Preliminary Results of a Survey on the Use of Self-Adaptation in Industry

Danny Weyns[1,3], Ilias Gerostathopoulos[2], Nadeem Abbas[3], Jesper Andersson[3]
Stefan Biffl[4], Premek Brada[5], Tomas Bures[6], Amleto Di Salle[7]
Patricia Lago[2], Angelika Musil[4,1], Juergen Musil[4] and Patrizio Pelliccione[8]
[1]Katholieke Universiteit Leuven; [2]Vrije Universiteit Amsterdam; [3]Linnaeus University; [4]CDL-SQI, TU Wien; [5]University of West Bohemia, Pilsen; [6]Charles University, Prague; [7]University of L'Aquila; [8]Gran Sasso Science Institute, L'Aquila
[1]Belgium, [2]The Netherlands, [3]Sweden, [4]Austria, [5]Czech Republic, [7,8]Italy
Contact:danny.weyns@kuleuven.be,i.g.gerostathopoulos@vu.nl

ABSTRACT

Self-adaptation equips a software system with a feedback loop that automates tasks that otherwise need to be performed by operators. Such feedback loops have found their way to a variety of practical applications, one typical example is an elastic cloud. Yet, the state of the practice in self-adaptation is currently not clear. To get insights into the use of self-adaptation in practice, we are running a large-scale survey with industry. This paper reports preliminary results based on survey data that we obtained from 113 practitioners spread over 16 countries, 62 of them work with concrete self-adaptive systems. We highlight the main insights obtained so far: motivations for self-adaptation, concrete use cases, and difficulties encountered when applying self-adaptation in practice. We conclude the paper with outlining our plans for the remainder of the study.

CCS CONCEPTS

• **Software and its engineering → Software system structures**; **Designing software**; **Maintaining software**.

KEYWORDS

Self-adaptation, feedback loops, motivations, industrial use cases, difficulties applying self-adaptation, survey

ACM Reference Format:
Danny Weyns[1,3], Ilias Gerostathopoulos[2], Nadeem Abbas[3], Jesper Andersson[3], Stefan Biffl[4], Premek Brada[5], Tomas Bures[6], Amleto Di Salle[7], Patricia Lago[2], Angelika Musil[4,1], Juergen Musil[4] and Patrizio Pelliccione[8]. 2022. Preliminary Results of a Survey on the Use of Self-Adaptation in Industry. In *17th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '22), May 18–23, 2022, PITTSBURGH, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3524844.3528077

1 INTRODUCTION

Software-intensive systems are systems where software plays a vital role in the construction and operation of these systems [18]. Software-intensive systems form the backbone of our factories, traffic, healthcare, telecommunication, finances, and so forth. The trustworthiness and sustainability of these systems is therefore vital for our society. Yet, achieving this is challenging due to complexity of these systems. Our focus here is on the challenge of these systems when dealing with uncertain conditions they face during operation.

A classic approach to deal with uncertainty is equipping a software-intensive system with a feedback loop that automates tasks that are otherwise performed by operators. The purpose of a feedback loop can be very diverse, ranging from minimising the cost of operation under workloads that are hard to predict, ensuring a required level of performance under changing network conditions, dealing with errors caused by external services, or defending the system against malicious attacks and the problems they may cause. Applying feedback loops to software-intensive systems has been subject of active research in academics. Back in 1998, Oreizy et al. [26] introduced the notion of *self-adaptation* that comprises two interacting processes: one that detects and responds to changes, and the other that deals with evolution of the system over time. Garlan et al. [13] pointed out the crucial role of first-class architectural models that enable a system to reason about system-wide change to adapt and satisfy its goals. Kramer and Magee [21] highlighted the role of software architecture in defining self-adaptive systems, distinguishing adaptation management from goal management. The last decade, the research community developed a vast body of knowledge and know-how on engineering self-adaptive systems, e.g., in research road maps [2, 7, 9, 10], books [22, 31] and literature studies, e.g., [1, 14, 16, 24, 25, 32, 36]. Yet, only a few joint efforts in collaboration with industry are reported, e.g., [4–6, 8, 19, 34, 35].

In parallel, the principles of feedback control were studied and applied by industry. Around 2000, IBM launched its legendary initiative on autonomic computing [20]. The central idea of autonomic computing was to enable computing systems to manage themselves based on high-level goals, similar to the human body. Four classic goals are self-optimisation, self-healing, self-protection, and self-configuration. Autonomic computing delegates the complex tasks that traditionally need to be performed by operators to the machine. Over the years, solutions based on feedback loops have found their way to practical applications, for instance, in the domains of elastic clouds and automated management of server parks, see e.g., [3, 28].

