To be published in the Proceedings of the 2024 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C): AI4AS Workshop

Meta-Adaptation Goals: Leveraging Feedback Loop Requirements for Effective Self-Adaptation

Raffaela Groner, Ricardo Diniz Caldas and Rebekka Wohlrab
Dept. of Computer Science and Engineering
Chalmers | University of Gothenburg
Gothenburg, Sweden
raffaela@chalmers.se, ricardo.caldas@chalmers.se, wohlrab@chalmers.se

***Abstract*— Self-* systems focus on optimizing systems regarding adaptation goals, e.g. availability, that relate to properties of the managed system. While these adaptation goals are important, the feedback loop must also be considered. It is problematic if systems adapt too often, make too time-consuming changes that involve insecure transition states, or react too slowly when planning adaptations. Current approaches do not allow to enforce requirements on the adaptation loop—including sensitivity, stability, and transition states. When planning adaptations, those meta-adaptation goals need to be considered and potentially traded off against traditional adaptation goals. Such a trade-off might entail, for instance, that it may be worth it to adapt a system less frequently, although the overall availability of the system could be higher with more adaptations.**

We propose to explicitly consider feedback loop requirements for self-adaptive systems. Concretely, we propose using machine learning to predict quality measures based on the state of the feedback loop and the managed system. This information is used to plan for better adaptations that consider the adaptation goals and the meta-adaptation goals of the feedback loop. The explicit representation of meta-adaptation goals and adaptation goals enables the optimization of adaptation strategies, trade-off analysis, and evaluation of adaptations.

***Index Terms*—feedback loop requirements, self-adaptation, machine learning, MAPE-K, adaptation goals**

I. Introduction

Self-* systems commonly focus on optimizing systems concerning adaptation goals that relate to properties of the managed system [1], such as performance, availability, or energy efficiency. Besides focusing on the qualities of the managed system, it is crucial to also consider the qualities of the feedback loop. While past research has explored the need to focus on properties of the adaptation loop, including sensitivity, stability, and transition states [2], [3], requirements on the feedback loop are often not explicitly taken into account during the planning and execution of adaptations.

A survey on self-adaptation in industry found that existing research does not tackle the risks that practitioners face when engineering self-adaptive systems, including performance degradation or reduced availability due to the adaptation process [4]. Existing works, e.g. [5]–[7], propose a model-based cost-benefit analysis for adaptation strategies. These models must be created by system experts who also know how to model the relationships between the system configurations and the desired quality requirements. Such a manual approach is very time-consuming, complex, and error-prone.

To tackle this issue, we advocate the idea of expanding the managed system-centered view by making the quality requirements of the feedback loop explicit and modeling them using machine learning. We call these quality requirements meta-adaptation goals (MAGs). Meta-adaptation goals are concerned with the sensitivity, stability, adaptation duration, and other quality attributes that concern the feedback loop and are, therefore, more complex than simple costs or benefits of applied adoption strategies. Examples of meta-adaptation goals are: "*The time it takes to execute the adaptation shall be less than 60s*." or "*The monitoring of the managed system should cause an overhead of less than 5%*."

We propose a systematic approach that considers meta-adaptation goals of the feedback loop as first-class citizens alongside the adaptation goals of the managed system. Our approach takes the adaptation goals of the managed system and the meta-adaptation goals into account when planning and implementing an adaptation. This requires an approach that can describe the relationship between the state of the managed system, the feedback loop, the adaptation goals, and meta-adaptation goals. However, such a description is not trivial due to the complex relationships between the individual aspects and the large number of possible configurations of a managed system. To this end, we propose using a machine learning-based regression model that predicts quality measures of a self-adaptive system based on the state of the feedback loop and the managed system. The regression model is needed because it is difficult to anticipate how long a planned adaptation takes and whether it will lead to a stable state. It is a data-driven method that is instantiated with simulation-based data and then continuously updated during runtime. This regression model can be used in the feedback loop to assess whether an adaptation that meets adaptation goals better than the current configuration improves the quality of the overall system or harms the system by violating meta-adaptation goals.

II. Illustrative Example

We exemplify our vision using the cyber-physical system DeltaIoT [8]. DeltaIoT is a real-world example of an Internet of Things system. The system enables dynamic adaptation of the network settings of individual motes (e.g., transmission power and spreading factor) to reduce energy consumption. To effectively implement such dynamic adaptation, Weyns

and Iftikhar implement ActivFORMS [9] leveraging timed automata modeling and statistical model checking to ensure correct and efficient adaptation at runtime. ActivFORMS has been evaluated and compared with RQV (Runtime Quantitative Verification), which uses quantitative verification to plan adaptations [10].

