Expressing the Adaptation Intent as a Sustainability Goal

Ilias Gerostathopoulos
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
i.g.gerostathopoulos@vu.nl

Claudia Raibulet
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
c.raibulet@vu.nl

Patricia Lago
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
p.lago@vu.nl

ABSTRACT

Adaptation and sustainability are two key challenges leading the development of software-systems nowadays. Adaptation denotes the capacity of a system to cope with variations and uncertainties at runtime in order to continue providing its functionalities with certain quality levels, *notwithstanding change*. But how can adaptation and its intent be expressed at design time so that to analyze its possible impact at runtime over a long period of time? To answer this question we look at adaptation from the sustainability point of view. Sustainability denotes the capacity of a system to both endure and preserve its function *over time*. We propose an approach which uses decision maps to make sustainability-driven decisions for adaptation in a systematic way. The proposed approach is illustrated through two self-adaptive exemplars as illustrative cases.

KEYWORDS

Self-adaptive systems, adaptation intent, sustainability goal.

ACM Reference Format:
Ilias Gerostathopoulos, Claudia Raibulet, and Patricia Lago. 2022. Expressing the Adaptation Intent as a Sustainability Goal. In *New Ideas and Emerging Results (ICSE-NIER'22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3510455.3512776

1 INTRODUCTION

With the current pace of digitalization, the role of software-intensive systems is becoming increasingly profound. Modern software-intensive systems such as robotic assistants, self-driving cars, and industrial automation systems need to operate in a variety of operational contexts and are expected to be able to deal with emerging uncertainties in these contexts [19]. For instance, they are expected to continue operating when sensors fail, faults occur, and their usage increases rapidly. Even more, they are expected to balance off properties such as operation cost and user-perceived performance to optimize themselves.

To deal with runtime uncertainties, such software-systems are typically designed as self-adaptive, *i.e.,* systems that are able to modify their structure or behavior at runtime in response to external or internal stimuli [20]. For instance, a system with autoscaling capabilities is able to change its deployment architecture by adding or removing resources (typically servers) in response to changes in its usage patterns (*e.g.,* increase in the number of user requests).

ICSE-NIER'22, May 21–29, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9224-2/22/05.
https://doi.org/10.1145/3510455.3512776

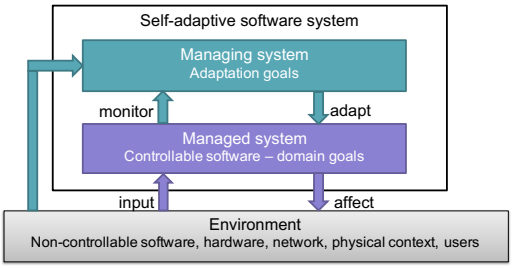

Figure 1: A self-adaptive system view (adapted from [22])

A canonical view of a self-adaptive system distinguishes between a managed system (the part of the system that can be changed at runtime) and a managing system (the part of the system responsible for performing the runtime changes to the managed part) [22]. As depicted in Figure 1, the managing system needs to continuously monitor both the managed system and its environment (*e.g.,* number of users) to decide on the actions to undertake.

In practice, the functionality of continuously monitoring a system and its environment, reasoning over the monitored data, deciding on an adaptation action (*e.g.,* deploying a new server), and performing it at runtime is often hard to design and develop, and even harder to analyse and test. To make matters worse, a managing system not properly tuned can result in oscillations in performance, unstable user experience, extra operational costs or even failures—on top of the cost of the extra complexity added to the system.

For self-adaptive systems to be truly successful they need to accommodate changes with a certain degree of autonomy. This way, they are expected to act upon potentially unexpected changes while preserving the original *intent* of the managed system over time [9]. In other words, the adaptation intent should correspond to the one of the managed system. To this end, there is a need to: (i) define the intent behind the adaptation functionality, (ii) develop the functionality to meet its intent, and (iii) assess the impact of the functionality based on the level its intent is actually met *over time*.