So on the one hand, academics have established a vast body of knowledge on engineering self-adaptive systems. On the other hand, the principles of feedback loops are applied by industry. While

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SEAMS '22, May 18–23, 2022, PITTSBURGH, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9305-8/22/05…$15.00
https://doi.org/10.1145/3524844.3528077

the output of academic research is documented, the current use of self-adaptation in industry and state of the practice is not clear.

To get insight into the use of self-adaptation in industry, we are running a large-scale survey with active practitioners. This survey aims at shining light on what motivates practitioners to apply self-adaptation, what kind of problems they solve using self-adaptation, how they solve these problems, whether they have any established practices, what risks and challenges they face in adopting self-adaptation, and what opportunities industry sees in this area. Such insights will help the research community to align their efforts with industrial needs, and they help practitioners by clarifying benefits as well as issues with applying self-adaptation and pointing out opportunities for tackling open challenges. To the best of our knowledge, no systematic study has been done that investigates and answers questions like those specified above.

In this paper, we report preliminary results of the survey based on responses of 62 practitioners from 16 countries that work with concrete self-adaptive systems. We highlight the main insights obtained so far: motivations for self-adaptation, concrete use cases, and difficulties encountered when applying self-adaptation in practice. We target the following three research questions:

RQ1: What motivates practitioners to apply self-adaptation?
RQ2: What are concrete use cases of self-adaptation in practice?
RQ3: What difficulties do practitioners face when applying self-adaptation?

Investigating these questions will give us initial insights in whether research and practice in self-adaptation are aligned on the studied topics and whether there are differences in emphasis.

The remainder of this paper is structured as follows. In Section 2, we summarise the study design and list the survey questions. Section 3 presents the results, grouped per research question. We discuss threats to validity in Section 4. Finally, we draw conclusions and outline our plans for the rest of the study in Section 5.

2 STUDY DESIGN

We present the target population and sampling, the survey instrument, the survey questions, and data analysis. For a full replication package with all study material, the raw data, and the analysis results, we refer to the study website [33].

Population and Sampling. Our target population are practitioners actively involved in the engineering of industrial software-intensive systems in any domain. This includes architects, designers, developers, testers, maintainers, operators, among others who have technical expertise in the engineering of software systems.

Concretely, we contacted 203 practitioners from a wide variety of companies via the networks of the researchers involved in this study to complete the survey. We used two criteria to invite people: (1) a good representation of domains of the current landscape of software-intensive systems, and (2) participants have the required expertise to answer the questions. The invited practitioners were spread over in total 16 countries.[1] The invitations were sent by personalised emails on November 30, 2020. We sent reminders according to a predefined schedule after one, two, and six weeks.

[1]Czech Republic 40, Sweden 35, Austria 29, Belgium 28, Germany 18, The Netherlands 16, Spain 9, Denmark 7, UK and USA each 5, France 3, Switzerland 2, and Greece, Poland, Norway, and Japan each 1

Survey Instrument. A survey collects data based on a set of predefined questions [15]. We used closed questions that have a predefined set of possible answers that participants can pick from, and open questions that provide space that respondents can use to explain an answer in detail. While closed questions allow getting a clear view on a particular topic using basic statistics, open questions allow getting detailed in-depth insights using qualitative analysis.

We used a self-administered anonymous online questionnaire (Survey & Report hosted by Linnaeus University), allowing to involve a large set of respondents with relatively low cost. We created an initial list of survey questions that were directly derived from the research questions of the study. We anticipated that completing the questionnaire would take approximately half an hour.

We validated the questionnaire in a pilot with eight randomly selected participants from the target population. For this pilot, we added additional meta-questions to the questionnaire about clarity of terminology, relevance and scope of the questions, and the time required to complete the survey. For both clarity of terminology and clarity of the questions we obtained an average score of 4.38 on a scale from 1 (Not clear at all) to 5 (Very clear). None of the participants indicated that questions should be removed or modified. The average reported time to complete the survey was 24 minutes. Based on the feedback, we made a few adjustments in the introductory part of the questionnaire and moved one of the questions up in the list. The finalised questionnaire was then distributed to the participants as explained above.

Survey Questions. We summarise here the questions that are relevant to the preliminary results reported in this paper. For the complete questionnaire we refer to the study website [33].

