation is theoretically modelled in some form in the majority of papers in the CRC set. However, a MAPE-K loop is only considered by a single work, and only from a high level perspective, without explicitly modeling or considering each layer of the loop. This lack of theoretical modeling, together with the lack of extensive evaluation as discussed above, pose significant challenges for the adoption

of these techniques in environments where the trust of user in a proposed solution and the confidence on specific outcomes are of paramount importance.

D. THREATS TO VALIDITY

The main threats to validity of this survey relate to bias in the selection or exclusion of a paper. To mitigate this bias, the authors initially conducted five calibration sessions, where ten papers were discussed and assessed in each round. Weekly meetings were also held, where each researcher discussed papers they were unsure about, in terms of whether they aligned with the inclusion criteria. In the classification and data collection stage, papers were assigned in a reverse order to researchers, thus ensuring that each paper was reviewed twice for inclusion in this study. Another threat relates to data collection where poor descriptions in the papers might have resulted in inference of information. When this occurred, researchers discussed the papers during the regular weekly meetings.

Another potential bias is introduced by the search string, which will inherently, through the presence of the word 'system', 'architecture', or 'design' not consider papers where these words are not present. This might focus the search on network, systems, and security papers and remove some application domains and thus not completely cover all fields where adaptation is considered. However, since our focus is on adaptive and autonomic architectures, systems, or designs, we believe that the coverage is sufficient, in particular to allow us to better understand adaptation in contested and resource constrained environments.

E. RESEARCH GAPS

Several research gaps can be identified from the discussion above, namely, (i) the lack of formalisms, techniques and frameworks for evaluating self-adaptation, (ii) the need of metrics and dissemination mechanisms that permit the use of local and global metrics to make self-adaptation decisions, (iii) the need to consider additional properties in order to design resilient self-adaptive systems, and (iv) the lack of machine learning techniques employed in self-adaptive solutions. We discuss these in detail in this section. Firstly, there is a critical need for extensive evaluation of self-adaptation in systems operating in contested and resource-constrained environments, where failures are frequent. Our analysis found that studies used metrics related to fault tolerance, but there were no experiments that looked at system resilience in the presence of high and frequent fault rates. We found that no papers presented an analysis of self-adaptation given environments with varying levels of stale or incomplete information. Similarly, we found no framework, formalism, or standard that would allow researchers to compare between self-adaptation approaches. An experimental framework to compare between different self-adaptive techniques is yet to be designed and implemented, resulting in a lack of understanding of mechanisms of adaptation and their suitability for different application domains. This framework would also need to rely on theoretical, formalised representations of self-adaptation in the managing system and of the specific constraints within the managed system and its environment.

Secondly, self-adaptation, regardless of the way in which it is implemented, requires decisions to be made about the environment and context in which the system is operating and thus inherently relies on accurate and effective local and global metrics. The nature of the environment, where information might not reach its destination, implies nodes have an incomplete view of the entire system and thus have to make decisions based on local snapshots. Mechanisms are needed by which local metrics can be aggregated and shared efficiently while considering bandwidth and other relevant constraints.

Thirdly, several system properties, such as evolvability and the tuple of self-organisation, self-healing and self-protecting are not considered. Evolvability would ensure that self-adaptation is dynamic, changing with the environment or with the available information. Self-organisation, self-healing and self-protection considered together with self-adaptation would allow for managed systems to be implemented as distributed systems effectively. Addressing both these gaps will allow for the implementation of resilient self-adaptive systems that can respond well to significant changes in the environment.

Fourthly, our analysis found that machine learning techniques, despite their promise, were not explicitly considered by the papers surveyed. Machine learning techniques can address several of the research gaps highlighted above, such as making decisions based on incomplete or uncertain data or metrics, or including multiple desired properties within the adaptation goal. They allow for decisions to be made online, without prior training and in the presence of uncertainties and incomplete data. In addition, excluding the training runtime, the decision runtime of a machine learning model is significantly smaller than that of other optimization techniques. Machine learning approaches pose also significant challenges, including explainability, sensitivity to specific inputs, and ease of adaptation to dynamic contexts, and as such there is a critical need to study their application to self-adaptive systems.