The novel idea we put forward in this paper is to express the *adaptation intent* of a software-system as a *sustainability goal* it needs to satisfy, *i.e.,* the network of quality concerns balanced over time [15]. So far, sustainability goals have been used in software projects to describe qualities that need to be minimized, maximized, or kept balanced to make a project sustainable or ensure that a project contributes to sustainability.

Interestingly, both the concepts of sustainability goal and sustainability-relevant quality concerns (in the technical-, economic-, environmental-, and social dimensions [8] of sustainability) can be used in framing the endurance and long-term success of an adaptation solution. This allows us to use methods and models originally

proposed to capture sustainability-relevant quality concerns (concerns for short) in software engineering (*e.g.,* decisions maps [13]) to define and assess the intent of self-adaptive systems, too.

The rest of the paper is organized as follows. Our vision and proposed approach are introduced in Section 2 which also illustrates the application of the approach to two well-known exemplars in self-adaptive software. Section 3 discusses our approach, while Sections 4 and 5 follow with related works and future plans, respectively.

2 FRAMING ADAPTATION INTENT AS A SUSTAINABILITY GOAL

This section introduces our vision and approach, and applies it to two well-known self-adaptation exemplars as illustrative cases.

2.1 The Vision

Adaptation denotes the capacity of a system to address expected or unexpected variations and uncertainties (variations for short) at runtime in order to continue providing its functionalities with certain quality levels. The occurrence of variations means that something has changed in the execution of a system or its context. To counteract such changes, the system adapts. While the adaptation mechanisms generally focus on counteracting changes that are applied in the short-term, we often forget about the overall adaptation intent, which is—or should be—to preserve the stability of the system in the long-term. The main research question is: *How can we express the adaptation and its intent at design time, so to analyze its possible impact at runtime, and over a long period of time?*

To answer this question we look at adaptation from the sustainability point of view. In the context of software engineering, sustainability denotes the capacity of a software-system to both endure and preserve its function *over time*[1]. Even if adaptation and sustainability seem to have different objectives, both are related to the good functioning/health of a system. While adaptation is framed in a short period of time (*i.e.,* it addresses variations promptly), sustainability is framed in a long period of time (*i.e.,* it addresses the impact of a system over time). Hence, through sustainability, adaptation has the potential to enrich its short-term perspective with an additional long-term perspective.

To make it possible to reason at design time in a systematic way about the adaptation and its possible impacts, and to uncover and make explicit the long-term adaptation intent, we envision adaptation as illustrated in Figure 2, where:

- The *adaptation boundaries* delineate the acceptable quality of the system. Adaptation may increase or decrease the overall quality of the system (the green and red arrows), but always within these boundaries.
- The *adaptation intent* is expressed as a sustainability goal, *i.e.,* as a network of positive and negative effects on the overall quality over time and within the adaptation boundaries. For instance, a negative effect may be an increase in the energy consumption of a software system, while a positive effect may be increased elasticity.

[1]In general, sustainability in software engineering has a much broader scope than the software-systems themselves [1]. To illustrate our idea, however, we limit the discussion to such scope, and touch base on the further implications in Section 5.

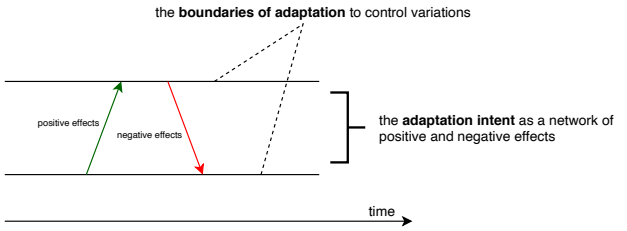

Figure 2: Adaptation explained as a sustainability goal

2.2 The Proposed Approach

Essentially, our proposal is to express the adaptation intent as a sustainability goal. To do this, we borrow from the software sustainability literature the "decision maps" (DMs), a tool that helps making sustainability-driven design decisions [13] in a systematic way. Once the sustainability goal is defined with the help of DMs, the adaptation boundaries are also identified and used to assess the adaptation intent over time.