The first set of questions probed whether the participant has worked with self-adaptive systems (Q0.1) and solicited demographic information about the domain (Q0.2), the size of the company (Q0.3), the participant's role (Q0.4) and expertise (Q0.5). The second set provides questions related to RQ1 collecting data about the types of problems for which the respondents apply self-adaptation (Q1.1), and the benefits they obtained from applying self-adaptation (Q1.2). The third set had a single question related to RQ2 on use cases for self-adaptation. The question asked the respondents to describe a concrete self-adaptive system they worked with (Q2.1). Finally, the fourth set addressed RQ3 on difficulties and challenges with applying self-adaptation. The first question investigates the frequency of difficulties (Q3.1); the second question elicited concrete examples of difficulties experienced by the participant (Q3.2). The concrete questions are further explained in the results section.

Data Analysis. To analyse closed questions, we used quantitative data analysis. We report percentages relative to the respective number of responses, and frequency analyses. To analyse textual answers, we used coding [30]. We developed codes and inferred categories from the data by labelling small coherent fragments of the comments [29]. We did not use a pre-defined coding schema, but interpreted comments in the context of the respective questions. Coding was first done individually and then consolidated in a sub-team. Another researcher crosschecked the consolidated coding and where necessary, the coding was adjusted in consensus. When reporting codes, Section 3, we provide a few examples using verbatim excerpts, including spelling and punctuation mistakes.

3 RESULTS

We present now the results of the data analysis. We start with demographics and then we zoom on the results per research question.

3.1 Experience and Demographics

We received 113 valid responses (i.e., a response rate of 56%). Based on the answer to the first question (Q0.1), we split the answers to the other questions of the demographics in two groups: those provided by all respondents and those provided by respondents that worked with concrete self-adaptive systems.

3.1.1 Experience with self-adaptation (Q0.1). Sixty two of the 113 participants (56%) have worked with self-adaptive systems. This indicates that self-adaptation is frequently applied in practice.

3.1.2 Software systems built by organisations (Q0.2). The most frequent type of systems participants build in their company is *Web/mobile systems* with 22 occurrences (19.5%), 11 of them apply self-adaptation. Second comes *embedded*, *cyber-physical*, *Internet of Things systems* (in particular *robotic* and *manufacturing*) with 16 occurrences (14.2%), 10 of them applying self-adaptation. Both *transportation* and *networks* have 8 occurrences with 5 applying self-adaptation. The remaining systems include *data management*, *e-commerce*, and *retail*. The results show that the participants work with a variety of systems and apply self-adaptation. This underpins the representatives of the data collected during the survey.

3.1.3 Software engineers working at companies (Q0.3). Fifty six of the 113 participants (49.5%) work at a company with +100 software engineers; 30 of them apply self-adaptation. Thirteen participants (11.5%) work at a company with less than 10 software engineers; seven of them apply self-adaptation. The remaining 44 participants (39%) work in a company with 10 to 100 software engineers; 25 of them apply self-adaptation. The application of self-adaptation is similar for all categories. These numbers show that companies with a wide range of employed software engineers are represented.

3.1.4 Roles of participants in their organisation (Q0.4). Around three in four participants indicated they have a single role in their organisation. The other participants indicated that they have two or more roles. The most frequent roles are project manager/coordinator with 49 occurrences (43.5%), 31 of them apply self-adaptation, and programmer with 48 occurrences (42.5%), 26 apply self-adaptation. Other roles are designer/architect (29, 13 applying self-adaptation), tester and maintainer (each 10, 5 of them applying self-adaptation), and operator (7, 4 applying self-adaptation). A variety of software engineering roles are represented in our sample and those that work with self-adaptive systems are more or less equally distributed.

3.1.5 Experience of participants (Q0.5). Seventy six of the 113 participants (67.3%) have more than nine years of experience as software engineer; 45 of them apply self-adaptation. Fifteen participants (13.2%) have between one and three years of experience; five of them apply self-adaptation. Of the remaining 22 participants (19.5%) with four to eight years of experience, 12 apply self-adaptation. These numbers show that most participants involved in the survey are mature software engineers and for all categories the representation of those that apply self-adaptation is similar.

3.2 Motivations for Self-adaptation (RQ1)

We analyse now the data that we collected for answering RQ1. This research question focuses on the types of problems they solve using self-adaptation and what the benefits of self-adaptation could be. Note that the data we used to answer RQ1 and the other research questions comes from the 62 participants that have experience with self-adaptive systems (i.e., participants that answered "Yes" to Q0.1).