We use the comparison between ActivFORMS and RQV in [9] as the basis of our illustrative example. Our example system should trigger an adaptation to switch between ActivFORMS and RQV. We consider **package loss** and **energy consumption** as adaptation goals of the managed system to be optimized. The meta-adaptation goals for the feedback loop in our example are **adaptation time** and the memory usage to plan adaptations (**computational space**), since realistic environments require fast adaptations that do not consume a lot of computational resources.

Our approach aims to reach a configuration of the managed system and the feedback loop that achieves the best possible compromise between all four quality goals, depending on the current environment. For instance, ActivFORMS comes with a shorter adaptation time than RQV. However, the packet loss using ActivFORMS is higher than when using RQV [9].

III. Related Work

Quality of the feedback loop: Several works deal with the quality of the feedback loop as well as related meta-adaptation goals. For instance, ASMs have been used for the verification and validation of feedback loops [11]. In contrast to our work, related research does not explicitly focus on the meta-adaptation goals of the feedback loop and their compliance at runtime.

Awareness requirements are a related category of requirements that describe the success or failure of other requirements at runtime [12]. In comparison to the meta-adaptation goals we consider, such as the duration of an adaptation, awareness requirements describe expectations on the managed system. For example, a service should always be available regardless of the current configuration of the managed system.

Related work has acknowledged the need to consider requirements at the feedback loop level, similar to meta-adaptation goals. Lists of such quality attributes have been compiled and mapped to related quality attributes of the managed systems [2], [3]. Additionally, the Adaptive Strategies Metric Suite [13] measures various design and runtime properties of adaptation strategies. While these works give a good overview of possible meta-adaptation goals, they lack a systemic integration of these goals into the feedback loop. Additionally, they do not simultaneously consider the state of the feedback loop and the state of the managed system, even though both affect the fulfillment of meta-adaptation goals and adaptation goals of the managed system. We want to address both shortcomings.

Cost-benefit trade-off: Some works deal with cost-benefit considerations for adaptations.

For example, the work presented in [5], [6] is a runtime extension of the Cost-Benefit Analysis Method [14]. Like our approach, this approach relies on predictions of expected costs and benefits to evaluate adaptation strategies. However, their approach requires a manually defined runtime model. Thus, to create a correct and reliable prediction model, one must fully understand the managed system and the managing system, as well as their interrelations.

There are also cost-benefit analyses that evaluate the gain or loss of a configuration based on empirically constructed functions [7].

The presented studies [5]–[7] only consider one meta-adaptation goal, namely the costs of an adaptation. Other possible meta-adaptation goals, such as memory consumption during planning, monitoring overhead, or increased analysis time due to more complex configurations, are not considered. To be able to take such meta-adaptation goals into account, we suggest extending the respective components of the feedback loop to monitor their state. Furthermore, the approaches in [5]–[7] are based on manually creating a model or function that can map the current system state to multiple adaptation goals and meta-adaptation goals. Therefore, we propose to replace this complex, error-prone manual step by training a machine learning approach.

Meta-adaptation layer: Some studies suggest the introduction of a second feedback loop that adapts the original feedback loop so that it can evolve and react optimally in unknown circumstances. The second feedback loop is also often called the self-improvement, evolution, or meta-adaptation layer.

Some works propose to integrate a further feedback loop to optimize the adaptation strategy of the original loop at runtime [15], [16]. This evolution of the original feedback loop optimizes its behavior but only to improve adaptation goals of the managed system.

Other works propose to adapt the adaptation strategy, policies [17], or logic to improve the system so it can deal with situations that were unknown at design time [18]. Some works [15]–[18] aim to optimize the behavior of the feedback loop. They do not consider any meta-adaptation goals or possible trade-offs between adaptation goals and meta-adaptation goals while analyzing or planning an adaptation.

Machine learning and feedback loops: Several proposed approaches rely on machine learning.