The main idea of a DM is to capture the architectural design and quality concerns relevant to sustainability and frame them around the four sustainability dimensions (see Figure 3). In practice, a DM model consists of (i) the features (or requirements) that a project needs to realize, (ii) the concerns, (iii) the effects between a feature and a concern or between two concerns. Such effects are always one-directional and capture the knowledge, or assumption, the designer has on the type of effect, namely positive, negative, or undecided. Throughout a project, a DM can be updated to reflect *e.g.,* changes in requirements or a better-informed understanding of the expected or actual effects.

DMs have proven useful in the exploration of the design space [13]. They also help stakeholders uncover and frame the sustainability goal(s) of a certain project. In general, a sustainability goal describes a concern that needs to be optimized (minimized or maximized) or kept within a threshold. Clearly, the goal of a project can be broken down into several sub-goals that need to be jointly satisfied.

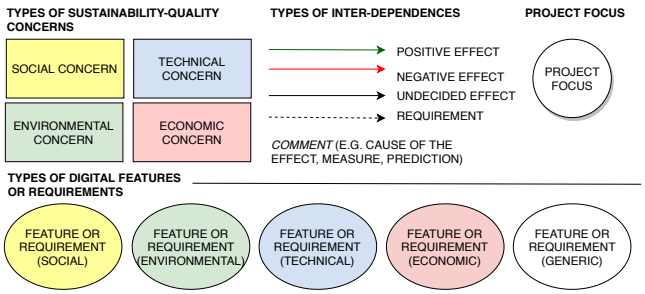

Figure 3: Decision Map notation

In the following, we show how DMs can be used for capturing the quality concerns of a managing system and expressing the adaptation intent as a sustainability goal. We do this by focusing on two well-known exemplars from the self-adaptation community[2], SWIM and DeltaIoT.

[2]Self-Adaptive Exemplars: www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars

2.3 SWIM

SWIM (Simulator for Web Infrastructure and Management) is a self-adaptation exemplar that simulates a multi-tier web application [16]. This application consists of a web and a database tier. The web tier serves client requests by having a load balancer assigning each request to one of the application servers in the web tier. A request is served by the assigned server by retrieving the necessary data from the database tier and rendering dynamic HTML pages.

The web application needs to deal with dynamic request load (the main source of uncertainty) and keep the system running without failures or negative impacts to its stakeholders (clients, but also service owners). To this end, it provides two ways to adapt itself at runtime: (1) add or remove application servers—similar to classic autoscaling policies, (2) adjust a parameter that controls the number of requests that serve additional content (*e.g.*, advertisements) on top of the mandatory content to be included in a response.

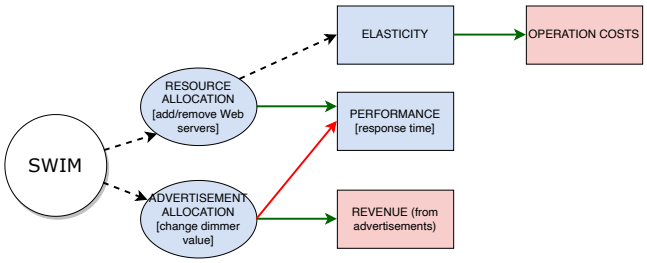

Figure 4: Decision map for SWIM

Figure 4 depicts the DM for SWIM. With respect to features that should be supported in the managing system of SWIM, we identify two technical ones, namely resource and advertisement allocation. The related concerns are both of technical and economic nature. In particular, the advertisement allocation is expected to have a positive impact on the generated ad-based revenue—an economic concern—and a negative effect on the performance visible to the clients as request response time—a technical concern. The reasoning behind this is that requests that do not include advertisements impose less load to the system. On the other hand, resource allocation is expected to have a positive effect on the same performance concern. Resource allocation also supports the elasticity concern which, in turn, is expected to have a positive effect on the operational costs of the system measured by the uptime of the application servers—an economic concern.