3.2.1 For which problems do you apply self-adaptation? (Q1.1). On average, the participants applied self-adaptation for 3.7 problems (from a list with seven options), see Table 1. Overall, practitioners apply self-adaptation to deal with a variety of problems. Optimising performance and automating tasks are the main problems tackled by self-adaptation (selected by 79% of the participants). Other problems selected by at least half of the participants are (re-)configuring systems, and deal with changes in the environment.

3.2.2 What could be benefits of applying self-adaptation? (Q1.3). Fifty-eight respondents (94% of those that worked with self-adaptive systems) provided useful descriptions of benefits of applying self-adaptation. On average, we identified 2.1 benefits per respondent. Table 2 summarises the main findings. For each category (gray-shaded rows) and for each code per category we provide the number of occurrences and we give a few illustrative quotes of respondents. The dominating benefits of applying self-adaptation are improved utility (48 participants, i.e., 83% of those that worked with self-adaptive systems) , in particular for robustness and performance, and savings in costs and resources (31 participants, 53%).

Key insights from RQ1:

(1) Self-adaptation is primarily used in industry to optimise performance, automate tasks, configure/re-configure systems, and deal with changes in the environment.
(2) The main reported benefits of applying self-adaptation are improved utility and savings (in costs and resources), improved human interaction, and handling dynamics.
(3) The problems tackled by industry are similar to those studied by academics. Yet, practitioners do not put an emphasis on uncertainties, which is a key focus in research.
(4) The four classic tasks of self-adaptation studied by researchers (self-healing, self-optimising, self-protecting, and self-configuring) are also relevant to practitioners.

Table 1: Problems to apply self-adaptation (Q1.1). Percentages are fractions of participants that selected the problems

Problem	Quantitative
To optimise performance	49 (79%)
To automate tasks	41 (66%)
To configure/reconfigure a system	39 (63%)
To deal with changes in the environment	35 (56%)
To detect and resolve errors	30 (48%)
To detect and protect a system against threats	25 (40%)
To deal with changes in the business goals	8 (13%)
Others	10 (16%)

Table 2: Analysis of answers - Reported benefits of self-adaptation (Q1.3)

Categories/codes	#	Example quotes
Improved utility	48	
Robustness	14	"fault tolerance, one node dies, a new one is spawned without manual intervention"; "better error handling and prompt disaster recovery"
Performance	12	"Improve performance and quality-of-service"; "increase in the speed of adaptation"
Availability	8	"[...] 99.9999% availability, which is crucial for some customers of these cloud-specific solutions"
Other qualities	14	"for IoT: optimized operations, improved energy usage"; "It is also an important part to guarantee the safety [...] of the overall system."
Savings	31	
Costs	17	"The primary benefit is cost reduction"; "the cheaper bills for running this in an efficient manner [...]
Resources	14	"scales down resources during hours when traffic is low, and scales up during peak hours, without any manual interference."
Improved human interaction	20	
User experience	12	"Keep Telco network in optimal condition so that QoS and user experience is maximized, and churn minimized"; "better user satisfaction because of prompt website responses"
Burden engineers	8	"removes most of the optimization burden from programmers, so they can be more productive"; "Reduce workload on human operators; make (the results of) certain actions [...] repeatable and predictable"
Handle dynamics	13	
Context dynamics	7	"Each machine is unique and its optimal operational parameters change over time due to ware, location, task and seasonal factor."
Load dynamics	6	"Change AGV behavior depending of the workload with the goal to save energy (battery life)."
Other improvements	8	
Various	8	" In case of spikes in incoming events the system is able to adapt [...] avoiding bottlenecks."; "Easier and faster market integration";

3.3 Use Cases of Self-adaptation (RQ2)

3.3.1 Explain a concrete self-adaptive system you worked with (Q2.1). Sixty one participants (98%) described a concrete self-adaptive system they worked with. Table 3 summarises the findings. We identified three categories that characterise self-adaptive systems. Sixty one participants (98%) described the subject of adaptation in their systems. System occurred 16 times, followed by module (i.e., a part of a system), and support system (infrastructure, platform, etc.), each with 12 times mentioned (20%). Fifty two participants provided input to the types of adaptation they apply (81%). Auto-scaling with 20 occurrences and automated reconfiguration with 10 occurrences are the most frequent types of adaptations applied by the participants. Finally, the participants explained in total 32 triggers for adaptation (52%). The main triggers are changes in the (work) load of the systems and events, each with nine occurrences.