Maggio et al. [19] investigated various decision-making techniques for self-optimization. In addition to heuristics of control theory, the authors also examined machine learning approaches and compared them in terms of their performance. A similar approach to ours [20] uses machine learning to map the influence of an adaptation on the adaptation goals. Several works use machine learning to reduce the adaptation space [21]–[23]. They all propose ways to enhance individual parts of the feedback loop by integrating machine learning techniques. Our approach, however, aims to consider meta-adaptation goals in the analysis and planning of an adaptation by using machine learning to shift the mainly one-sided focus on achieving the adaptation goals of the managed system.

There is also work such as [24], [25], which uses machine learning to predict the satisfaction of adaptation goals and

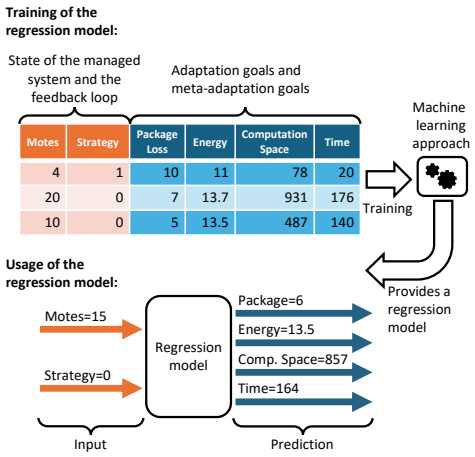

Fig. 1. Overview of training a machine learning approach

trigger an adaptation proactively. These works aim to optimize towards adaptation goals and do not consider the desired meta-adaptation goals for the feedback loop.

IV. Multi-output Regression

In our approach, we leverage multi-output regression to predict the quality measures related to the adaptation goals of the managed system and the meta-adaptation goals of the feedback loop. The idea is that when selecting adaptation strategies, the planner needs to be able to assess how long an adaptation will take, whether any insecure intermediate steps are involved and whether the planned adaptation is likely to lead the system to a stable state. Therefore, it is important to have realistic predictions of the quality measures related to the adaptation goals and the meta-adaptation goals.

Since we want to predict values for several target labels, a so-called multi-output regression [26] approach is a suitable solution. These approaches can map a vector of input data (features) to a vector of target labels and thus predict values for several dependent variables. Figure 1 illustrates how we can predict the expected package loss, energy consumption, computation space, and adaptation time (target labels) for a given number of motes and ActivFORMS or RQV as an adaptation strategy (features).

V. Meta-adaptation goals as first-class citizens

Our goal is to consider meta-adaptation goals (MAGs) of the feedback loop as first-class citizens in a self-adaptive system to improve its overall quality. To this end, we propose to extend all phases of the MAPE-K feedback loop and integrate a multi-output regression model. An overview of our proposed extensions of the MAPE-K feedback loop is shown in Figure 2. The centerpiece of our approach is a regression model that extends the functionality of the *Analyze* and *Plan* components to consider MAGs. Furthermore, we extend the *Monitor* and *Execute* components to integrate the optimization of our regression model as part of the feedback loop.

The regression model maps the monitored state of the managed system and the feedback loop to the expected quality measures in this state. For example, our regression model may predict the expected packet loss and energy consumption if we use a certain network configuration and how long it takes to adapt the network configuration of the managed system. This mapping can then be used to predict how possible target or intermediate states will affect the adaptation goals and MAGs by using a description of these states as input to the regression model.

With the help of a trade-off analysis method, the *Analyze* and the *Plan* phase can then evaluate whether a target state or an intermediate state achieves the desired trade-off between optimization of the adaptation goals of the managed system and compliance with the MAGs. Ultimately, this optimizes the managed system for adaptation goals and expands the feedback loop with additional knowledge to improve the quality of the overall self-adaptive system.

In the following, we describe the necessary actions, challenges, and our proposed approaches to tackle them in the context of the MAPE-K architecture. To trace challenges and ideas to tackle them, we assign identifiers using the schema **C**<ID> for challenges and **A**<ID> for proposed approaches. Table I provides an overview of the challenges and our proposed approaches.

TABLE I
Overview of the challenges and our proposed approaches.

