Software Sustainability in the Age of Everything as a Service

Andrikopoulos, Vasilios; Lago, Patricia

published in
Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future: Essays Dedicated to Michael Papazoglou on the Occasion of His 65th Birthday and His Retirement
2021

DOI (link to publisher)
10.1007/978-3-030-73203-5_3

Software Sustainability in the Age of Everything as a Service

Vasilios Andrikopoulos[1](✉) and Patricia Lago[2,3]

[1] University of Groningen, Groningen, The Netherlands
v.andrikopoulos@rug.nl

[2] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
p.lago@vu.nl

[3] Chalmers University of Technology, Gothenburg, Sweden

Abstract. The need for acknowledging and managing sustainability as an essential quality of software systems has been steadily increasing over the past few years, in part as a reaction to the implications of "software eating the world". Especially the widespread adoption of the Everything as a Service (*aaS) model of delivering software and (virtualized) hardware through cloud computing has put two sustainability dimensions upfront and center. On the one hand, services must be sustainable on a technical level by ensuring continuity of operations for both providers and consumers despite, or even better, while taking into account their evolution. On the other hand, the prosuming of services must also be financially sustainable for the involved stakeholders.

In this work, we discuss the need for a software architecting approach that encompasses in a holistic manner the other two dimensions of software sustainability as well, namely the social and environmental aspects of services. We highlight relevant works and identify key challenges still to be addressed in the context of software systems operating across different models for cloud delivery and deployment. We then present our vision for an architecting framework that allows system stakeholders to work in tandem towards improving a set of sustainability indicators specifically tailored for the *aaS model.

Keywords: Software sustainability · *aaS · Cloud computing · Software architecture · Vision

1 Introduction

The last decades have generated an increasing awareness of the need for a more sustainable world that is at the same time increasingly being run by software systems [13]. Sustainability can be defined under two lenses as discussed in [16]: that of *sustainable use* of a system with regard to a function over a time horizon, and that of *sustainable development* that meets current needs without compromising the ability of future generations to do the same. Computing in general, and software in particular has the potential to enable sustainability across the societal spectrum by reducing the impact of production and consumption through digitalization, presuming its own footprint is brought under control. In the recent years, sustainability has been acknowledged as an essential software quality across four dimensions [23]: a *technical* one related to the ability of a software system to evolve and remain used in the long term, an *economic* one concerning the preservation and creation of capital and value, a *social* dimension focusing on the continuity of communities using the system, and an *environmental* one aiming to minimize the impact of the system on natural resources. The ubiquity of software, with the world proverbially eaten up by software day by day[1], makes this concept even more important, and nowhere more than in the *Everything as a Service (*aaS)* model as enabled by the extended Service-Oriented Architecture (SOA) paradigm [29]. This is because in that model, service owners and providers have a responsibility to not only their own stakeholders, with respect to generating sufficient profits to justify the existence of services, but also to the consumers of their services so as not to disrupt their operations [28]. Online services act as the backbone of online communities supporting their communication (e.g. Slack and similar) and collaborative work (e.g. GitHub). There are, however, increasing concerns about the environmental footprint of the software industry, especially with respect to the energy consumption of large data centers such as those realizing the public cloud these services rely upon [15,18].

As a consequence, the state of the art on sustainable software engineering has been evolving significantly in the last couple of decades, starting with a focus on 'green' software, i.e. focusing on the environmental dimension, but expanding also to the other ones. A recent survey of sustainable software research based on the 5Ws formula (why, when, who, where, and what) [3], shows that this is a very active and collaborative area of research, with a good level of maturity. Despite the efforts from various research communities, however, there is still a lot of ground to be covered with respect to establishing and exploiting the relation between software architecture and sustainability as a software quality [30], and especially for software services. This gap is even wider when cloud computing as the de facto computing model through which services are delivered to their consumers is brought in the picture. In spite of early efforts in that area, e.g. [9], consequent works have focused almost exclusively on improving the efficiency of cloud data centers with respect to energy consumption and/or CO_2 emissions, i.e. on the environmental dimension of sustainability. Moreover, very little evidence exists on the true software-driven optimizations applied in the data center industry beyond, for instance, the adoption of renewable energy resources or smart techniques for cooling and hardware-level efficiencies [5,18]. This creates the urgent need for an approach encompassing all dimensions in architecting sustainable systems in the *aaS model.