Key insights from RQ2:

(1) Self-adaptation is applied at different levels of software systems: from complete systems to parts and support systems.
(2) The dominating types of adaptation applied in industry are auto-scaling, automated re-configurations, and auto-tuning.
(3) Adaptions in industrial systems are primarily triggered by dynamics in (work) load, the occurrence of events, and changes in properties of systems and their environments.
(4) Technologies such as elastic cloud, auto-scaling, and container-orchestration systems such as Kubernetes, are key enablers for the realisation of self-adaptation in practice.

3.4 Difficulties with Self-Adaptation (RQ3)

3.4.1 Frequency of difficulties or challenges encountered when engineering or maintaining self-adaptive systems (Q3.1). Twenty three respondents report that they sometimes face difficulties or challenges with applying self-adaptation (37%). Nineteen frequently or very frequently encounter issues, while 7 rarely or very rarely have difficulties. One respondent reported to have always problems, while 6 reported that they have never problems.

3.4.2 Types of difficulties or challenges encountered when engineering or maintaining self-adaptive systems (Q3.2). Forty five respondents (73%) reported on average 1.53 difficulties or challenges with engineering self-adaptive systems. Table 4 summarises the findings. Most frequently reported issues (22 in total) relate to life cycle issues, in particular tuning/debugging. Design challenges were reported by 20 respondents, particularly reliable/optimal design. Other issues concern people and process issues and runtime challenges.

Key insights from RQ3:

(1) Seventy percent of the participants report that they face at least sometimes difficulties or challenges with self-adaptation.
(2) The difficulties and challenges crosscut the whole life cycle, from design and testing time to runtime and evolution.
(3) Top reported issues are tuning/debugging, reliable/optimal design and design complexity.
(4) Issues with skills and experience of people as well as process management are frequently reported.

Table 3: Analysis of answers – Explain a concrete self-adaptive system you worked with (Q2.1)

Categories and codes	#	Example quotes
Subject of adaptation	61	
System	16	"Autotuning of machine producing electricity."
Module	12	"Monitoring the memory/CPU/disk consumption of our servers and suggesting measures to fix it through human intervention."
Support system	12	"The HotSpot JVM of OpenJDK. It reads a program's Java bytecode, and adaptively tunes the performance of the program at runtime, adapting to runtime profiles."; "The adaptive integration platform (AIP)"
Cluster	7	"auto-scaling of application cluster (computation resources) based on workload submitted by user - done in multiple environments"
Cloud	7	"EBS (cloud auto-scaling). Monitors runtime performance of cloud instances and allows the establishment of automatic rules for scaling in/out the instances, e.g. responding to workload changes"
Application	4	"auto-tuning systems (vehicles with driving support) [...] It could also adapt based on external factors like other vehicles."
Generic	3	"We are currently developing a generic self-adaptation platform that can be applied to numerous cases. "
Type of adaptation	52	
Auto-scaling	20	"Automated horizontal scaling of AWS EC2 instances for medical data processing systems"; "autoscale a cluster based on the resource usage of the nodes of the cluster."
Automated reconfiguration	10	"Our company develops safety critical systems for railway. Systems architecture is often with redundancy - e.g. 2 out of 3 system, where is automatic reconfiguration implemented. Purpose is high safety and availability."
Auto-tuning	8	"A mink feeding robot, that can adjust the food amount according to a set of feeding rules and the food left over from last feeding. "
Monitor/Analysis	6	"We configured AWS alarms to monitor performance of our systems in case we get more than few number of HTTP 400/500 errors"; "Monitoring the memory/CPU/disk consumption of our servers and suggesting measures to fix it through human intervention."
Automated CI	3	"Continuos integration system - Monitors codebase & starts building & testing a new version as soon as it detects code changes Build alignment - Creates a new release whenever a subsystem builds successfully"
Other	5	"Proprietary load balancers that sits in front of bare metal servers."
Trigger of adaptation	32	
Load	9	"Kubernetes, for handling load intensive periods for scaling up, and self recover from crashes."; "Autoscaling of SaaS applications in function of load on AWS and Azure clouds."
Events	9	"We use kubernetes which provides notification callbacks on any event such as host/pod not available, based on these events we auto mark the node was inactive and do not use those nodes for further write or read operations"; "Auto Scaling an EMR cluster in AWS based on incoming event data"
Environment properties	6	"An IoT system running in Kubernetes and used to monitor water leaking for household insurance."
System properties	6	"Auto-scaling functionality of an Azure Service Fabric cluster running a transformation load for processing AGV statistical and playback data."
User actions	2	"[adapt] cache warm up strategy based on user interactions"

4 THREATS TO VALIDITY

The sample of practitioners that participated in this survey were recruited from our networks with industry. Since several of the researchers involved in this study are active in the field of engineering self-adaptive systems, the practitioners of these networks may have been biased and inclined to apply self-adaptation more often. To anticipate this validity threat, we did not particularly focused on practitioners we have worked with in projects, but invited practitioners in various software engineering roles that are active across a wide range of domains.

