Towards understanding adaptation latency in self-adaptive systems*

Claas Keller and Zoltán Ádám Mann

University of Duisburg-Essen, Essen, Germany

Abstract. An important feature of service-based and cloud-based systems is their ability to perform self-adaptation. Through self-adaptation, such systems can automatically react to changes and thus ensure the continued satisfaction of their functional and non-functional requirements. Self-adaptation may take non-negligible time (which we term adaptation latency), and during this period the self-adaptive system may exhibit degraded performance or other negative impact. Hence, it is important to understand how long self-adaptations take and what influences the adaptation latency. However, we are not aware of a systematic study of this question in the literature. This paper is a first step in this direction. We present (i) a model of adaptation latency that breaks it down into four components and (ii) a preliminary survey, limited to one conference series and to service-based and cloud-based systems, to analyze information about adaptation latency in the available literature on self-adaptive systems. According to the findings from this preliminary survey, although some components of the adaptation latency are studied in some publications, the whole adaptation delay is seldom considered.

1 Introduction

Modern software systems must operate in highly dynamic environments. To effectively cope with changes of the environment, the concept of self-adaptation has been proposed [31]. A self-adaptive system reacts to changes in the environment by adapting its own structure or behavior at run time, so that the system continues to satisfy its requirements [5]. For example, in a service-based application, if one of the used services becomes unavailable, an alternative service can automatically be involved instead to ensure that the service-based application remains functional [14]. As another example, a cloud-based application can react to an increasing workload by automatically scaling out to use more virtual machines [21].

From the moment the change in the environment happens, it takes some time until the self-adaptive system resolves the issue. We call this time *adaptation latency*. The amount of the adaptation latency can be very different, depending on the type of change of the environment, the type of adaptation used by the

* This paper was published in: Service-Oriented Computing – ICSOC 2019 Workshops, pp. 42-53, Lecture Notes in Computer Science, volume 12019, Springer, 2020

self-adaptive system etc. During the period of the adaptation latency, the self-adaptive system may be in a transient state in which its performance may be degraded and some requirements may be temporarily violated [23]. For example, if a cloud-based application scales out to support an increased number of user requests, it takes some time until this adaptation action takes effect, and in the meantime, the application's response time may be too high (e.g., higher than stipulated in the service level agreement) [24].

The adaptation latency is important for multiple reasons. First, it is a fundamental goal of self-adaptation to reach a new system state in which the requirements are again satisfied as soon as possible, i.e., with minimum adaptation latency, so as to minimize the negative impact of the transient state during the adaptation process [12]. Second, if the new state is reached with a high delay, this increases the likelihood that in the meantime the environment has changed again, so that the ongoing adaptation will not be effective anymore. In other words, the speed of adaptation should be commensurate with the speed of change in the environment [23]. Third, the self-adaptive system may be able to make better adaptation decisions if it is aware of the latency associated with the possible adaptation actions. For example, knowing how long it takes to spin up a new virtual machine, a cloud-based application can start the scale-out in a proactive way, early enough [25, 26].

Despite the importance of adaptation latency for self-adaptation, we are not aware of a systematic study about adaptation latency, the factors influencing adaptation latency, or the implications of adaptation latency. In fact, there is not even consensus about the name and the exact scope of adaptation latency. For example, Tamura et al. call it "settling time" and include the time for making an adaptation decision and executing it [35]. On the other hand, Gambi et al. consider what they call "actuation delay", which includes the time for executing an adaptation and the time it takes for the adaptation to show its effect [13]. Cámara et al. use the terms "adaptation latency"[1] and synonymously "adaptation tactic latency" to refer to the time it takes to execute an adaptation action [8].