To this aim, we present our vision for an architecting framework focusing on *software systems running either on public clouds or on private/public hybrid cloud deployments*, potentially but not necessarily as *multi- or federated cloud solutions*, offering their functionality *as services* to their users. We scope our

[1] As famously noted by Marc Andreessen in his 2011 Wall Street Journal interview.

proposal to such systems because they are becoming the standard model for delivering software to its users in practice. The intended audience for our proposal, and therefore the primary stakeholders to be involved in this framework, are mainly system and solution architects as the potential adopters of our proposed approach. Ultimately, however, all providers and consumers of services are in scope. The objective of this work is therefore *to provide architects with guidance in designing, maintaining and evaluating sustainable systems in the scope of interest across all dimensions of sustainability.*

2 Related Works

In the following, we discuss the state of the art on sustainability in the context of service orientation and cloud computing, since these two fields provide the backdrop for our proposal.

Service Oriented Computing

Early works on sustainability in SOA focused on energy awareness of service-based applications (SBAs) and energy efficiency of services and service providers, that is, on the environmental aspect. Mello Ferreira et al. [25] for example define an energy efficiency metric specifically for services, which aggregates energy consumption with execution time. This metric allows them to treat the problem of implementing SBAs as compositions of services as an optimization problem of the tradeoff between performance and energy consumption. Lago and Jansen [22], on the other hand, discuss the role of software services in making enterprises greener, i.e., more environmentally sustainable. The proposed approach is focused on the creation of awareness across three problem areas: processes, services, and people. A portfolio of green software services for enterprises accessible through the Web that define the environmental strategies to be implemented and the metrics to be used for their evaluation, is envisioned for this purpose. In [8], Dustdar et al. analyze the case for delivering software services that are sustainable across all aspects of the term. The key proposition of this work is the creation of a marketplace of such services which is founded on business models promoting the collaboration of stakeholders towards delivering such services.

Cloud Computing

One of the earliest and most comprehensive works on sustainability for cloud computing was produced by the OPTIMIS EU project. In that context, Ferrer et al. [9] identified *dependable sociability* as one of the major challenges of cloud computing, a concept which encompasses explicitly economic and environmental sustainability, but also incorporates strong elements of social and technical sustainability through its trust and risk aspects. However, this sustainability is to be enabled by means of a service lifecycle management toolkit which focuses on cloud service and infrastructure optimization. As such, architecting of software systems is effectively restricted to the models supported by the toolkit. A lock-in dependence on the toolkit is also inadvertently created by the reliance on it. A similar, CO_2-emission optimizing approach is offered by the ECO_2Clouds project [31] across federated cloud solutions, which also includes an application controller that is responsible for redeploying application components across providers in order to minimize carbon emissions.

Branching out from the 2012 work on Green Cloud Architecture by Garg and Buyya [11], a series of works have examined the environmental impact of cloud data centers and aimed to reduce their energy footprint. SMICloud by Garg et al. [12], for example, folds in the concepts developed in [11] for the ranking of cloud service providers. In the recent survey by Gill and Buyya [14] on sustainable cloud computing there are more than a dozen surveys on the subject from the last decade mentioned as related work. The primary focus of these secondary studies, and by definition also of the primary studies they survey, is on optimizing the energy consumption as the means for minimizing the financial and environmental footprint for operating as cloud service providers. Even when other dimensions of sustainability are discussed, as for example in [15], the foreseen benefits are all effectively derived from energy saving and reclaiming techniques on the provider side. However, as discussed by Hilty and Aebischer [16], focusing on energy efficiency in systems where energy costs are a major component, like data centers, creates a high risk of *rebound effect*, i.e. improved energy consumption resulting into more or larger data centers consuming more electricity as a net sum. The continuous growth in the number of cloud data centers belonging to the major cloud providers actually points to this rebound effect in practice during at least the last decade.

Pahl et al. [26] frame the continuous evolution and adaptation of cloud-based software systems in a context of technical and economic sustainability. The proposed approach is building on the use of self-adaptation control loops across cloud delivery models as the means for ensuring the continuity of operations. The biggest difference of that approach with the ones presented above is that it clearly focuses on software systems as consumers of cloud services, instead of taking the service provider perspective.