Further, our sample is not well balanced on a global scale, hence it may not be possible to generalise the results to industry in parts of the world that are only marginally represented in the sample. To anticipate this threat, we are currently running a second round of our survey to compensate for underrepresented areas to guarantee a better geographic coverage.

We used the term self-adaptation to formulate questions about systems (or parts) that are equipped with a feedback loop. Hence, most questions required basic knowledge of the concept of self-adaptation. Respondents may have misinterpreted the concept or some of the questions. To anticipate this threat, we have carefully introduced the notion of self-adaptation in the survey, leveraging on the input we received from validating the questionnaire.

Since our sample for the survey was drawn from practitioners of our networks with industry, they may not be representative for the population of software engineers in general and those that apply self-adaptation in particular. However, the results of the demographics of our sample show that the respondents were active

Table 4: Analysis of answers - Difficulties and challenges encountered with engineering self-adaptive systems (Q3.2)

Categories and codes	#	Example quotes
Lifecycle issues	22	
Tuning/debugging	10	"Debugging the root cause of a scaling failure might be time-consuming: also, in some cases the problem might be outside of your control (e.g. temporary lack of EC2 Spot capacity in AWS)"
Limitations of tools/methods	7	"The metrics available are not always fully transparent and built with auto-scaling in mind"; "IAM permissions are hard to deal with when configuring these self-adaptive systems. Usually, the permission to scale or to notify is not properly configured."
System/environment evolution	5	"If the functionality is not designed in from the beginning then it is a huge amount of work to implement later."; "System architecture over lifetime (nee features to be added...)"
Design challenges	20	
Reliable/optimal design	10	"If things starts to fail - then we have major delivery problems very fast. This area is critical."; "the main challenge is to design adaptation function with respect to computation context - e.g., selecting right trade-off between shutting down instance or keeping it running longer time since boot of instance can be time-consuming."
Design complexity	9	"Complexity in defining the adaptation rules. Conditions are not always obvious."; "Self-adaptiveness or resilience have to be taken into consideration at each stage of the software production and operation workflow of a distributed system. This is really a challenge as more often than not these are concepts that are completely obscure to the average programmer/devop mind."
Security	1	"Software updates/dependencies, security"
People and process issues	14	
Skills/experience	8	"Every self-adapt system must be tuned up which is sometimes tricky and needs high skilled engineers."; "The Kubernetes/Openshift cloud and centralized log storage (ElasticSearch - Logstash - Kibana) require experienced administration staff and vast knowledge of many networking concepts ([...], DNS, NAT"
Process and management	5	"We are not yet very experienced with the system, the main challenges were to convince the central IT department this was the way to go, then to design the system, and obviously master the technology"
Documentation	1	"humans don't write things down and don't like to revisit what they do to facilitate writing things down"
Runtime challenges	13	
Runtime uncertainty	6	"Many self-adaptive systems are based on unproven heuristics. Therefore, they usually do not work in many cases."; "It is hard to guess how much can the environment affect the system. Usually, we develop the system on one or a limited set of units. It is hard to extend the parameters to cover whole production."
Data collection/evaluation	5	"Gathering quantitative data samples to evaluate the performance is very complicated."; "sensors gives wrong reading values"
Delayed/missing changes	2	"Autoscaling is often too slow or triggered too late."

practitioners with sufficient expertise in various roles across companies of different sizes.

Regarding the qualitative analysis of answers with free text we used coding. Coding is a creative task that is influenced by the experience (and even opinions) of the coders [12]. To mitigate potential subjectivity, the data analysis was performed by multiple researchers and the codes and concepts that emerged were discussed to reach consensus.

5 CONCLUSIONS AND FUTURE WORK

This paper reports preliminary results of a survey that aims at understanding the state of the practice in the use of self-adaptation. We focused on the motivations of practitioners to apply self-adaptation, concrete use cases, and difficulties practitioners face when applying self-adaptation. The results show that practitioners apply self-adaptation for problems similar to those that are studied in academics. Yet, practitioners put more emphasis on costs and resources (compared to an emphasis on quality properties by researchers) and less on uncertainty (which is a key concern in academics, see e.g., [11, 17, 23, 27]). Self-adaptation is applied at different levels of granularity of the system, and particular emphasis is given to system support (deployment and execution environment, integration platforms, etc.). The main focus is on auto-scaling, automated reconfiguration, and auto-tuning, exploiting key technologies such as elastic cloud and container-orchestration. A majority of practitioners face both technical difficulties and challenges with realising self-adaptation as people and process issues.