	Challenge	Approach
Knowledge	**C1** Encoding the monitoring data. **C2** Encoding of quality requirements of the managed system and the MAGs. **C3** Identifying ML approach that provides a good regression model.	**A1** Assess suitability of different encodings and machine learning (ML) approaches.
Monitor	**C4** Suitable monitoring approach. **C5** Efficient data management for the training data.	**A2** Monitoring approaches from performance engineering. **A3** Clustering techniques to reduce amount of stored data.
Analyze	**C6** Large state space exploration when evaluating target states. **C7** Integrate analyses to assess complex MAGs. **C8** Interpretation model that indicates the best trade-off between adaptation goals and MAGs. **C9** Identify an approach to assess the quality of the regression model.	**A4** Examine approaches to prevent state space explosion. **A5** Possible reuse of safety/security analysis methods. **A6** Explore trade-off analyses to derive an interpretation model. **A7** Simulation-based experiments to identify suitable quality assessment methods.
Plan	**C10** Minimize the state space exploration during the evaluation of possible intermediate states.	**A8** Examine different existing approaches to prevent state space explosion. **A9** Discard inappropriate solutions at an early stage.
Execute	**C11** Parallel exec. of the adaptation of the managed system and the retraining of the regression model.	**A10** Examine existing parallelization approaches.

A. Knowledge

Proposal: We propose to extend the *Knowledge* component with a model that describes the relationships between the managed system, feedback loop, adaptation goals of the managed system, and MAGs.

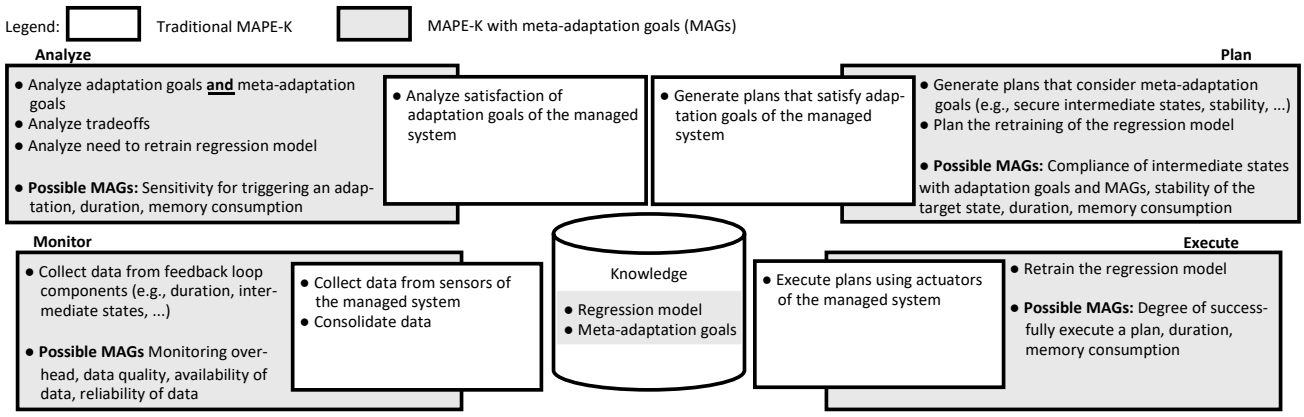

Fig. 2. Overview of our meta-adaptation goals (MAGs) approach

Implementation: We propose to train a multi-output regression model, using training data from simulations, that relates the state of the feedback loop and the managed system as input to the adaptation goals of the managed system and MAGs as output.

We acknowledge that the simulated data does not fully represent real-world values and, consequently, requires continuous updates of the regression model. We propose to integrate the regression model updates into the feedback loop. This means that the *Analyze* component triggers the adaptation of the managed system and the retraining of the multi-output regression model. To realize the retraining, we propose to collect the real-world data needed for the retraining during runtime by the *Monitor* component and store it in the knowledge base. Another component should perform the retraining to avoid affecting the performance of the feedback loop.

For example, in DeltaIoT, the managed system state is defined by the *number of active motes*, and the feedback loop state is defined by the instantiated *adaptation strategy*, either ActivFORMS or RQV. The multi-regression model uses these values as input and provides a prediction of the values for package loss, energy consumption, adaptation time, and computational space. Additionally, the simulator by Iftikhar et al. [8] can be used to obtain training data for the initial training of the model.

Challenges: Several challenges have to be addressed to obtain an appropriate regression model.

We need to examine whether the data obtained by the *Monitor* is suitable input data for the regression model and how the input needs to be encoded (**C1**). The quality requirements of the managed system and MAGs must be encoded so that they can be predicted by the regression model (**C2**). Finally, a machine learning (ML) approach needs to be chosen that can provide a multi-output regression model that maps system information to the adaptation goals of the managed system and MAGs (**C3**).

To address **C1**–**C3**, we propose to use experiments that compare different ML approaches and combinations of encodings for input and output data. Since the information about the system and the considered quality goals are system-specific, we plan to experiment with different systems (**A1**).