Outside of "sustainability in cloud computing", Domdouzis [6] discusses cloud computing as an *enabler of sustainability* across social, environmental and economic dimensions. Cloud computing is hailed as a significant factor for achieving sustainability by means of promoting academic research, facilitating business development, and strengthening business competition. A similar multi-dimensional perspective is shared by GeSI (Global enabling Sustainability Initiative): in a recent webinar on "The Cloud as an Enabler of Innovation and Sustainability"[2] (see also [13]), CEO Luis Neves argued that any company (incl. data center leadership) should be driven by sustainability to become more

[2] 6 Oct. 2020.

profitable, and that transparency, accountability, trustability are key drivers to remain on the market.

3 Challenges

Architecting sustainable systems in general is acknowledged to be one of the grand challenges in modern software engineering, as discussed by Venters et al. [30]. The need for a more comprehensive view which (i) integrates the various dimensions while taking into consideration their relations and potential trade-offs, and (ii) enables further investigation into the relations of other software qualities and metrics, is also discussed by Condori-Fernandez et al. [4] and Lago et al. [24]. Beyond these general issues with sustainability in software architecture, however, there are additional concerns that rise due to infrastructure, platforms, and software being offered and used as a service, both on and off premises. For starters, *the inherent diffusion of responsibility boundaries in prosuming relations creates a series of challenges*:

1. Acting as a provider puts the onus of responsibility on the service itself to be sustainable over time; however when services are consumed as part of the implementation of said service then there are obvious external dependencies that are in principle outside of the control of the service stakeholders.
2. There is in many cases a tension between the responsibility to the stakeholders/owner of the service in terms of sufficient revenue generation as the justification for the continued existence of the service, and the responsibilities to the wider society and the environment.

In order to deliver loose coupling [27], service orientation relies on the architecting principles of information hiding and encapsulation behind opaque interfaces. As a result, *services are intentionally offered as black boxes*, with little to no visibility of how they are implemented or operated. This inadvertently leads to a reliance on self-labeling (if at all) of the sustainability and other quality of service metrics by the service providers. It can be easily seen how this model can and has been abused by service providers, sometimes unintentionally so. Furthermore, while *private clouds are for all practical purposes white boxes* and can for example be instrumented to monitor their environmental impact in terms of energy consumption and heat generation, *public clouds are essentially black boxes*. If architecting of systems has to rely on information exposed by the provider, then there has to be a way to verify the provider claim in a controlled manner, even if this is under strict experimental conditions. An objective, standardized and auditable labeling mechanism instead is therefore required for this purpose. Architecting approaches for software systems that operate in this environment have therefore to acknowledge and fulfill the following requirements:

REQ1 The approach must concern itself with how all dimensions of sustainability (economic, technical, social, and environmental) are affected by the system under consideration.

REQ2 The approach must take into account that services are both used for the design, implementation, and operation of the system (i.e. consumed), and delivered to the users of the system (i.e. provided).

REQ3 The approach must acknowledge the fact there is limited to no visibility into the workings of consumed services.

4 A Sustainability-Aware Architecting Framework: Vision and Future Work

Now that (i) practitioners start feeling the urgency to address sustainability concerns in their business, and (ii) the business models entailing networks of multiple stakeholders gain traction, a key challenge is to define clear and quantifiable responsibilities and accountability toward (shared) sustainability goals. In practice, this requires the introduction of a reference architecture, and common standards and metrics. Such architecting tools will allow practitioners to analyze, design, evaluate, and maintain software systems in collaboration with the rest of the system stakeholders, both internal and external to their organization. However, as discussed in the previous sections, while we have some knowledge on how to architect systems for sustainability in general, we still have major challenges to overcome in developing cloud-based systems that rely on services on multiple levels and expose their functionality as services in a sustainable manner. As an evolutionary step towards this goal, we propose a sustainability-aware framework which allows existing and to-be developed architecting methods and techniques to interact with each other towards addressing these challenges. The underlying idea is to put first the focus on raising awareness on how architecting decisions affect system sustainability, and to reuse known best practices, before looking into further empirical evidence on what works best for improving it.

Grounded in SoSA [20], the proposed framework aims to help software architects to address the challenges described in the previous section, taking into account the need for operating in a hybrid (multi-)cloud deployment model. As summarized in Fig. 1, it consists of:

- a set of *stakeholders* with distinct roles and responsibilities, and
- a set of *sustainability indicators*, one per dimension, that we perceive as existing in a positive feedback loop relation with each other.