VII. CONCLUSION

Self-adaptive systems are a promising approach for addressing the uncertainty and complexity of today's software environments. However, existing software environments are also prone to attacks or failures, resulting in poor availability of information about all system components. In addition, resource congestion frequently occurs in environments where resources are limited or large amounts of information are exchanged between critical components that are computationally expensive.

In this systematic literature review of self-adaptive systems papers, we identified works that considered contested or resource-constrained environments. Our analysis showed significant progress in the field, in particular with respect to

the breadth of application domains and the implementation of self-CHOP properties, but also identified critical research gaps. We found that although the majority of the managed systems were distributed, most managing systems proposed centralised solutions, thus representing a critical point of failure and reducing the applicability of these solutions to adversarial environments and/or those where failures are frequent and costly.

As large-scale, failure prone systems are becoming ubiquitous, our review identifies that there is a critical need for decentralised self-adaptive managing systems that consider resource congestion and the lack of complete readily available information. In addition, in order for self-adaptive systems to be employed in domains with critical operation constraints such as military operations, it is paramount that the evaluation of these systems is extensive, under a variety of scenarios and configuration parameters, and that metrics to identify adaptability itself are investigated and employed. Our analysis found that few evaluations showed the benefits of a self-adaptive solution over a baseline approach and that an evaluation framework or benchmark for self-adaptive solutions is yet to be proposed. We also found that there is limited understanding of the benefits and trade-offs of employed self-adaptive techniques in achieving specific outcomes and the question of which self-adaptive technique is best suited for a particular purpose remains unanswered. Lastly, adaptation is only rarely explicitly modeled, either theoretically or formally, and thus the system's behavior cannot be formally verified or validated. This offers few guarantees about the expected behavior of the system and can have implications about its use in practice, in particular in crisis situations or military operations.

With the need to evaluate and formally model self-adaptation and managed systems comes the need to consider more than self-adaptation in order to ensure that the system is resilient to failures and uncertainties. This includes considering evolvability, self-organisation, self-healing, and self-protection holistically in the design of the system. There is a research gap in understanding how this can be achieved with minimal compromise or trade-offs. Lastly, a critical research gap remains in the use of machine learning techniques to improve self-adaptation decisions and actions, which has not been explored yet. All these are research challenges with significant potential to further widen the reach of self-adaptation across a multitude of application domains.