B. Monitor

Proposal: The *Monitor* component shall be extended to collect data on the components of the feedback loop.

Implementation: The collected data depends on the managed system, the adaptation goals of the managed system, and the MAGs under consideration. These also determine other data collection characteristics, such as whether a hardware or software monitor should be used.

The *Monitor* component for the self-adaptive DeltaIoT case collects the managed system state, values related to the adaptation goals, the current adaptation strategy, and values related to the MAGs.

In our example, the monitoring collects the number of active motes to observe the managed system and the adaptation strategy used to observe the feedback loop. In addition, the monitoring collects data to obtain the current values for the adaptation goals package loss and energy consumption and the MAGs adaptation time and computational space.

Challenges: One challenge is to decide which data needs to be collected and how (**C4**). If a property that influences the adaptation goals of the managed system or MAGs is not measured, the regression model cannot represent its influence. Measuring too many properties results in an unnecessary overhead, and the regression model might learn false relationships.

The implementation of the data collection is system-dependent and depends on the adaptation goals and MAGs under consideration since we plan to collect data for retraining the regression model at runtime. We also need to monitor the feedback loop as efficiently as possible to ensure the functionality of the managed system. Thus, we plan to investigate which already used monitoring approaches from the performance engineering area can be reused (**A2**).

The collected data needs to be stored since we want to use it to retrain the regression model. Thus, we need a suitable persistence approach that also realizes efficient data management (**C5**). A possible solution would be to cluster the

stored data regularly, e.g., using k-means clustering [27], and keep only selected representatives for each cluster (**A3**).

C. Analyze

Proposal: We propose to extend the *Analyze* component in two ways. Firstly, our regression model should be used to decide whether an adaptation makes sense regarding the adaptation goals of the managed system and the MAGs. Secondly, the quality of the regression model should be evaluated and, if necessary, a retraining should be triggered.

Implementation: We envision that the *Analyze* component must assess whether a possible adaptation improves or at least meets the desired MAGs and adaptation goals of the managed system. We consider the current state of the behavior or structure of the system and the feedback loop and assess whether any adaptations might lead to a better target state. For the evaluation of the target states, we plan to use our regression model. As shown in Figure 1, it relies on information about the state of the managed system and the feedback loop, as well as an envisioned adaptation strategy or target state. As output, our regression model predicts the values of the adaptation goals of the managed system and MAGs.

The *Analyze* component must also be able to evaluate the quality of our regression model used. If the prediction of the regression model for the given input values deviates too much from the real-world values, a new training of the regression model must be triggered.

In DeltaIoT, the *Analyze* component queries the regression model to understand whether there is a need for changing the adaptation strategy. The analyzer queries the regression model by providing the number of active motes, the number of motes that should be active to satisfy the managed system adaptation goals, and the available adaptation strategies (ActivFORMS or RQV). Consequently, the regression model returns a tuple of predicted package loss, energy consumption, computation space, and adaptation time.

Since the regression model returns a vector of values as a result, a trade-off analysis to assess adaptation options is necessary. Salama et al. [28] present a catalog of methods that can be used for trade-off analysis. Many of them are appropriate for MAGs as well, e.g., utility theory or multi-objective optimization. The trade-off analysis results in a model that is called *interpretation model* and allows to relate inputs and outputs for optimal adaptation.

Challenges: Our approach results in a more complex analysis since it has to explore the different target states (**C6**). Such exploration can be extremely time-consuming, so we need to investigate which existing techniques are applicable to prevent a state space explosion (**A4**).

More complex MAGs, such as safety or security, require additional analyses to map these requirements into measurable values, e.g., the probability of a fault during the execution of an adaptation (**C7**). Thus, techniques for carrying out these analyses as efficiently as possible must be examined (**A5**).

Our regression model provides one vector of values per prediction. Thus, to be able to compare the different vectors obtained for different target states in the *Analyze* component, we need an interpretation model (**C8**). For this purpose, we plan to investigate existing work on utility functions and the methods in [28] to realize an interpretation of the values (**A6**).

We assume that the acceptable deviation between the values predicted by the regression model and the real-world values depends on the use case. For example, the deviation for a safety-relevant system should be smaller than for an online store (**C9**). Thus, we need to examine if simulations can help to define suitable thresholds and how often the quality of the regression model should be evaluated (**A7**).