Therefore, this paper makes two contributions towards a better understanding of adaptation latency in self-adaptive systems. First, we present a simple model of adaptation latency that breaks it down into four components. This model makes it easier to compare different notions of adaptation latency used by different authors. Second, we present preliminary results of a literature survey, so far limited to one relevant conference series (Software Engineering for Adaptive and Self-Managing Systems, SEAMS[2]) and to the topics of service-based and cloud-based systems. In the survey, we identified the papers that contain specific information about adaptation latency and mapped them on our model of

[1] It should be noted that this is different from the meaning of "adaptation latency" in this paper. The adaptation latency considered by Cámara et al. is only a part of the adaptation latency considered in this paper.

[2] http://self-adaptive.org/seams/

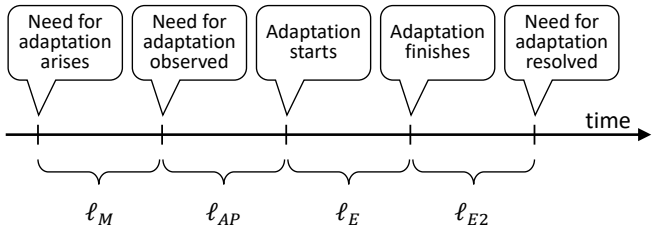

Fig. 1. Model of adaptation latency

adaptation latency. According to the preliminary findings, most of the relevant papers address only some components of the adaptation latency.

The rest of the paper is organized as follows. Section 2 presents our model of adaptation latency and the components of adaptation latency. Section 3 describes the methodology and the results of our preliminary literature survey. Section 4 discusses the findings, and Section 5 concludes the paper.

2 A model of adaptation latency

As already mentioned in Section 1, different authors consider different latencies when evaluating the speed of adaptation. To make a meaningful comparison between different approaches, we provide a simple model of adaptation latency that we believe to be a good basis for capturing different temporal aspects of adaptation.

Our model is related to the well-known MAPE model of self-adaptive systems [19]. According to the MAPE model, a self-adaptive system monitors (M) its environment to detect changes, analyzes (A) the changes to decide if adaptation is necessary, plans (P) adaptations if necessary, and executes (E) the adaptations.

The proposed model for adaptation latency is shown in Fig. 1. As can be seen, there are five key points in time:

1. First, the need for adaptation arises, typically in the form of a change in the environment. For example, the number of users of a cloud-based application starts to increase.
2. After some time ℓ_M, which is related to the monitoring activity, the self-adaptive system recognizes the change. For example, the cloud-based application observes that the queue length of user requests grew over a threshold.
3. This is followed by the analysis and planning activities, taking altogether ℓ_{AP} time, leading to the decision to perform a specific adaptation. For example, the cloud-based application decides to turn on a new virtual machine for the purpose of scale-out.
4. The execution of the adaptation takes ℓ_E time. In our example, this is the time until the new virtual machine is turned on and registered with the load balancer.

5. Finally, it takes further ℓ_{E2} time until the adaptation shows its effect. In our example, this is the time until the queue length is normalized again as a result of the increased processing power.

There are two main differences between our model and the MAPE model. First, we do not differentiate between the analysis and planning activities. The differentiation between analysis and planning is a purely internal concern of the self-adaptive system; moreover, there are a number of approaches to self-adaptation that do not have separate analysis and planning activities [31, 3]. The other difference is that our model also includes ℓ_{E2} which does not have an equivalent in the MAPE model, but it is important for understanding the overall temporal behavior of self-adaptive systems.

We define the *adaptation latency* as $L = \ell_M + \ell_{AP} + \ell_E + \ell_{E2}$. The individual delays $\ell_M, \ell_{AP}, \ell_E, \ell_{E2}$ are called the *components* of the adaptation latency.

3 Preliminary literature study

Using the model introduced above, we performed a (limited) literature survey. We first describe the methodology of this survey in Section 3.1, followed by the main results in Section 3.2. Finally, in Section 3.3, we mention some further papers excluded during the literature survey which could nevertheless provide interesting impetuses to further research.

3.1 Methodology

In the long run, we plan to perform a comprehensive literature review on the topic of adaptation latency in self-adaptive systems. As a first step, we systematically reviewed all papers published in the SEAMS (Software Engineering for Adaptive and Self-Managing Systems) conference series from 2009 to 2019. We filtered the papers according to the following criteria:

- We only included papers that contain some information about adaptation latency. This also includes papers in which information about adaptation latency is only present in diagrams about experiments.
- We only included papers that are related to the field of service-based or cloud-based systems.