Different stakeholders are involved in different aspects of sustainability, with the exception of system architects that act as the natural hub of decision making and consider all dimensions. Not shown in the figure are the architecting decisions taken over time by the system architects in consultation with the other involved stakeholders. These are to be reflected and documented following an appropriate approach for this purpose as it will be discussed below. In the following we present the components of this framework.

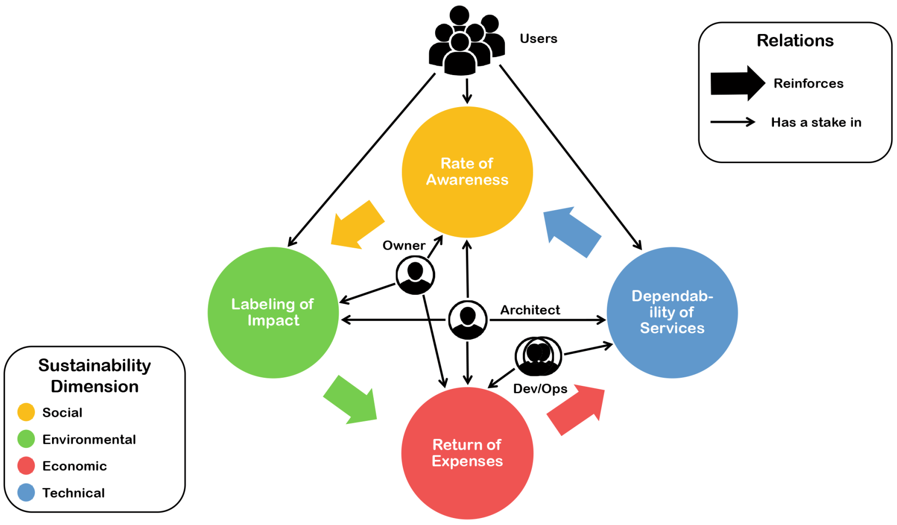

Fig. 1. A high-level view of the sustainability-aware architecting framework: stakeholders, sustainability dimensions and indicators, and their relations (Color figure online)

4.1 Stakeholders

The main stakeholder roles under consideration as summarized by Fig. 1 are:

Owner. The owners of the system to be architected, usually but not necessarily software-intensive enterprises; system owners are also expected to act as the service providers for the involved services.

Users. The sets of users relying on the functionalities delivered by the system; system users are acting as service consumers, and they might be in the same or different organization as the system Owner.

Dev/Ops. These are the developers, operators, and/or DevOps personnel responsible for the realization and running of the system services.

Architect. The software architect (or architects) involved in the requirement analysis, design and evaluation of the system.

Architects are to use architectural decision maps as per [21] to elucidate and communicate the effect of design decisions to the other stakeholders with respect to each dimension, balance concerns across the different dimensions, and exploit the relation of the different aspects as a positive feedback loop to be leveraged to deliver sustainability. As such, they take into consideration and are interested in improving all sustainability indicators discussed below. Owners also have potentially a stake in all indicators, however, the technical aspect of the system is actually relevant to them only if it has implications for the other dimensions, e.g. it results into a loss of revenue. For this purpose, this relation, from the Owner to the technical sustainability indicator is omitted from the figure. Similarly, Users are primarily interested in the continuity of the provided services and the impact that consuming them has on the environment, since it affects them indirectly, and as such they are not involved in decisions related to the economic sustainability of the system. Finally, the technical personnel is involved in principle only in two aspects: its economic one, in terms of e.g. service selection for hosting the deployed software, and of course its technical one. While it is tempting to have all stakeholders interested in all aspects of sustainability, and in some specific situations this might be indeed true, we feel that this mapping between aspects and responsibilities reflects better the stakeholder dynamics for the majority of systems of interest.

4.2 Indicators

Each of the four dimensions is associated in the framework with a *sustainability indicator* serving a dual function, in a manner similar to Hilty and Aebischer [16]: as a *metric* to provide insight into the current state and potential impact of architecting decisions to the stakeholders, and as a *goal* to optimize the architecture in order to reinforce the sustainability of the system. In the following, we discuss the associated indicator per dimension, and identify the research objectives and foreseen outcomes that we aim to address in the future. We start the discussion from the bottom of Fig. 1 and continue counter-clockwise.