For the managing system of SWIM to be sustainable, it needs to preserve *over time* its overall sustainability goal. This involves a number of subgoals expressed by the concerns identified in the DM, namely:

- Minimizing the operation costs;
- Maximizing the revenue;
- Keeping the response time (performance) within a threshold.

2.4 DeltaIoT

DeltaIoT is a self-adaptation exemplar from the domain of Internet of Things (IoT) [10]. It consists of a number of motes (sensor devices) that communicate with each other and with an IoT gateway via a LoRa multi-hop wireless network. In particular, the motes periodically sense different physical attributes (temperature, occupancy status, object tracking) and send their measurements to the gateway, possibly by relaying the data packets via intermediate motes, for central storage and analysis.

The IoT application deals with the different levels of interference in wireless communication, the variable traffic load (number of messages to send), and the fluidity in the architecture of the system as motes may fail, new motes may appear, and motes may be moved around. To this end, the system itself can at runtime: (1) change the transmission power of each mote, (2) change the spreading factor of each mote (*i.e.,* bit-encoding per symbol of transmitted packet), (3) modify the path a packet uses to reach the gateway.

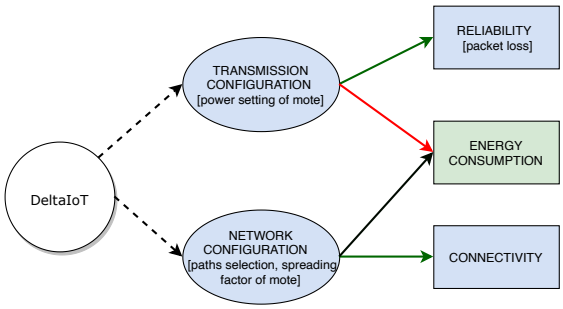

Figure 5: Decision map for DeltaIoT

As shown in Figure 5, the DM for DeltaIoT comprises two technical features, transmission- and network configuration. In particular, transmission configuration is expected to have a positive effect on the reliability concern, *i.e.,* the packet loss in the network, while it is expected to negatively affect the energy consumption, since a higher transmission power for a mote yields lower probability of packet loss but higher energy spent in the transmission. On the other side, network configuration encompasses strategies to select a path for a packet to be delivered to the gateway, and controls the spreading factor of each mote. This feature is expected to have a positive effect on the connectivity (ensuring that each mote is always connected via a multi-hop network to the gateway). Its effect on energy consumption is undecided, since an increase (resp. decrease) in the spreading factors can increase (resp. decrease) the energy spent, while choosing different paths to balance the re-transmission load is expected to affect the energy consumption positively.

All in all, the sustainability goal of the managing system is to:

- Minimize the energy consumption;
- Maximize the reliability (equivalently, minimize packet loss);
- Ensure the connectivity of the system.

In applying our proposal to SWIM and DeltaIoT, we observe that we can effectively express the sustainability goal of each managing system in terms of the concerns identified in the respective DM; and that such sustainability goal captures the underlying adaptation intent of each system.

3 DISCUSSION

In this work we aim at providing a systematic way to capture and ultimately measure the sustainability of a managed system, or in

other words, assess whether its quality over time can be bounded. We purposely leave out of the approach and DM notation the modeling of other aspects important to self-adaptive systems, such as the modeling of the context, as we believe it can complement, not replace, other architectural design approaches in self-adaptive systems that focus on context and adaptation actions (*e.g.,* [11]).