We performed this limited literature study manually, i.e., looking at each paper published in SEAMS in the given period. Although it is more common to perform a literature survey using a set of search strings applied to a set of databases, we opted for the manual approach focused on one conference series because of the difficulties associated with finding the appropriate search terms. This way, we do not run the risk of missing whole classes of relevant papers because of a poorly chosen search string. In fact, the papers found with the help of the manual search can serve in the future as baseline for identifying appropriate search terms, which can then be applied in a database search in the future. For now, we manually went through all 225 papers published in SEAMS between 2009 and 2019 and applied the above filter criteria.

Table 1. Relevant publications

Paper	ℓ_M	ℓ_{AP}	ℓ_E	ℓ_{E2}	Time	Notion
[9]	✓	✓	✓	✓	≈ 300 s	none explicitly mentioned
[34]		✓			118–389 ms	time to find a reconfiguration
[20]		✓			≥ 9 min	planning time
[32]		✓			≥ 1513 ms	time for generating a workflow
[1]		✓	✓		0.2–1.02 ms	performance of adaptation
[35]		✓	✓		1.85–2.34 ms	performance, settling time
[36]		✓	✓		12 s – 15 min	redeployment time
[29]			✓	✓	125–625 μs	transition time
[13]			✓	✓	115–420 s	actuation delay
[7]				✓	∅ 19.74 s	time to recovery
[38]		✓	✓			execution time
[8]			✓			adaptation (tactic) latency
[28]		*	✓			tactic latency

3.2 Results

The papers identified as relevant according to the above criteria are summarized in Table 1. For each of the relevant papers, the components of the adaptation latency covered by the paper are indicated, as well as the name that the authors of the paper used to call the considered part of the adaptation latency. In the first half of the table, also the specific duration measured in the paper (corresponding to the sum of the marked components of the adaptation latency) is given; the papers in the second half of the table did not contain such values.

As can be seen from Table 1, only [9] considers the whole adaptation latency. However, that paper does not make any specific statement about the adaptation latency. The quoted information about adaptation latency can only be read off from a diagram of an experiment within the paper. The experiment shows how a self-adaptive web application can manage the slashdot effect by appropriate adaptations, so that the response time of the web application goes after an initial increase back to its normal values.

Several papers focus on the ℓ_{AP} component of the adaptation latency. This may be attributed to the fact that analysis and planning exhibit the most interesting challenges algorithmically, leading to high research attention. In particular, [34] proposes a sophisticated planning algorithm using Pseudo-Boolean constraints; the performance of the planning algorithm was also in the focus of the evaluation using the Heroku platform-as-a-service environment. Similarly, [20] proposes a planner using genetic programming, and makes statements about the time it takes to create a plan by their planner and a baseline planner based on an existing model checker. [32] considers the problem of generating a workflow for dynamically changing the configuration of a self-adaptive system by integrating and testing new components at run time. This problem is a part of the general planning activity of a self-adaptive system (e.g., it does not include

the choice of components to add), but the authors formulate it as a planning problem on its own and evaluate the time needed for this planning.

Some papers take, beside ℓ_{AP}, also ℓ_E into account. [1] considers the proactive adaptation of service compositions. In evaluating their approach, they measure the time of determining the need for adaptation (analysis), the time to determine the necessary changes to the service composition (planning) and the time to actually change it (execution). An interesting finding is that problems occurring early in the workflow of composed services lead to higher adaptation latency than problems occurring later. This is because the space of possible solutions is larger if the problem occurs early. [35] considers the adaptation of the monitoring infrastructure for an adaptive web application. In their experimental evaluation the authors measure the time from detecting a change until the adaptation is finished. [36] addresses the problem of dynamically redeploying service-oriented systems, also measuring latency from detecting a change until the adaptation is finished.

