




Sustainable IT in an Agile DevOps Setup Leads to a Shift Left in Sustainability Engineering

Alexander Poth^(✉) , Daniela Eißfeldt, Christian Heimann, and Stefan Waschk

Volkswagen AG, Berliner Ring 2, 38436 Wolfsburg, Germany
{alexander.poth,daniela.eissfeldt,christian.heimann,
stefan.waschk}@volkswagen.de

Abstract. Today green IT is mostly driven by the measurement of CO₂e of data centers. However, this is a symptom treatment approach, since the operating parameters of software are defined during build-time. This implies that the consumption during run-time of a software cannot be changed in a wide range. To ensure that enterprise IT can be operated within a higher sustainable setup the software and systems engineering has to consider sustainability aspects during development phase. Furthermore, sustainability is more than measuring and optimizing CO₂e of applications – it includes e.g. reuse aspects. Each software component which is reused reduces resource allocation during development and maintenance. IT sustainability step by step becomes a quality characteristic of software. This work presents a more holistic view for sustainable software engineering from an enterprise IT perspective which can be integrated into agile software development especially within DevOps teams.

Keywords: sustainable software engineering · green coding · agile transformation · green IT · ISO 14001 · DevOps

1 Motivation, Context and Methodology

Many enterprises in different countries and regions are working hard to become more sustainable like [1, 2, 3]. A typical starting point is to measure and optimize respective CO₂e emissions. To ensure professional management of the optimization standards like the ISO 14001 are used [4]. However, many companies are focused on their production or operations while measuring and improving their sustainability footprint. With a focus on enterprise IT the situation becomes more complex as to use an operation service driven approach, because the parameters for IT systems and software operations are defined in an earlier life-cycle stage. The important life-cycle stage to ensure a sustainable software of IT systems is defined during design and development. To increase the effectiveness of the sustainability actions and efforts an interdisciplinary agile team – with skills in design, development and operation - can facilitate a shift left of sustainability topics into the build phase of enterprise IT software. This shift left approach has to be aligned with the enterprise environmental and sustainability strategy and adapted to the working level procedures. For agile teams this implies to aware of sustainable software engineering

with the corresponding methods and tools fostering the integration of sustainable engineering into their value stream workflows. The approach has to consider aspects from the Greensoft model [5], impacts of SLAs [6] for IT based/supported services [7] and the “bin packing” problem [8] to ensure that resources are adequate allocated for a high utilization.

The research questions around this enterprise IT sustainability setup are:

RQ1: What are the main dimensions for a holistic sustainable software engineering?

RQ2: What is needed for agile teams to perform sustainable software engineering?

Section 2 elaborates the sustainability model for enterprise IT. Section 3 presents the use case for agile DevOps team and Sect. 4 gives an example form the Volkswagen Group IT. Finally, Sect. 5 concludes by summarizing the article’s key contributions to research and practice and giving an outlook to the authors’ ongoing research activities.

2 Methodology and Outcome Design

An Action Research (AR) [9] approach is used to ensure that the outputs are usable by the value stream teams and the outcomes fit into real enterprise IT setups. To answer RQ1 the sustainability model is derived and refined as followed: To establish a shift left in sustainability engineering a life-cycle approach is developed. Furthermore, a consistent refinement from the overall sustainability objective to the product or service specific actions has to be established. To address this two dimensions the matrix of Fig. 1 was developed. Each dimension is based on factors. One dimension (the y-axis in the figure) represents the organizational abstraction level, the other dimension represents the life-cycle phases (the x-axis in the figure) of the software.

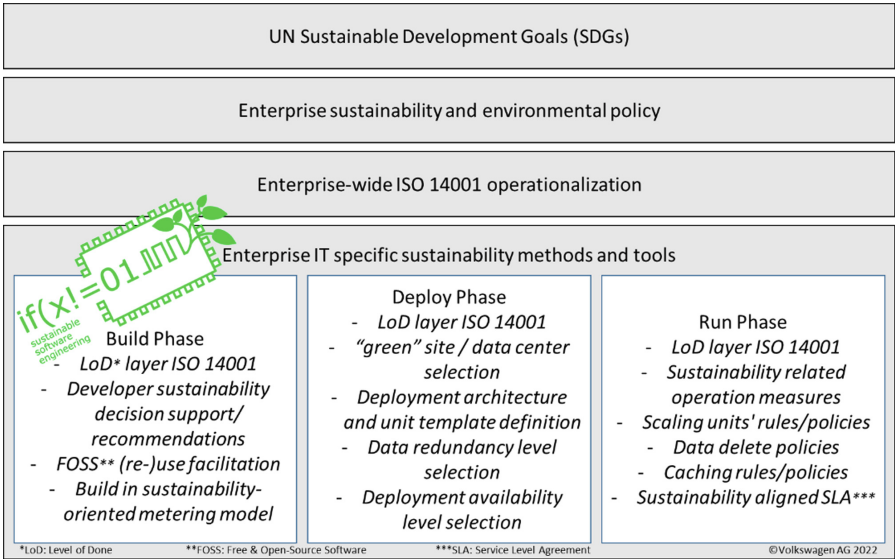


Fig. 1. Sustainability model for enterprise IT software and systems.