DMs are a starting point for understanding the trade-offs between the different concerns at design time. Most importantly, for each concern, relevant metrics can be identified to measure its degree of satisfaction. So far, the Goal-Question-Metric (GQM) has been used to identify metrics related to a particular concern [8], but that is only one possibility. Such metrics can be used to both evaluate a *new* self-adaptation mechanism, and compare different self-adaptation mechanisms [7] with respect to their sustainability.

Concretely, once each concern is quantified by one or more metrics, the satisfaction of the overall sustainability goal of a system can be quantified, too. This will typically involve several sub-goals (three in the case of SWIM or DeltaIoT). Assuming that static weights can be assigned to each sub-goal, it is possible to calculate a value capturing the overall quality of the system. Moreover, assuming the best and worse case (but still acceptable) scenarios for each of the sub-goals and calculating the quality for those scenarios, would help us determine the adaptation boundaries depicted in Figure 2. Then, the sustainability of the managed system over time could be assessed by checking whether the values of overall quality fall within the adaptation boundaries.

We envisage using the Sustainability Assessment Framework (SAF) toolkit [3] (which includes the DMs) that leverages the mentioned metrics as a sustainability goal; hence measuring the contribution of each metric to the sustainability of a system design.

The impact of expressing a software adaptation intent as a sustainability goal would potentially help software engineers create self-adaptive systems that endure change in the long-term *by design* — hence addressing our main research question (cf. Section 2).

4 RELATED WORK

To the best of our knowledge, there is no similar research expressing an adaptation intent of a software-system as a sustainability goal. The most-related available approaches investigate the benefits of adaptation with respect to its cost. For example, Tao Chen *et al.* [2] introduce the term *temporal adaptation debt*, *i.e.,* a technical debt which captures the economic health (*e.g.,* net debt) of an adaptive system and indicates whether an adaptation should be performed or not (*e.g.,* to obtain net profit) at runtime. A situation-dependent approach to measure the degree of adaptation to avoid possible conflicts between the benefits of adaptation and stability issues in the system or user acceptance is shown in [21]. The leading idea is to adapt if the adaptation has an overall positive effect in the system, or avoid adaptation otherwise. Esfahani *et al.* [4] propose a general quantitative approach to tackle the complexity of automatically making adaptation decisions under internal uncertainties at runtime. Further, they investigate possible positive and negative effects of the adaptation decisions. The same authors extended their work in [5] to summarize the main sources of uncertainty and to outline the importance of considering uncertainty in the adaptation decision process. They also argue that the optimality of adaptation decisions should be considered as a range of values, not just a single value at runtime. The quality-driven adaptive continuous experimentation approach in [12] aims to reduce development risks and operational costs through experiments at runtime to investigate variants in infrastructure configuration and architectural design.

The main differences between our approach and the existing solutions are: (i) available solutions consider the trade-off between the positive and negative effects of adaptation at runtime for each single adaptation, while we consider the trade-off between such effects at design time as well as over time, *i.e.,* to address sustainability, and (ii) our proposal outlines that the average between positive and negative adaptation effects should be in a range of values to address sustainability, again, in the long-term.

5 FUTURE PLANS

Identifying and collecting concerns and related metrics for self-adaptive systems. We plan to support software engineers in identifying the adaptation intent by providing a catalogue of sustainability-related concerns that are important and recurring in self-adaptive systems. As a starting point, we will collect such adaptation concerns by both surveying existing literature in self-adaptive systems (and in particular surveys related to the evaluation of self-adaptive systems, *e.g.,* [7, 17, 18]) and by performing interviews and focus groups with experts from industry and academia. We will also focus on the way such concerns are typically combined in specifying adaptation intents. Finally, for each concern, we will extract the metrics that can be used to quantify it [6].