[29] presents an approach for the dynamic change between pre-compiled variants of a software at run time. In the experimental evaluation, the time of transitioning from one variant to another is measured, corresponding to $\ell_E + \ell_{E2}$ in our model. Similarly, also [13] considers the time $\ell_E + \ell_{E2}$ and calls it actuation delay. In contrast to most other found papers which only make statements about the adaptation latency in the context of their empirical evaluation, [13] focuses explicitly on the problem of estimating the actuation delay.

We also found one paper that focuses specifically on the ℓ_{E2} component of the adaptation latency. [7] investigates to what extent and how quickly self-adaptive systems can recover after changes. The authors define the metric Mean Time To Recover and measure it for an adaptive web service.

The second part of Table 1 lists papers that do not contain specific duration information, but still explicitly address (some components of) the adaptation latency. [38] presents a simulator of a self-adaptive system in which different adaptation engines can be evaluated, and the simulator measures the execution time of the adaptation engine. [8] shows that taking into account the latency associated with the execution of different adaptation tactics leads to better adaptation decisions. [28] also takes into account the latency of adaptation tactics (ℓ_E); in addition, it aims to speed up decision-making (ℓ_{AP}, denoted by a * in the table to make clear that this duration is not part of the tactic latency).

In addition to the papers in Table 1, also [16] should be mentioned. This paper is not about the latency of specific adaptations, but about overall metrics to quantify the performance of cloud elasticity solutions, thus aggregating the effect of a series of adaptations.

3.3 Adaptation latency in other papers

Although our present study was limited to papers about service-based and cloud-based systems, we also found papers published in SEAMS that are unrelated to these domains but contained interesting information about adaptation latency.

For example, [10] addresses the problem of reverting short-term remediation actions. The suggested approach is evaluated using an example from the smart homes domain, which is not relevant to the domains covered here. However, the evaluation contains information about all four components of the adaptation latency. [2] proposes an adaptive approach for the mitigation of Denial-of-Service attacks; the experimental results also showcase the full adaptation latency with all its four components. [39] addresses adaptations in networked embedded systems under real-time constraints, where the adaptation latency must remain within given bounds even in the worst case. [6] improves the self-adaptation behavior of an industrial data acquisition and control system using architecture-based self-adaptation and shows that the re-engineered system can recover from disturbances faster.

Other works contain information about specific components of the adaptation latency. [30] investigates the application of genetic algorithms to find optimal adaptations for mobile applications and is specifically concerned with the time taken by the algorithm (ℓ_{AP}). [33] devises an approach for the self-adaptation of access control policies, and measures the execution time of the proposed approach (ℓ_{AP}). [4] proposes an approach which can adapt the requirements if the available resources are not sufficient to satisfy the requirements, applies this approach in meal planning to reduce food waste, and measures the processing time ($\ell_{AP} + \ell_E$). [27] presents an exemplar for self-adaptation approaches for cyber-physical systems, and emphasizes the importance of timing in this domain. In particular, the exemplar explicitly supports tactic latencies (ℓ_E). Also in the domain of cyber-physical systems, [17] investigates how offline machine learning can reduce the time needed for online planning (ℓ_{AP}).

[15] presents a systematic literature study about self-adaptation in mobile apps. Regarding timing, the result of the study was that all found approaches were best-effort, i.e., without any guarantees for the adaptation latency.

[18] is domain-independent and defines a large set of metrics for the evaluation of self-adaptive systems. The metric that comes closest to our adaptation latency is "Time for Adaptation" which is defined as the "time to return to a nominal behavior after a perturbation". Similarly, [37] defines a set of properties and metrics for the evaluation of self-adaptive systems. That paper uses "settling time", defined as "the time required for the adaptive system to achieve the desired state", but it is mentioned that several other terms are used in the literature (recovery time, reaction time, healing time).

[11] proposes a control-theoretic approach to self-adaptation, which allows to derive an upper bound on the settling time. The resulting estimate is actually the number of iterations of the control loop, after which the investigated system property will be within given proximity of the goal.