The figure has on its y-axis four levels respectively factors.

The *UN SDG* [10] – UN Sustainable Development Goals are the most generic sustainability objectives. These are a good starting point for all sustainability initiatives.

The *enterprise sustainability and environmental policy* selects from the UN SDG or can be mapped to them. The policies set focus and foster transparency in the effects of activities.

The *enterprise wide ISO 14001 operationalization* is to refine respective instantiate and operate the enterprise policy. The management system establishes an efficient organization of environmental improvement. Important is that this is the base for a deduction and refinement to specific business domains and value streams. It sets the boundaries in which strategies are acted. Typical strategies are efficiency, consistency and sufficiency in the context of sustainability [11].

For the *enterprise IT specific sustainability methods and tools* an IT domain specific set of methods and tools for implementation is used. The set is used to facilitate a wide adoption by agile value stream teams. This methods address sustainability related service pricing, effective reuse e.g. based on FOSS (Free and Open Source Software) and systematic sustainable software engineering recommendations.

The x-axis is structured into three core phases. Within the phases are specific factors which have to be considered by the engineers. The core phases are applied to releases of the software.

The *build phase* defines all the parameters for the following phases. Examples include the modularization of components which are workload-sensitive to be able to scale them elastic to the current workload. Implement algorithms which are resource efficient for the workload. Offer parameterization of the software to address specific data life-time or redundancy of the data and software components (availability).

The *deployment phase* determinates the operation environment and its “green-factor” for the following phase. Here it is important to select the most energy efficient environment available parameters. Two advices are to select the data center with a highest Power Usage Effectiveness (PUE) [12] available, and to select the most energy efficient architecture platform like ARM over AMD over Intel CPUs. This selection is motivated by Koomey’s Law [13] which indicates a continuous efficiency optimization with each CPU generation. However, here the options can be limited by the delivered software components which are for e.g. build for x86 or enterprise policies which includes a commitment to x86-architecture. Furthermore, to select adequate data-lifetime policies and redundancy strategies is recommended.

The *run phase* optimize the resource consumption within the determined environment and pre-defined parameters. Examples are to select the newest generation of instances for the most power efficient execution, use optimized instance types for the workloads, adjust scaling policies optimize caching and routing for the specific software system. Furthermore, establishing sustainability related measure as base for further optimization is advised.

A decommission phase is not defined as a core phase, because with a good architecture combined with continuous refactoring keeps the software “young”. However, every

software has its end of life which also addresses the switch or migration to another software which it is not in scope in our presented view from the sustainability perspective. Related data has to be handled with an objective to delete or reuse it in another IT system.

Furthermore, a plan phase is not defined, because as IT sustainability is a IT delivery characteristic it is not a topic to elaborate it with the business like the Product Owner. This is similar to IT Security – it is driven by the IT architecture and for its implementation the business is asked for some decisions were needed, but is made and driven by IT experts. The business has to ensure that only valuable and relevant functionality and capabilities are built by the IT as a core contribution of sustainability form a business perspective. The entire IT deliverables have to be seen as a part of the business sustainability concept.

The sustainability model with its dimensions is open for additional enterprise respective product/service domain specific factors. For example in a finance domain established mainframe system based infrastructure FOSS reuse is a limited applicable factor – or reduced to an inner-source mindset. Figure 1 shows a basic set of factors which are generic for many enterprise IT setups. The figure is inspired by the Plan-Build-Run approach [14]. Between Build and Run a Deployment is added to make transparent that here some important setting are made which can have impact to the sustainability of the service delivery. The Build phase includes solution architecture and design. The Plan phase is not focused as it does not address technical IT respective implementation aspects.

To establish all three phases of ISO 14001 teams need to be organized in an agile way like with the LoD layer approach [15]. Fostering common strategic aspects such as metrics or reuse – all which is needed to work over the three phases “smooth” is important as well.

The described sustainability model shows that IT and software sustainability is an additional quality characteristic of deliverables which have to be developed and established in enterprise IT organizations. The sustainability model uses efficiency strategy building blocks like algorithm optimization, consistency strategy building blocks like reuse via FOSS and it uses the sufficiency strategy building blocks like adequate scaling units.