D. Plan

Proposal: We propose to use our regression model in the *Plan* component to evaluate the target state and the intermediate states of an adaptation. This should ensure that intermediate states also consider the adaptation goals of the managed system and MAGs.

Implementation: Like the analysis, we need an interpretation model (cf. **C8**) for the *Plan* component that helps to evaluate the various possible intermediate states of the system during its adaptation. To realize this, we plan to reuse our interpretation model from the *Analyze* component.

In the DeltaIoT example, an adaptation can consist of adding further motes and changing the adaptation strategy. To realize such a multistage adaptation, several steps are necessary, which lead to intermediate states that should also fulfill all goals as optimally as possible. For example, it may be better to switch the adaptation strategy first and then integrate more motes into the system since as the number of motes increases, the computational space of ActivFORMS increases likewise and makes it unfeasible to generate plans.

Challenges: Since evaluating all possible intermediate states requires a state space exploration, countermeasures must be taken, as in the *Analyze* component, to prevent an explosion of the state space (**C10**). We plan to examine the application of reuse mechanisms to reuse (partial) solutions (**A8**). Additionally, plans that already compromise the adaptation goals and MAGs should be detected at an early stage of the state space exploration and should be discarded (**A9**).

E. Execute

Proposal: The *Execute* component carries out the adaptation and the retraining of the regression model.

Implementation: The *Execute* component must be extended so that it can (i) adapt the managed system and (ii) update the regression model. Since both types of changes can occur simultaneously, the *Execute* component must be able to execute them in parallel.

In our example, the execution of an adaptation can affect three different components. i) The managed system by adding or removing motes. ii) The feedback loop by changing the adaptation strategy. iii) The regression model, which is replaced after retraining.

Challenges: Since it is possible that retraining the regression model is necessary while the managed system is adapting, the

execution phase must be able to be parallelized (**C11**). Thus, we plan to examine which existing parallelization techniques can be reused (**A10**).

VI. Conclusion

In the past, requirements engineering for self-adaptive systems has mainly focused on adaptation goals of the managed system. However, in practice, requirements of the feedback loop are often just as crucial for the overall quality of self-* systems. To this end, we propose modeling meta-adaptation goals as first-class citizens.

We describe the steps needed to arrive at our vision by elaborating on 11 challenges along with proposed approaches. Concretely, we propose using a regression model that maps monitoring data about the managed system and the feedback loop to corresponding quality measures. We enrich the *Analyze* and *Plan* components with our regression model so that they find a suitable trade-off between the optimization of adaptation goals and meta-adaptation goals.

Overall, our proposed approach can contribute towards self-adaptive systems that better meet their stakeholders' needs. Systems that consider meta-adaptation goals can trade off the adaptation overhead against the expected increase in the quality of the managed system. This enables self-adaptive systems to avoid unnecessary, instable, or overly costly adaptations and thereby become more effective and robust.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] D. Weyns, *An introduction to self-adaptive systems: A contemporary software engineering perspective*. John Wiley & Sons, 2020.

[2] A. Farahani, E. Nazemi, G. Cabri, and A. Rafizadeh, "An evaluation method for self-adaptive systems," in *SMC'16*, Oct. 2016, pp. 002 814–002 820.

[3] N. M. Villegas, H. A. Müller, G. Tamura, L. Duchien, and R. Casallas, "A framework for evaluating quality-driven self-adaptive software systems," in *SEAMS '11*, May 2011, pp. 80–89.

[4] D. Weyns, I. Gerostathopoulos, N. Abbas, J. Andersson, S. Biffl, P. Brada, T. Bures, A. Di Salle, M. Galster, P. Lago, G. Lewis, M. Litoiu, A. Musil, J. Musil, P. Patros, and P. Pelliccione, "Self-adaptation in industry: A survey," *ACM Trans. Auton. Adapt. Syst.*, vol. 18, no. 2, may 2023.

[5] M. J. Van Der Donckt, D. Weyns, M. U. Iftikhar, and R. Kumar Singh, "Cost-benefit analysis at runtime for self-adaptive systems applied to an internet of things application," in *ENASE'18*, 2018, p. 478–490.

[6] J. Van Der Donckt, D. Weyns, M. U. Iftikhar, and S. S. Buttar, "Effective decision making in self-adaptive systems using cost-benefit analysis at runtime and online learning of adaptation spaces," in *Evaluation of Novel Approaches to Software Engineering*, E. Damiani, G. Spanoudakis, and L. A. Maciaszek, Eds. Cham: Springer International Publishing, 2019, pp. 373–403.