REFERENCES

[1] J. Kramer and J. Magee, "Self-managed systems: An architectural challenge," in *Proc. Future Softw. Eng. (FOSE)*, May 2007, pp. 259–268.
[2] R. Calinescu, S. Gerasimou, and A. Banks, "Self-adaptive software with decentralised control loops," in *Proc. Int. Conf. Fundam. Approaches Softw. Eng.* Berlin, Germany: Springer, 2015, pp. 235–251.
[3] R. D. Lemos *et al.*, "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*. Berlin, Germany: Springer, 2013, pp. 1–32.
[4] A. Gadafi, A. Tchana, D. Hagimont, and L. Broto, "Autonomic energy management in clusters," in *Proc. 1st Workshop Green Comput. (GCM)*, New York, NY, USA, 2010, pp. 16–21.
[5] J. Xu, M. Zhao, and J. A. B. Fortes, "Cooperative autonomic management in dynamic distributed systems," in *Stabilization, Safety, and Security of Distributed Systems*. Berlin, Germany: Springer, 2009.
[6] Z. Sui, M. Malensek, N. Harvey, and S. Pallickara, "Autonomous orchestration of distributed discrete event simulations in the presence of resource uncertainty," *ACM Trans. Auton. Adapt. Syst.*, vol. 10, no. 3, pp. 18:1–18:20, Sep. 2015.
[7] C. An, Y. Luo, and A. Timm-Giel, "Adaptive routing in wireless sensor networks for fire fighting," in *Information and Communication Technologies*. Berlin, Germany: Springer, 2012.
[8] B. Lin, A. I. Sundararaj, and P. A. Dinda, "Time-sharing parallel applications through performance-targeted feedback-controlled real-time scheduling," *Cluster Comput.*, vol. 11, no. 3, pp. 273–285, Sep. 2008.
[9] A. Taherkordi, Q. Le-Trung, R. Rouvoy, and F. Eliassen, "WiSeKit: A distributed middleware to support application-level adaptation in sensor networks," in *Distributed Applications and Interoperable Systems*. Berlin, Germany: Springer, 2009.
[10] M. Farshchi, J.-G. Schneider, I. Weber, and J. Grundy, "Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis," in *Proc. IEEE 26th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2015, pp. 24–34.
[11] *Downtime, Outages and Failures—Understanding their True Costs*. Accessed: Sep. 20, 2020. [Online]. Available: http://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html
[12] *The Cost of Downtime*. Accessed: Sep. 20, 2020. [Online]. Available: http://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/
[13] G. Guerrero-Contreras, J. L. Garrido, S. Balderas-Diaz, and C. Rodriguez-Dominguez, "A context-aware architecture supporting service availability in mobile cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 956–968, Dec. 2017.
[14] E. F. Coutinho, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "An architecture for providing elasticity based on autonomic computing concepts," in *Proc. 31st Annu. ACM Symp. Appl. Comput. (SAC)*, 2016, pp. 412–419.
[15] *Australian Government 2009—Department of Defence, Defence White Paper, Paragraph 9.40*. Accessed: Sep. 20, 2020. [Online]. Available: http://www.defence.gov.au/whitepaper/2009/docs/defence_white_paper_2009.pdf
[16] *Australian Government 2013—Department of Defence, Defence White Paper, Paragraph 8.72*. Accessed: Sep. 20, 2020. [Online]. Available: http://www.defence.gov.au/whitepaper/2013/docs/WP_2013_web.pdf
[17] *Australian Government 2016—Department of Defence White Paper, Paragraph 4.52*. Accessed: Sep. 20, 2020. [Online]. Available: http://www.defence.gov.au/WhitePaper/Docs/2016-Defence-White-Paper.pdf
[18] D. Abdulmasih, P. Oikonomidis, R. Annis, E. Stipidis, and P. Charchalakis, "Operational integrity monitoring for military vehicle's integrated vetronics architecture," in *Proc. 6th IET Int. Conf. Syst. Saf.*, 2011, pp. 1–6.
[19] *NATO Generic Systems Architecture (NGVA) for Land Systems*, Standard STANAG NATO. 4754, 1st ed., 2016.
[20] M. S. Vassiliou, J. R. Agre, S. Shah, and T. MacDonald, "Crucial differences between commercial and military communications technology needs: Why the military still needs its own research," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2013, pp. 342–347.
[21] G. Elmasry, "A comparative review of commercial vs. tactical wireless networks," *IEEE Commun. Mag.*, vol. 48, no. 10, pp. 54–59, Oct. 2010.
[22] D. Sinreich, "An architectural blueprint for autonomic computing," Autonomic Comput., IBM, Armonk, NY, USA, White Paper 7, 2006, pp. 1–6, vol. 31.
[23] TASDCRC. (2019). *Trusted Autonomous Systems Defence Collaborative Research Centre*. Accessed: Jan. 13, 2021. [Online]. Available: https://tasdcrc.com.au/
[24] R. Sale, "Trusted autonomous systems defence CRC," Defence Sci. Technol. Group, Australia, Tech. Rep. 72179, 2018.
[25] M. Sloman, E. Asmare, A. Gopalan, E. Lupu, and N. Dulay, "Adaptive self-management of teams of autonomous vehicles," Univ. Durham, Durham, U.K., Tech. Rep. 4334, 2008.
[26] D. J. Nowak, G. B. Lamont, and G. L. Peterson, "Emergent architecture in self organized swarm systems for military applications," in *Proc. 10th Annu. Conf. Companion Genet. Evol. Comput.*, 2008, pp. 1913–1920.
[27] A. S. Nascimento, C. M. Rubira, and F. Castor, "Arcmape: A software product line infrastructure to support fault-tolerant composite services," in *Proc. IEEE 15th Int. Symp. High-Assurance Syst. Eng. (HASE)*, Jan. 2014, pp. 41–48.