Economic Dimension: The economic sustainability of a system in our framework is indicated by its *Return of Expenses* (RoE), that is, the degree of efficiency in consuming cloud resources (services like VM, storage, network etc.) with respect to the generated revenue by the system. Cloud computing was popularized on the basis of capital to operation expenses transfer [2]. RoE is meant to show how successful this transfer actually is, taking into account that despite the utility billing model of cloud services, it is fairly common for cloud-based software to generate *waste*. Waste here refers to the over–provisioning of computational resources within each billing cycle, e.g. VMs with low utilization or even idling for long periods relative to the billing unit. This waste can be generated for multiple and potentially overlapping reasons: over–estimation of the foreseen load for the provided services resulting into selection of over–sized configurations of cloud services during provisioning, absence of an appropriate scaling policy or sufficient guidance in the creation of such policy for the consumed cloud services, an inefficient assumption of underlying infrastructure as "always on" instead of on demand, typically as a result of a lift and shift migratory strategy (Type III migration in the taxonomy of [2]), and/or the inability of the scaling mechanisms put into place to compensate for quickly fluctuating load in an effective manner.

In any case, for software deployed on a public cloud provider the generated waste has a very real monetary impact on the monthly utilization bill which becomes obvious, in the literal sense of the word, to the system stakeholders. The annual practitioner surveys on the State of the Cloud conducted by Rightscale (currently under the Flexera banner) are, for example, showing a consistent waste of around 30% year after year. This is clearly an unacceptable level of resources misuse that needs to be addressed on a systemic level. What's more, addressing inefficiencies in the RoE creates also a positive effect on the technical dimension: the freed up financial resources due to lower expenses can be used instead for improving software quality, and especially "longevity" of the system throughout its evolution over time (red arrow in Fig. 1).

The main objective of the framework with respect to the financial sustainability of systems can therefore be summarized as:

> How to maximize *Return of Expenses* by minimizing the *wasting* of cloud resources in order to shift the *reclaimed revenue* towards improving the overall *software quality* of the system, and its *dependability* in particular?

Technical Dimension: The technical sustainability aspect of the framework is concerned with the particular characteristics of software quality that can benefit from the additional resources generated by its financial aspect, and more specifically, with the *Dependability of Services* that the system exposes to its users (Fig. 1). Dependability as a software quality refers to *the ability of services to deliver their functionality to their consumers in depth of time* [1]. It incorporates both operational attributes like availability, reliability, and scalability, and architectural ones like adaptability. The objective is therefore to allow system architects to develop the mechanisms required to answer the question:

> How to ensure the *continuous evolution* of a system within the confines prescribed by the desired *quality of service* levels to be offered, and the need to improve *the overall quality* of the system and specifically the *Dependability of its Services*?

In principle, the need for continuous evolution of software systems is dictated by the need to address changes across their operating environment: both internal (shifting of priorities and product focus) and external (technological and market changes). The dual nature of services that act as both providers and consumers creates additional complications that need to be taken into consideration:

1. When acting as *providers* of services, systems need to evolve their exposed APIs/service endpoints in a manner which preserves their compatibility with their clients in terms of both functional and non-functional expectations [28].
2. When acting as service *consumers*, and especially of cloud services, the overall quality of service offered by software systems very much depends on the quality of consumed services [2].

Social Dimension: Achieving higher dependability through continuous and controlled evolution empowered by more efficient use of financial resources leads naturally to an increased life expectancy of the offered services, and potentially a much heavier dependence on these services by their consumers (i.e. the Users in Fig. 1). In turn, this leads to the need for more awareness of the sustainability of the system across stakeholders in order to enable the communities of consumers in the long term. A proactive and constructive way of managing this need is by allowing sustainability-affecting decisions to become directly visible to the relevant stakeholders (owners and users). Achieving this effect in the most direct manner entails involving them actively in the decision making loop, or at the very least communicating to them the foreseen impact of e.g. changes to the architecture of the system. By these means, the level of awareness of the various stakeholders in decision making can be used as an indicator of social sustainability:

> How to increase the *Rate of Awareness* across the involved system stakeholders as the means for engaging them in the effort to increase sustainability?