[7] I. Gerostathopoulos, C. Raibulet, and E. Alberts, "Assessing self-adaptation strategies using cost-benefit analysis," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, 2022, pp. 92–95.

[8] M. U. Iftikhar, G. S. Ramachandran, P. Bollansée, D. Weyns, and D. Hughes, "DeltaIoT: A self-adaptive internet of things exemplar," in *SEAMS'17*. IEEE, 2017, pp. 76–82.

[9] D. Weyns and U. M. Iftikhar, "ActivFORMS: A formally founded model-based approach to engineer self-adaptive systems," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 1, pp. 1–48, 2023.

[10] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic QoS management and optimization in service-based systems," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2010.

[11] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation," in *SEAMS'15*, May 2015, pp. 13–23, iSSN: 2157-2321.

[12] V. E. Silva Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, "Awareness requirements for adaptive systems," in *SEAMS'11*, 2011, pp. 60–69.

[13] K. Kraaijveld, C. Raibulet *et al.*, "Asms: A metrics suite to measure adaptive strategies of self-adaptive systems," in *Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering-ENASE*, vol. 2023. Science and Technology Publications, Lda, 2023, pp. 238–249.

[14] C. M. U. Software Engineering Institute, "Cost benefit analysis method (CBAM)," 2018, accessed: 03-07-2024. [Online]. Available: https://www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm

[15] I. Gerostathopoulos, T. Bures, P. Hnetynka, A. Hujecek, F. Plasil, and D. Skoda, "Strengthening Adaptation in Cyber-Physical Systems via Meta-Adaptation Strategies," *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 3, pp. 13:1–13:25, Apr. 2017.

[16] V. Lesch, M. Hadry, C. Krupitzer, and S. Kounev, "Self-aware Optimization of Adaptation Planning Strategies," *ACM Trans. Auton. Adapt. Syst.*, vol. 18, no. 3, pp. 10:1–10:35, Sep. 2023.

[17] G. Perrouin, B. Morin, F. Chauvel, F. Fleurey, J. Klein, Y. Le Traon, O. Barais, and J.-M. Jézéquel, "Towards flexible evolution of Dynamically Adaptive Systems," in *ICSE'12*, Jun. 2012, pp. 1353–1356, iSSN: 1558-1225.

[18] C. Krupitzer, J. Otto, F. M. Roth, A. Frömmgen, and C. Becker, "Adding Self-Improvement to an Autonomic Traffic Management System," in *ICAC'17*, 2017, pp. 209–214, iSSN: 2474-0756.

[19] M. Maggio, H. Hoffmann, A. V. Papadopoulos, J. Panerati, M. D. Santambrogio, A. Agarwal, and A. Leva, "Comparison of decision-making strategies for self-optimization in autonomic computing systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 4, dec 2012.

[20] A. Elkhodary, N. Esfahani, and S. Malek, "Fusion: a framework for engineering self-tuning self-adaptive software systems," in *FSE'10*, 2010, p. 7–16.

[21] O. Gheibi, D. Weyns, and F. Quin, "On the impact of applying machine learning in the decision-making of self-adaptive systems," in *SEAMS'21*, 2021, pp. 104–110.

[22] F. Quin, D. Weyns, T. Bamelis, S. Singh Buttar, and S. Michiels, "Efficient analysis of large adaptation spaces in self-adaptive systems using machine learning," in *SEAMS'19*, 2019, pp. 1–12.

[23] J. Van Der Donckt, D. Weyns, F. Quin, J. Van Der Donckt, and S. Michiels, "Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals," in *SEAMS'20*, 2020, p. 20–30.

[24] H. Muccini and K. Vaidhyanathan, "A machine learning-driven approach for proactive decision making in adaptive architectures," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, 2019, pp. 242–245.

[25] ——, "Leveraging machine learning techniques for architecting self-adaptive IoT systems," in *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2020, pp. 65–72.

[26] H. Borchani, G. Varando, C. Bielza, and P. Larranaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[27] J. Wu, *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012.

[28] M. Salama, R. Bahsoon, and N. Bencomo, "Managing trade-offs in self-adaptive software architectures: A systematic mapping study," *Managing trade-offs in adaptable software architectures*, pp. 249–297, 2017.