[28] A. J. da Silva and E. F. V. Rebello, "A hybrid fault tolerance scheme for EasyGrid MPI applications," in *Proc. 9th Int. Workshop Middleware Grids, Clouds e-Sci. (MGC)*, New York, NY, USA, 2011, pp. 4:1–4:6.

[29] G. Tyson, P. Grace, A. Mauthe, and S. Kaune, "The survival of the fittest: An evolutionary approach to deploying adaptive functionality in peer-to-peer systems," in *Proc. 7th Workshop Reflective Adapt. Middleware*, 2008, pp. 23–28.

[30] A. Brocco, "Ozmos: Bio-inspired load balancing in a chord-based P2P grid," in *Proc. 3rd Workshop Biologically Inspired Algorithms Distrib. Syst. (BADS)*, New York, NY, USA, 2011, pp. 9–16.

[31] I. B. Rodriguez, K. Guennoun, K. Drira, C. Chassot, and M. Jmaiel, "Implementing a rule-driven approach for architectural self configuration in collaborative activities using a graph rewriting formalism," in *Proc. 5th Int. Conf. Soft Comput. Transdisciplinary Sci. Technol.*, 2008, pp. 484–491.

[32] T. Patikirikorala, A. Colman, J. Han, and L. Wang, "A systematic survey on the design of self-adaptive software systems using control engineering approaches," in *Proc. 7th Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst.*, 2012, pp. 33–42.

[33] H. Muccini, M. Sharaf, and D. Weyns, "Self-adaptation for cyber-physical systems: A systematic literature review," in *Proc. 11th Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst.*, 2016, pp. 75–81.

[34] M. Salama, R. Bahsoon, and N. Bencomo, "Managing trade-offs in self-adaptive software architectures: A systematic mapping study," in *Managing Trade-Offs in Adaptable Software Architectures*. Amsterdam, The Netherlands: Elsevier, 2017, pp. 249–297.

[35] S. Mahdavi-Hezavehi, V. H. Durelli, D. Weyns, and P. Avgeriou, "A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems," *Inf. Softw. Technol.*, vol. 90, pp. 1–26, Oct. 2017.

[36] R. D. Lemos *et al.*, "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in *Software Engineering for Self-Adaptive Systems III. Assurances*. Cham, Switzerland: Springer, 2017, pp. 3–30.

[37] E. Yuan, N. Esfahani, and S. Malek, "A systematic survey of self-protecting software systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 8, no. 4, p. 17, 2014.

[38] D. Weyns, M. U. Iftikhar, S. Malek, and J. Andersson, "Claims and supporting evidence for self-adaptive systems: A literature study," in *Proc. 7th Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst. (SEAMS)*, Jun. 2012, pp. 89–98.

[39] D. Weyns, M. U. Iftikhar, D. G. de la Iglesia, and T. Ahmad, "A survey of formal methods in self-adaptive systems," in *Proc. 5th Int. C* Conf. Comput. Sci. Softw. Eng. (CSE)*, 2012, pp. 67–79.

[40] H. V. D. Parunak and S. A. Brueckner, "Software engineering for self-organizing systems," *Knowl. Eng. Rev.*, vol. 30, no. 4, pp. 419–434, 2015.