Existing approaches such as [7] have already identified methods for involving stakeholders during the analysis phase of architecting. However, our proposal is to expand their participation throughout the architecting life cycle.

Environmental Dimension: The longer life expectancy of offered services and prolonged consumption by engaged user groups creates inadvertently a larger environmental footprint. For the purposes of the framework, this footprint is primarily defined in terms of the energy consumed for operating and cooling the servers hosting the services, or the CO_2 emitted in the atmosphere as a by-product of converting fossil fuels into energy for this purpose. However, we know that eventually we will have to expand our work to also include second and third order effects to the environment through e.g. the consumption of raw material to build the equipment for the data centers hosting the servers, and the impact of the proliferation of services to the lifestyle of their users. Going back to the first order effects for now, being able to assess and label the system footprint in an objective and transparent manner is crucial for diagnosing inefficiencies in it, resulting to the following objective:

> How to assign the appropriate *Labeling of Impact* to the environment created through the continuous operation of the offered services?

Ideally, the impact label of an offered service would be something as easy to communicate as the energy ranking of commercial devices, allowing for easy comparison between them by the users, and resulting in raising awareness across the board [10]. However, achieving this goal is far from trivial. First of all, the energy consumption of even simple systems is sensitive to small architecting changes, and extensive benchmarking and monitoring might be required in order to establish the actual footprint of the system [19]. Second, there is still a lack of appropriate methodological aspect and shared benchmarking data sets and tools for this purpose, as highlighted in [17]. Finally, as we discussed in the previous section, while the services owned by an organization can be the subject of both experimental measurement during development and monitoring in production, services consumed e.g. leased virtual machines in the cloud, are effectively opaque to such procedures. Using the published specifications of such services to establish an approximate model of their energy consumption, for example, could be an interesting approach in bypassing this opaqueness. However, further work is required in this direction.

Closing now the feedback loop of the indicators back to the economic dimension, it needs to be pointed out that better environmental labels, both in sense of their accuracy, and in terms of indicating low impact to the environment by the labeled system, have a potential positive effect on the economic dimension (green arrow in Fig. 1). The reasoning behind this is fairly straightforward: energy costs money; better, more reliable labels mean eventually less expenses through better service selection (from the perspective of the users) and less costs for operating data centers (for the system owner, provided they do not fall in the trap of the rebound effect [16]). It is therefore our proposition that it is possible, at least on conceptual level, to use these indicators to create a self-sustaining system which delivers sustainability in turn to its stakeholders across all dimensions. Realizing and assessing the efficacy of this vision lays ahead of us.

5 Conclusions

Prior works have already highlighted the relation between software architecture and sustainable development. When considering, however, software systems consuming cloud services while being offered as services on their own, the majority of existing approaches focuses primarily on the environmental aspect of the systems. As a reaction to this, we proposed our vision for an architecting framework defined along the lines of involved stakeholders and indicators that are used to govern the architecting process. Such indicators are to be used as the means for both improving the architecture of a system in terms of sustainability, and for communicating sustainability-affecting decisions to stakeholders. If implemented correctly, they allow architects to create a positive feedback loop which delivers sustainability across multiple dimensions. Putting this framework into practice, however, requires addressing a number of complex research questions in our immediate future.

References

1. Quality reference model for SBA. Technical report, S-Cube consortium (2008). https://s-cube-network.eu/results/deliverables/wp-jra-1.3/Reference_Model_for_SBA.pdf
2. Andrikopoulos, V., Binz, T., Leymann, F., Strauch, S.: How to adapt applications for the cloud environment. Computing **95**(6), 493–535 (2013). https://doi.org/10.1007/s00607-012-0248-2
3. Calero, C., et al.: 5Ws of green and sustainable software. Tsinghua Sci. Technol. **25**(3), 401–414 (2020)
4. Condori-Fernandez, N., Lago, P., Luaces, M.R., Places, Á.S.: An action research for improving the sustainability assessment framework instruments. Sustain. Sci. Pract. Policy **12**(4), 1682 (2020)
5. DDCA: State of the Dutch Data Centers 2020: go digital, act sustainable. Technical report, Dutch Data Center Association, June 2020
6. Domdouzis, K.: Sustainable cloud computing. In: Green Information Technology: A Sustainable Approach, pp. 95–110. Elsevier Inc., March 2015