Tool-based support for specifying and evaluating adaptation intents over time. Having an overview of the various concerns (*e.g.,* performance) and the related metrics (*e.g.,* "average response time per five minutes"), we plan to provide tools to model the different concerns in decision maps, and associate metrics to them in an intuitive way. For instance, such association can be done via selecting a data stream (*e.g.,* response time) from the list of data streams monitored by the managing system and selecting a function (*e.g.,* average) to be applied over a data window (*e.g.,* five minutes). In a similar vein, we should be able to specify sustainability goals in terms of the quantified concerns and create a service that continuously measures them over time [14].

Evaluation of capturing and measuring sustainability of self-adaptive systems. We plan to use the DM notation and modeling process explored in this paper to capture the sustainability concerns of SWIM and DeltaIoT with more self-adaptive systems, including industrial-grade systems, to evaluate the applicability and also evolve the proposed approach. To evaluate the usability of the approach, we intend to perform both case studies and controlled experiments. Finally, we envision that our proposed approach and related tool support will help build the foundations for a systematic process that covers the different phases of the design and development of a self-adaptive system.

ACKNOWLEDGEMENT

We would like to thank Michele Pugno for his valuable feedback.

REFERENCES

[1] Coral Calero, Mª Ángeles Moraga, and Mario Piattini. 2021. Introduction to Software Sustainability. In *Software Sustainability*, Coral Calero, Mª Ángeles Moraga, and Mario Piattini (Eds.). Springer International Publishing, 1–15.

[2] Tao Chen, Rami Bahsoon, Shuo Wang, and Xin Yao. 2018. To Adapt or Not to Adapt?: Technical Debt and Learning Driven Self-Adaptation for Managing Runtime Performance. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering, ICPE 2018, Berlin, Germany, April 09-13, 2018*, Katinka Wolter, William J. Knottenbelt, André van Hoorn, and Manoj Nambiar (Eds.). ACM, 48–55. https://doi.org/10.1145/3184407.3184413

[3] Nelly Condori-Fernandez and Patricia Lago. 2018. Characterizing the contribution of quality requirements to software sustainability. *The Journal of systems and software* 137, 3 (2018), 289–305. https://www.sciencedirect.com/science/article/pii/S0164121217302984

[4] Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek. 2011. Taming uncertainty in self-adaptive software. In *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, Tibor Gyimóthy and Andreas Zeller (Eds.). ACM, 234–244. https://doi.org/10.1145/2025113.2025147

[5] Naeem Esfahani and Sam Malek. 2013. Uncertainty in Self-Adaptive Software Systems. In *Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers (Lecture Notes in Computer Science, Vol. 7475)*, Rogério de Lemos, Holger Giese, Hausi A. Müller, and Mary Shaw (Eds.). Springer, 214–238. https://doi.org/10.1007/978-3-642-35813-5_9

[6] Ilias Gerostathopoulos, Claudia Raibulet, and Elvin Alberts. 2022. Assessing Self-Adaptation Strategies Using Cost-Benefit Analysis. In *Proc. of 44th International Conference on Software Engineering Companion, ICSA 2022*. In press.

[7] Ilias Gerostathopoulos, Thomas Vogel, Danny Weyns, and Patricia Lago. 2021. How do we Evaluate Self-adaptive Software Systems?: A Ten-Year Perspective of SEAMS. In *Proceedings of the IEEE/ACM 16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2021)*. IEEE Computer Society, 59–70. https://doi.org/10.1109/SEAMS51251.2021.00018

[8] Sarthak Gupta, Patricia Lago, and Roel Donker. 2021. A Framework of Software Architecture Principles for Sustainability-driven Design and Measurement. In *18th IEEE International Conference on Software Architecture Companion, ICSA Companion 2021, Stuttgart, Germany, March 22-26, 2021*. IEEE, 31–37. https://doi.org/10.1109/ICSA-C52384.2021.00012

[9] Marieke Huisman, Herbert Bos, Sjaak Brinkkemper, Arie van Deursen, Jan Friso Groote, Patricia Lago, Jaco van de Pol, and Eelco Visser. 2016. Software that Meets Its Intent. In *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications (Lecture Notes in Computer Science)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 609–625. https://doi.org/10.1007/978-3-319-47169-3_47

[10] Muhammad Usman Iftikhar, Gowri Sankar Ramachandran, Pablo Bollansée, Danny Weyns, and Danny Hughes. 2017. DeltaIoT: A Self-Adaptive Internet of Things Exemplar. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 76–82.