[41] N. M. Villegas, H. A. Müller, G. Tamura, L. Duchien, and R. Casallas, "A framework for evaluating quality-driven self-adaptive software systems," in *Proc. 6th Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst. (SEAMS)*, 2011, pp. 80–89.

[42] B. A. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Dept. Comput. Sci., Univ. Durham, Durham, U.K., EBSE Tech. Rep. EBSE-2007-01, 2007.

[43] J. C. de Almeida Biolchini, P. G. Mian, A. C. C. Natali, T. U. Conte, and G. H. Travassos, "Scientific research ontology to support systematic review in software engineering," *Adv. Eng. Inform.*, vol. 21, no. 2, pp. 133–151, Apr. 2007.

[44] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, Jan. 2009.

[45] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," *Ease*, vol. 8, pp. 68–77, Jun. 2008.

[46] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015.

[47] *Zotero*. Accessed: Sep. 20, 2020. [Online]. Available: https://www.zotero.org/

[48] *Systematic Literature Review Paper Set*. Accessed: Sep. 20, 2020. [Online]. Available: https://tinyurl.com/ieeeaccess2020

[49] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka, "On patterns for decentralized control in self-adaptive systems," in *Software Engineering for Self-Adaptive Systems II*. Berlin, Germany: Springer, 2013, pp. 76–107.

[50] C. Urquhart, *Grounded Theory for Qualitative Research*. Newbury Park, CA, USA: Sage, 2012.

[51] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 26, no. 1, pp. 41–50, 2003.

[52] P. Jamshidi, A. Sharifloo, C. Pahl, H. Arabnejad, A. Metzger, and G. Estrada, "Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures," in *Proc. 12th Int. ACM SIGSOFT Conf. Qual. Softw. Archit. (QoSA)*, Apr. 2016, pp. 70–79.

[53] S. Latré and F. D. Turck, "Joint in-network video rate adaptation and measurement-based admission control: Algorithm design and evaluation," *J. Netw. Syst. Manage.*, vol. 21, no. 4, pp. 588–622, 2013.

[54] W. Chainbi, H. Mezni, and K. Ghedira, "PECoDiM: An agent based framework for autonomic Web services," in *Proc. 6th World Congr. Services*, Jul. 2010, pp. 543–550.

[55] Y. Li, Q. Li, L. Wang, W. Cheng, and T. Wu, "ADAPT: An agent-based development toolkit and operation platform for self-adaptive systems," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Nov. 2017, pp. 53–58.

[56] M. S. Aslanpour, S. E. Dashti, M. Ghobaei-Arani, and A. A. Rahmanian, "Resource provisioning for cloud applications: A 3-D, provident and flexible approach," *J. Supercomput.*, vol. 74, no. 12, pp. 6470–6501, Dec. 2018.

[57] F. E. Coutinho, A. L. P. Rego, G. D. Gomes, and J. N. D. Souza, "An architecture for providing elasticity based on autonomic computing concepts," in *Proc. 31st Annu. ACM Symp. Appl. Comput. (SAC)*, New York, NY, USA, 2016, pp. 412–419.

[58] J. Sahni and D. P. Vidyarthi, "Heterogeneity-aware adaptive auto-scaling heuristic for improved QoS and resource usage in cloud environments," *Computing*, vol. 99, no. 4, pp. 351–381, Apr. 2017.

[59] R. Aschoff and A. Zisman, "QoS-driven proactive adaptation of service composition," in *Service-Oriented Computing*. Berlin, Germany: Springer, 2011.

[60] T. A. Reza and M. Barbeau, "QoS aware adaptive security scheme for video streaming in MANETs," in *Foundations and Practice of Security*. Berlin, Germany: Springer, 2013.

[61] A. Bernadas, R. Alfano, A. Manzalini, J. Sole-Pareta, and S. Spadaro, "Demonstrating communication services based on autonomic self-organization," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst.*, Mar. 2008, pp. 101–107.