4 Discussion

Regarding the model of adaptation latency proposed in Section 2, some details may require further elaboration. For example, it is not always clear when exactly

the "need for adaptation arises" (which is the point in time from which ℓ_M is measured). Like any model, also our model of adaptation latency abstracts from some details of reality and thus may leave some room for interpretation when being applied to a specific scenario. We found this level of uncertainty acceptable when analyzing the literature, and we could determine in each case which components of the adaptation latency are involved. Also the question of how appropriate the model is can be answered in this context: we found the model very useful for structuring the literature relating to adaptation latency. For other purposes, it may or may not be appropriate, depending on the required level of detail.

Regarding the results of the survey presented in Section 3, several observations can be made:

- In most of the found papers, information about adaptation latency was only presented in the context of an experimental evaluation. In most cases, the proposed approaches were not adaptation-latency-aware themselves, i.e., they did not perform any reasoning on latency-related information. Such reasoning, however, could be very useful [25]. Also those papers that did reasoning about adaptation latency, were only concerned with the latency of the execution of adaptation tactics. Hence we expect to see more research on adaptation-latency-aware self-adaptation approaches in the future.
- Information about adaptation latency was limited in most papers to ℓ_{AP} and/or ℓ_E, which are the parts of the adaptation latency that are mostly internal to the self-adaptive system. The other parts of the adaptation latency, which are more strongly related to the environment (ℓ_M and ℓ_{E2}) are considered less frequently. On the one hand, this is understandable since we can better control the system-related components (ℓ_{AP} and ℓ_E). However, the effectiveness of self-adaptation is ultimately determined by the adaptation latency as a whole, in which the environment-related components (ℓ_M and ℓ_{E2}) can be just as important as the system-related components. Hence, more research may be needed on the environment-related components of the adaptation latency.
- The specific timing information collected in the penultimate column of Table 1 has huge variance. Obviously, timing measurements stemming from different technical environments cannot be directly compared to each other, but there could be some trends at least concerning the orders of magnitudes (especially since the considered papers are all from similar domains). However, not even such trends are observable: for each component of the adaptation latency for which we have multiple measurements, these vary by several orders of magnitude.
- As shown in the last column of Table 1, there is no generally accepted name for adaptation latency. Rather, the authors usually resort to different, longer expressions to describe adaptation latency. Naming is also challenging because of the ambiguity with the base functionality of the self-adaptive system (e.g., latency for processing web page requests versus latency of an adapta-

tion). Unfortunately, the lack of a generally accepted term for adaptation latency makes it difficult to search for relevant work using keyword search.

Of course, these observations are based on the limited literature survey presented in this paper, and should hence be seen as preliminary. It remains an important task for future research to check whether the observations are supported also by a comprehensive survey of the relevant literature.

5 Conclusions and future work

This paper is a first step towards a better understanding of adaptation latency in self-adaptive systems. In particular, we have presented a model of adaptation latency that identifies its main components. Furthermore, we conducted a literature survey on information about adaptation latency, for the time being restricted to the SEAMS conference series and to service-based and cloud-based systems, and used our model of adaptation latency to categorize the found papers. The results of the literature survey show that there is some awareness of the importance of adaptation latency in the research community, but this awareness is limited. One of the identified limitations is that most of the relevant papers only consider some components of the adaptation latency and ignore other components that could also be important. Another limitation is that most of the relevant papers deal with adaptation latency only in their experimental evaluation, which means that most of the presented approaches are not adaptation-latency-aware. On the other hand, adaptation-latency-aware approaches can be very powerful, even if limited to awareness of a component of the adaptation latency, like the latency of adaptation execution [25] or the latency of planning [22]. Hence we expect to see more research in this direction in the future.