7. Duboc, L., et al.: Do we really know what we are building? Raising awareness of potential sustainability effects of software systems in requirements engineering. In: International Conference on Requirements Engineering, pp. 6–16. IEEE, September 2019
8. Dustdar, S., et al.: Green software services: from requirements to business models. In: Proceedings of the 2nd International Workshop on Green and Sustainable Software (GREENS 2013), pp. 1–7. IEEE Computer Society (2013)
9. Ferrer, A.J., et al.: OPTIMIS: a holistic approach to cloud service provisioning. Future Gener. Comput. Syst. **28**(1), 66–77 (2012)
10. Fonseca, A., Kazman, R., Lago, P.: A manifesto for energy-aware software. IEEE Softw. **36**(6), 79–82 (2019)
11. Garg, S.K., Buyya, R.: Green cloud computing and environmental sustainability. In: Harnessing Green IT, pp. 315–339. Wiley, September 2012
12. Garg, S.K., Versteeg, S., Buyya, R.: A framework for ranking of cloud computing services. Future Gener. Comput. Syst. **29**(4), 1012–1023 (2013)
13. GeSI: digital with purpose: delivering a SMARTer 2030. Technical report, GeSI (2019)
14. Gill, S.S., Buyya, R.: A taxonomy and future directions for sustainable cloud computing: 360 degree view. ACM Comput. Surv. **51**(5), 1–33 (2018)
15. Gill, S.S., et al.: Holistic resource management for sustainable and reliable cloud computing: an innovative solution to global challenge. J. Syst. Softw. **155**, 104–129 (2019)
16. Hilty, L.M., Aebischer, B.: ICT for sustainability: an emerging research field. In: Hilty, L.M., Aebischer, B. (eds.) ICT Innovations for Sustainability. AISC, vol. 310, pp. 3–36. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-09228-7_1
17. Hindle, A.: Green software engineering: the curse of methodology. In: International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 46–55. IEEE, May 2016
18. Jones, N.: How to stop data centres from gobbling up the world's electricity. Nature **561**(7722), 163–166 (2018)
19. Kazman, R., Haziyev, S., Yakuba, A., Tamburri, D.A.: Managing energy consumption as an architectural quality attribute. IEEE Softw. **35**(5), 102–107 (2018)
20. Lago, P.: SoSA: A Software Sustainability Assessment Method. European Computer Science Summit, October 2016. https://goo.gl/HuY6tf
21. Lago, P.: Architecture design decision maps for software sustainability. In: Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS 2019, pp. 61–64. IEEE, May 2019
22. Lago, P., Jansen, T.: Creating environmental awareness in service oriented software engineering. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 181–186. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19394-1_19
23. Lago, P., Koçak, S.A., Crnkovic, I., Penzenstadler, B.: Framing sustainability as a property of software quality. Commun. ACM **58**(10), 70–78 (2015)
24. Lago, P., et al.: Designing for sustainability: lessons learned from four industrial projects. In: Kamilaris, A., Wohlgemuth, V., Karatzas, K., Athanasiadis, I.N. (eds.) Advances and New Trends in Environmental Informatics. PI, pp. 3–18. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-61969-5_1

25. Mello Ferreira, A., Kritikos, K., Pernici, B.: Energy-aware design of service-based applications. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC/ServiceWave - 2009. LNCS, vol. 5900, pp. 99–114. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10383-4_7
26. Pahl, C., Jamshidi, P., Weyns, D.: Cloud architecture continuity: change models and change rules for sustainable cloud software architectures. J. Softw. Evol. Process **29**(2), e1849 (2017)
27. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson Education, Harlow (2008)
28. Papazoglou, M.P., Andrikopoulos, V., Benbernou, S.: Managing evolving services. IEEE Softw. **28**(3), 49–55 (2011)
29. Papazoglou, M.P., Van Den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. VLDB J. **16**(3), 389–415 (2007). https://doi.org/10.1007/s00778-007-0044-3
30. Venters, C.C., et al.: Software sustainability: research and practice from a software architecture viewpoint. J. Syst. Softw. **138**, 174–188 (2018)
31. Wajid, U., et al.: On achieving energy efficiency and reducing CO_2 footprint in cloud computing. IEEE Trans. Cloud Comput. **4**(2), 138–151 (2016)