[62] G. A. Moreno, J. Cámara, D. Garlan, and M. Klein, "Uncertainty reduction in self-adaptive systems," in *Proc. 13th Int. Conf. Softw. Eng. Adapt. Self-Manag. Syst.*, May 2018, pp. 51–57.

[63] P. Antoniou and A. Pitsillides, "Congestion control in autonomous decentralized networks based on the Lotka-Volterra competition model," in *Artificial Neural Networks*. Berlin, Germany: Springer, 2009.

[64] W. Berrayana, G. Pujolle, and H. Youssef, "XLEngine: A cross-layer autonomic architecture with network wide knowledge for QoS support in wireless networks," in *Proc. Int. Conf. Wireless Commun. Mobile Comput., Connecting World Wirelessly (IWCMC)*, New York, NY, USA, 2009, pp. 170–175.

[65] R. Ranjan and L. Zhao, "Peer-to-peer service provisioning in cloud computing environments," *J. Supercomput.*, vol. 65, no. 1, pp. 154–184, Jul. 2013.

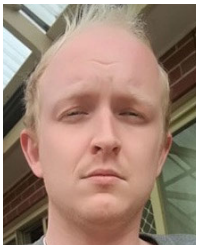
[66] H. Müller and N. Villegas, "Runtime evolution of highly dynamic software," in *Evolving Software Systems*. Berlin, Germany: Springer, 2014, pp. 229–264.

[67] D. B. Abeywickrama and E. Ovaska, "A survey of autonomic computing methods in digital service ecosystems," *Service Oriented Comput. Appl.*, vol. 11, no. 1, pp. 1–31, 2017.

[68] W. Qian, X. Peng, B. Chen, J. Mylopoulos, H. Wang, and W. Zhao, "Rationalism with a dose of empiricism: Combining goal reasoning and case-based reasoning for self-adaptive software systems," *Requirements Eng.*, vol. 20, no. 3, pp. 233–252, 2015.

CLAUDIA SZABO (Member, IEEE) is currently an Associate Professor with the School of Computer Science and the Director of the Complex Systems Research Group, The University of Adelaide. Her research interests include modeling and analysis of complex systems and on practical applications of complexity theory and emergent behavior identification approaches to various application domains.

BRENDAN SIMS received the bachelor's degree (Hons.) in mechatronic engineering from The University of Adelaide, in 2008. He has been employed at the DST Group since 2009, and in that time, he has worked as a part of the Land Division (formerly the Land Operations Division). His research interests include agent-based distributed decision making and the evaluation of distributed self-managing software systems.



THOMAS MCATEE is currently pursuing the master's degree in computer science with the School of Computer Science, The University of Adelaide. His research interests include in the area of artificial life and complex and adaptive systems.

RILEY LODGE received the bachelor's degree (Hons.) in mechatronic engineering and the bachelor's degree in mathematical and computer sciences from The University of Adelaide, in 2017.

He is currently working as a Researcher with the Defence Science and Technology Group, Edinburgh, SA, Australia. His research interests include machine learning, distributed control, and multi-agent systems.

ROBERT HUNJET received the Ph.D. degree from The University of Adelaide, in 2012, for his thesis on the use of autonomy to enable adaptive network topologies. He has been working with the Australia's Defence Science and Technology Group, since 2001. He is currently the Group Leader of their Advanced Vehicle Systems Science and Technology Capability, and is also an Adjunct Associate Professor with the UNSW Canberra's School of Engineering and Information Technology. He serves as the Co-Lead for the Networked Land Autonomy theme within the Defence Collaborative Research Centre on Trusted Autonomous Systems. He also leads a Research Team which seeks to enable autonomy through distributed control of sensors and effectors across federated land vehicles. His research interests include wireless network performance, swarming and emergence, autonomous decision making, network survivability, and distributed control.