The next step in our research is to extend the literature survey to other publication venues and to other domains of self-adaptive systems. This way, we expect to collect a larger body of related papers, allowing us to do a more comprehensive qualitative and quantitative analysis, also comparing different research communities in terms of their relation to adaptation latency. We are particularly interested in (i) insights into the aspects influencing adaptation latency, (ii) experience about the consequences of adaptation latency, and (iii) approaches that explicitly take into account adaptation latency, either reactively (e.g., taking into account ongoing adaptations while planning new ones) or proactively (e.g., preferring quick planning algorithms or quick adaptation tactics in cases of urgency). In the long run, we hope to contribute to building better self-adaptive systems by raising the awareness of adaptation latency in the research community, and incorporating such aspects in approaches to self-adaptation.

Acknowledgments

Research leading to these results received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731678

(RestAssured). Useful comments of Javier Cámara on an earlier version of the paper are gratefully acknowledged.

References

1. Aschoff, R.R., Zisman, A.: Proactive adaptation of service composition. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 1–10. IEEE Press (2012)
2. Barna, C., Shtern, M., Smit, M., Tzerpos, V., Litoiu, M.: Model-based adaptive DoS attack mitigation. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 119–128. IEEE Press (2012)
3. Bartók, D., Mann, Z.Á.: A branch-and-bound approach to virtual machine placement. In: Proceedings of the 3rd HPI Cloud Symposium "Operating the Cloud". pp. 49–63 (2015)
4. Bennaceur, A., Zisman, A., McCormick, C., Barthaud, D., Nuseibeh, B.: Won't take no for an answer: Resource-driven requirements adaptation. In: Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 77–88 (2019)
5. Calinescu, R., Ghezzi, C., Kwiatkowska, M., Mirandola, R.: Self-adaptive software needs quantitative verification at runtime. Communications of the ACM 55(9), 69–77 (2012)
6. Cámara, J., Correia, P., De Lemos, R., Garlan, D., Gomes, P., Schmerl, B., Ventura, R.: Evolving an adaptive industrial software system to use architecture-based self-adaptation. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 13–22. IEEE Press (2013)
7. Cámara, J., de Lemos, R.: Evaluation of resilience in self-adaptive systems using probabilistic model-checking. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 53–62. IEEE Press (2012)
8. Cámara, J., Moreno, G.A., Garlan, D.: Stochastic game analysis and latency awareness for proactive self-adaptation. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 155–164. ACM (2014)
9. Cheng, S.W., Garlan, D., Schmerl, B.: Evaluating the effectiveness of the Rainbow self-adaptive system. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. pp. 132–141. IEEE (2009)
10. Faccin, J., Nunes, I.: Cleaning up the mess: a formal framework for autonomously reverting BDI agent actions. In: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems. pp. 108–118. ACM (2018)
11. Filieri, A., Maggio, M., Angelopoulos, K., d'Ippolito, N., Gerostathopoulos, I., Hempel, A.B., Hoffmann, H., Jamshidi, P., Kalyvianaki, E., Klein, C., et al.: Software engineering meets control theory. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 71–82. IEEE Press (2015)
12. Filieri, A., Maggio, M., Angelopoulos, K., D'ippolito, N., Gerostathopoulos, I., Hempel, A.B., Hoffmann, H., Jamshidi, P., Kalyvianaki, E., Klein, C., et al.: Control strategies for self-adaptive software systems. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 11(4), 24 (2017)