The results reported in this paper are part of a large-scale study that is ongoing at the time of writing. We recently launched a second round of the survey with 100 additional practitioners. Besides the topics reported in this paper, we collect also data about *how* practitioners realise self-adaptation, how they ensure *trustworthiness* of their solutions, what *risks* they face and how they mitigate these risks, what future *opportunities* they see for self-adaptation, and how *collaboration* with academics could help them with using self-adaptation in practice. In addition, we are running a series of interviews with practitioners that reported not to apply self-adaptation to better understand the reasons for this.

ACKNOWLEDGMENT

We thank the participants that contributed input to the study, the reviewers of the survey protocol, and Linnaeus University for hosting the survey. The financial support by Christian Doppler Research Association, the BMDW and the National Foundation for Research, Technology and Development is gratefully acknowledged.

REFERENCES

[1] I. Alfonso, K. GarcÃĺs, and H. et al. Castro. 2021. Self-adaptive Architectures in IoT Systems: A Systematic Literature Review. *Journal on Internet Services and Applications* 12, 4 (2021).

[2] U. Aßmann, S. Götz, J.M. Jézéquel, B. Morin, and M. Trapp. 2014. *A Reference Architecture and Roadmap for Models@run.time Systems.* Springer, 1–18. https://doi.org/10.1007/978-3-319-08915-7_1

[3] B. Beyer, C. Jones, N. Murphy, and J. Petoff. 2016. *Site Reliability Engineering, How Google Runs Production Systems.* O'Reilly Media, Inc.

[4] T. Bolender, G. BÃijrvenich, M. Dalibor, B. Rumpe, and A. Wortmann. 2021. Self-Adaptive Manufacturing with Digital Twins. In *Software Engineering for Adaptive and Self-Managing Systems.* IEEE, 156–166. https://doi.org/10.1109/SEAMS51251.2021.00029

[5] J. Camara, P. Correia, R. de Lemos, D. Garlan, P. Gomes, B. Schmerl, and R. Ventura. 2013. Evolving an adaptive industrial software system to use architecture-based self-adaptation. In *8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* https://doi.org/10.1109/SEAMS.2013.6595488

[6] J. Cámara, P. Correia, R. de Lemos, D. Garlan, P. Gomes, B. Schmerl, and R. Ventura. 2016. Incorporating Architecture-Based Self-Adaptation into an Adaptive Industrial Software System. *J. Syst. Softw.* 122, C (dec 2016), 507âĂŞ523. https://doi.org/10.1016/j.jss.2015.09.021

[7] B. Cheng, R. de Lemos, H. Giese, et al. 2009. Software engineering for self-adaptive systems: A research roadmap. *Software Engineering for Self-Adaptive Systems* (2009), 1–26.

[8] C. da Silva, JosÃľ D. Saraiva da Silva, C. Paterson, and R. Calinescu. 2017. Self-Adaptive Role-Based Access Control for Business Processes. In *12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* https://doi.org/10.1109/SEAMS.2017.13

[9] R. de Lemos, D. Garlan, C. Ghezzi, et al. 2017. Software Engineering for Self-Adaptive Systems: Research Challenges in the Provision of Assurances. In *Software Engineering for Self-Adaptive Systems III. Assurances.* Springer, 3–30.

[10] R. De Lemos, H. Giese, H. Müller, et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II.* Springer, 1–32.

[11] N. Esfahani and S. Malek. 2013. Uncertainty in Self-Adaptive Software Systems. In *Software Engineering for Self-Adaptive Systems II.* Springer.

[12] D. MÃĺndez FernÃąndez, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. VetrÃš, T. Conte, M. Christiansson, D. Greer, C. Lassenius, T. MÃďnnistÃű, M. Nayabi, M. Oivo, B. Penzenstadler, and D. Pfahl. 2016. Naming the Pain in Requirements Engineering. *Empirical Software Engineering* 22 (2016), 2298–2338.

[13] D. Garlan, S.W. Cheng, A.C. Huang, et al. 2004. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer* 37, 10 (2004), 46–54.

[14] O. Gheibi, D. Weyns, and F. Quin. 2021. Applying Machine Learning in Self-Adaptive Systems: A Systematic Literature Review. *ACM Transactions on Autonomous and Adaptive Systems* 15, 3 (2021).