[11] Muhammad Usman Iftikhar and Danny Weyns. 2014. ActivFORMS: active formal models for self-adaptation. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2014)*. Association for Computing Machinery, Hyderabad, India, 125–134. https://doi.org/10.1145/2593929.2593944

[12] Miguel A. Jiménez, Luis F. Rivera, Norha M. Villegas, Gabriel Tamura, Hausi A. Müller, and Nelly Bencomo. 2019. An architectural framework for quality-driven adaptive continuous experimentation. In *Proceedings of the Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution, RCoSE-DDrEE@ICSE 2019, Montreal, QC, Canada, May 27, 2019*. IEEE / ACM, 20–23. https://doi.org/10.1109/RCoSE/DDrEE.2019.00012

[13] Patricia Lago. 2019. Architecture Design Decision Maps for Software Sustainability. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. 61–64.

[14] Patricia Lago and Toon Jansen. 2011. Creating Environmental Awareness in Service Oriented Software Engineering. In *Service-Oriented Computing*, E. Michael Maximilien, Gustavo Rossi, Soe-Tsyr Yuan, Heiko Ludwig, and Marcelo Fantinato (Eds.). Springer, 181–186.

[15] Patricia Lago, Roberto Verdecchia, Nelly Condori-Fernandez, Eko Rahmadian, Janina Sturm, Thijmen van Nijnanten, Rex Bosma, Christophe Debuysscher, and Paulo Ricardo. 2021. Designing for Sustainability: Lessons Learned from Four Industrial Projects. In *Advances and New Trends in Environmental Informatics*. Springer International Publishing, 3–18.

[16] Gabriel A Moreno, Bradley Schmerl, and David Garlan. 2018. SWIM: an exemplar for evaluation and comparison of self-adaptation approaches for web applications. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems* (Gothenburg, Sweden) *(SEAMS '18)*. Association for Computing Machinery, New York, NY, USA, 137–143.

[17] Claudia Raibulet and Francesca Arcelli Fontana. 2017. Evaluation of self-adaptive systems: a women perspective. In *11th European Conference on Software Architecture, ECSA 2017, Companion*. 23–30. https://doi.org/10.1145/3129790.3129825

[18] Claudia Raibulet, Francesca Arcelli Fontana, and Simone Carettoni. 2020. A preliminary analysis of self-adaptive systems according to different issues. *Softw. Qual. J.* 28, 3 (2020), 1213–1243. https://doi.org/10.1007/s11219-020-09502-5

[19] Andres J. Ramirez, Adam C. Jensen, and Betty H. C. Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2012, Zurich, Switzerland, June 4-5, 2012*, Hausi A. Müller and Luciano Baresi (Eds.). IEEE Computer Society, 99–108. https://doi.org/10.1109/SEAMS.2012.6224396

[20] Mazeiar Salehie and Ladan Tahvildari. 2009. Self-Adaptive Software: Landscape and Research Challenges. *ACM TAAS* 4, 2, May (2009), 1–40. https://doi.org/10.1145/1516533.1516538

[21] Sven Tomforde and Martin Goller. 2020. To Adapt or Not to Adapt: A Quantification Technique for Measuring an Expected Degree of Self-Adaptation. *MDPI Computers* 9, 1 (2020), 21. https://doi.org/10.3390/computers9010021

[22] Danny Weyns. 2018. Engineering Self-Adaptive Software Systems – An Organized Tour. In *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. 1–2. https://doi.org/10.1109/FAS-W.2018.00012