13. Gambi, A., Moldovan, D., Copil, G., Truong, H.L., Dustdar, S.: On estimating actuation delays in elastic computing systems. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 33–42. IEEE Press (2013)
14. Ghezzi, C., Pinto, L.S., Spoletini, P., Tamburrelli, G.: Managing non-functional uncertainty via model-driven adaptivity. In: 35th International Conference on Software Engineering (ICSE). pp. 33–42. IEEE (2013)
15. Grua, E.M., Malavolta, I., Lago, P.: Self-adaptation in mobile apps: a systematic literature study. In: Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 51–62 (2019)
16. Herbst, N.R., Kounev, S., Weber, A., Groenda, H.: BUNGEE: an elasticity benchmark for self-adaptive IaaS cloud environments. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 46–56. IEEE Press (2015)
17. Jamshidi, P., Cámara, J., Schmerl, B., Kästner, C., Garlan, D.: Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots. In: Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 39–50 (2019)
18. Kaddoum, E., Raibulet, C., Georgé, J.P., Picard, G., Gleizes, M.P.: Criteria for the evaluation of self-* systems. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. pp. 29–38. ACM (2010)
19. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
20. Kinneer, C., Coker, Z., Wang, J., Garlan, D., Goues, C.L.: Managing uncertainty in self-adaptive systems with plan reuse and stochastic search. In: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems. pp. 40–50. ACM (2018)
21. Mann, Z.Á.: Resource optimization across the cloud stack. IEEE Transactions on Parallel and Distributed Systems 29(1), 169–182 (2017)
22. Mann, Z.Á.: Two are better than one: An algorithm portfolio approach to cloud resource management. In: European Conference on Service-Oriented and Cloud Computing. pp. 93–108. Springer (2017)
23. Mann, Z.Á., Metzger, A.: Auto-adjusting self-adaptive software systems. In: IEEE International Conference on Autonomic Computing (ICAC). pp. 181–186. IEEE (2018)
24. Mao, M., Li, J., Humphrey, M.: Cloud auto-scaling with deadline and budget constraints. In: 11th IEEE/ACM International Conference on Grid Computing. pp. 41–48. IEEE (2010)
25. Moreno, G.A., Cámara, J., Garlan, D., Schmerl, B.: Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering. pp. 1–12. ACM (2015)
26. Moreno, G.A., Cámara, J., Garlan, D., Schmerl, B.: Efficient decision-making under uncertainty for proactive self-adaptation. In: IEEE International Conference on Autonomic Computing (ICAC). pp. 147–156. IEEE (2016)
27. Moreno, G.A., Kinneer, C., Pandey, A., Garlan, D.: DARTSim: An exemplar for evaluation and comparison of self-adaptation approaches for smart cyber-physical systems. In: Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 181–187 (2019)

28. Moreno, G.A., Strichman, O., Chaki, S., Vaisman, R.: Decision-making with cross-entropy for self-adaptation. In: Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 90–101. IEEE (2017)
29. Neamtiu, I.: Elastic executions from inelastic programs. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 178–183. ACM (2011)
30. Pascual, G.G., Pinto, M., Fuentes, L.: Run-time adaptation of mobile applications using genetic algorithms. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 73–82. IEEE Press (2013)
31. Salehie, M., Tahvildari, L.: Self-adaptive software: Landscape and research challenges. ACM Transactions on Autonomous and Adaptive Systems 4(2), 14 (2009)
32. da Silva, C.E., de Lemos, R.: Dynamic plans for integration testing of self-adaptive software systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 148–157. ACM (2011)
33. da Silva, C.E., da Silva, J.D.S., Paterson, C., Calinescu, R.: Self-adaptive role-based access control for business processes. In: Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 193–203. IEEE Press (2017)
34. Sousa, G., Rudametkin, W., Duchien, L.: Extending dynamic software product lines with temporal constraints. In: Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 129–139. IEEE Press (2017)
35. Tamura, G., Villegas, N.M., Muller, H.A., Duchien, L., Seinturier, L.: Improving context-awareness in self-adaptation using the DYNAMICO reference model. In: 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). pp. 153–162. IEEE (2013)
36. Van Der Burg, S., Dolstra, E.: A self-adaptive deployment framework for service-oriented systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 208–217. ACM (2011)
37. Villegas, N.M., Müller, H.A., Tamura, G., Duchien, L., Casallas, R.: A framework for evaluating quality-driven self-adaptive software systems. In: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 80–89. ACM (2011)
38. Vogel, T.: mRUBiS: An exemplar for model-based architectural self-healing and self-optimization. In: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems. pp. 101–107. ACM (2018)
39. Zeller, M., Prehofer, C.: Timing constraints for runtime adaptation in real-time, networked embedded systems. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 73–82. IEEE (2012)