[15] D. Gray. 2013. *Doing research in the real world.* SAGE Publications Ltd.

[16] M. Grua, I. Malavolta, and P. Lago. 2019. Self-Adaptation in Mobile Apps: a Systematic Literature Study. In *IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* 51–62. https://doi.org/10.1109/SEAMS.2019.00016

[17] S. Hezavehi, D. Weyns, P. Avgeriou, R. Calinescu, R. Mirandola, and D. Perez-Palacin. 2021. Uncertainty in Self-Adaptive Systems: A Research Community Perspective. *ACM Transactions on Autonomous and Adaptive Systems* 15, 4, Article 10 (2021). https://doi.org/10.1145/3487921

[18] M. Hölzl, A. Rauschmayer, and M. Wirsing. 2008. *Engineering of Software-Intensive Systems: State of the Art and Research Challenges.* Springer, 1–44. https://doi.org/10.1007/978-3-540-89437-7_1

[19] S. Hurtado, S. Sen, and R. Casallas. 2011. Reusing Legacy Software in a Self-Adaptive Middleware Framework. In *Adaptive and Reflective Middleware.* ACM, 29àĂŞ35. https://doi.org/10.1145/2088876.2088881

[20] J. Kephart and D. Chess. 2003. The Vision of Autonomic Computing. *Computer* 36, 1 (2003), 41–50.

[21] J. Kramer and J. Magee. 2007. Self-Managed Systems: an Architectural Challenge. In *Future of Software Engineering (FOSE '07).* 259–268.

[22] P. Lalanda, J. McCann, and A. Diaconescu. 2013. *Autonomic Computing Principles, Design and Implementation.* Springer.

[23] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns. 2017. A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements. In *Managing Trade-Offs in Adaptable Software Architectures.* Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-802855-1.00003-4

[24] M. Tourchi Moghaddam and É. Rutten. 2020. Self-adaptive Middleware Support for IoT and CPS A Systematic Literature Review. In *https://cps4eu.eu/wp-content/uploads/2021/09/Self-adaptiveMiddlewareSupportforIoTandCPS.pdf*.

[25] A. Musil, J. Musil, D. Weyns, T. Bures, H. Muccini, and M. Sharaf. 2017. *Patterns for Self-Adaptation in Cyber-Physical Systems.* Springer, 331–368. https://doi.org/10.1007/978-3-319-56345-9_13

[26] P. Oreizy, M.M. Gorlick, R.N. Taylor, and Others. 1999. An architecture-based approach to self-adaptive software. *Intelligent Systems and their Applications* 14, 3 (1999), 54–62.

[27] A. Ramirez, A. Jensen, and B. Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* https://doi.org/10.1109/SEAMS.2012.6224396

[28] A. Spyker. 9/2020. Disenchantment: Netflix Titus, Its Feisty Team, and Daemons. *InfoQ* (9/2020). https://www.infoq.com/presentations/netflix-titus-2018/

[29] K. Stol, P. Ralph, and B. Fitzgerald. 2016. Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. In *38th International conference on Software Engineering (ICSE).* 120–131.

[30] A. Strauss and J. Corbin. 1990. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques.* SAGE.

[31] D. Weyns. 2021. *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective.* Wiley. https://books.google.be/books?id=zaC9vgEACAAJ ISBN: 978-1-119-57494-1.

[32] D. Weyns and T. Ahmad. 2013. Claims and Evidence for Architecture-based Self Adaptation - A Systematic Literature Review. In *7th European Conference on Software Architecture (ECSA).* 249–265.

[33] D. Weyns, I. Gerostathopoulos, N. Abbas, et al. 2022. Project Website: Survey on the Use of Self-Adaptation in Industry. https://people.cs.kuleuven.be/~danny.weyns/surveys/sas-in-industry/. Last accessed: 2022-03-18.

[34] D. Weyns and U. Iftikhar. 2022. ActivFORMS: A Formally Founded Model-Based Approach to Engineer Self-Adaptive Systems. *ACM Transactions on Software Engineering and Methodology* (2022). (in print).

[35] D. Weyns, U. Iftikhar, D. Hughes, and N. Matthys. 2018. Applying Architecture-Based Adaptation to Automate the Management of Internet-of-Things. In *Software Architecture.* Springer, 49–67.

[36] T. Wong, M. Wagner, and C. Treude. 2021. Self-Adaptive Systems: A Systematic Literature Review Across Categories and Domains. arXiv:2101.00125 [cs.SE]