3 Leveraging Sustainability with the Sustainability Model

To address RQ2 depending on the organization, different implementation scenarios are possible. The most flexible setup is found in DevOps value streams. In this cases the initial starting point can be a top-down driven by a values stream team or bottom-up driven by management approach. Furthermore, the run phase can be used as starting point to make a quick-win to optimize within the current predefined parameters footprint driven optimization back to the build phase. Also a build phase initiated sustainability engineering is possible for strategic actions with large levers.

In organizations without DevOps teams an ops driven bottom-up approach is potentially limited by the “silo” borders between ops and dev. A dev driven bottom-up approach has more success, because dev propagates the sustainability options to the later phase within the by design delivery flow. Also a top-down approach which addresses dev and ops has a higher success probability. The higher success is given by the option

to define common sustainability measures as base for e.g. common Objectives & Key Results (OKR) for the dev and ops team in a delivery stream. This ensures that both teams cooperate to achieve the same sustainability objectives.

However, in every point in the proposed sustainability model for enterprise IT is a possible starting point for a sustainability initiative, at least it can optimize the sustainability from a local perspective.

These analysis recommend a rollout scenario depending on the current organizational setup:

- Agile DevOps team: the sustainability initiative can initiate “everywhere”
- Independent agile teams for dev and ops: avoid to start in the ops team, because the silo boarder can limit effective rollout in direction dev team.
- Top-down is always possible by establish alignment of the teams with the defined objectives and goals

The organizational setup of the value stream teams has an influence how effective the sustainability efforts show effects and their impact on the service delivery. With the organizational setup like DevOps teams or management actions like OKR the base for a shift-left of sustainability actions is supported to act sustainable by design were possible.

4 Instantiation and Evaluation

Everything starts with the freedom to act for change. Within the Volkswagen Group IT triggers are established to act on the topic sustainability such as the “1-h project” [16] and initiatives like go2Zero [17]. These triggers encourage teams to invest into their sustainability capabilities to deliver more environmental friendly services to their users. A representative example for an enterprise IT setup is the Test-Runtime execution (T-Rex) cloud testing service DevOps team. The DevOps team of the service applies agile and lean principles and working methods. The team is formed by engineers with a T-shaped skill profile to ensure that all relevant aspects of a cloud-native service are handled like architecture, quality and testing within the software development. Furthermore, the team is not static by design because it is also a training on the job place for vocal education – every 6 months at least a team members either joins or leaves the team. Additionally, the team is composed of internal developers and contractors. The team started its sustainability journey over 2 years ago. Initially it began with explicit actions to optimize the infrastructure footprint during development – a quick win action. A parallel action in direction shift left was to optimize the effects of the software usage. This also includes the evaluation of alternative architecture approaches like serverless [18] and the insight to establish resource allocation related service models – here the shift left journey starts. To professionalize the sustainability actions an alignment with the ISO 14001 followed soon. As the team is using the method kit based efiS® framework [19] with the LoD layer [20] ISO 14001 which was an reusable output from the instantiation. This output is a first component which can be reused by other teams within the Volkswagen Group IT. The deduction of the enterprise environmental policy within IT was an additional outcome and described in [15]. This serves as a blueprint for other teams, too. The refinement work leads to the insight that a wider scaling within an enterprise IT more than the

LoD layer ISO 14001 is needed. Therefore the development of the cheat-sheet for sustainable software engineering was triggered [21] and it will be distributed to interested teams. The cheat-sheet consolidates knowledge about sustainable software engineering for an easy application within the Volkswagen Group IT. Parallel the systematic evaluation of FOSS components was initiated to professionalize software reuse. The gap was that established FOSS evaluations are a manual activity which does not scale for an extensive reuse – the objective was a (semi-)automation of the evaluation. This leads to the development of the Open Source Quality Radar (OSQR) [22] for facilitation of the DevOps team. By design OSQR was developed as a service which can be shared within the Group IT. The FOSS component focused approach addresses that optimized algorithms in libraries and frameworks are mostly by design more efficient than a self-developed algorithm – think, e.g., about compression, cryptography which is a non-trivial domain and should be handled in a professional and efficient way. Furthermore, it reduces the allocation of engineering resources for the topic by reusing existing and proven in use components.

5 Conclusion and Outlook

This work shows how enterprise IT agile value stream teams can evolve their sustainability engineering capabilities. It is important that the teams have the freedom to develop their sustainability skills. This has to be ensured by the organizations culture and habits to foster team autonomy and degrees of freedom. The examples show that sustainable software engineering can be developed and shared by doing respective operational delivery. An insight is that not always an explicit and expensive project for method development is needed. Important is that the team culture includes a higher agile mindset with established collaboration and sharing principles. A limitation within the shift-left approach is a separation of dev and ops. Because, as the example shows, the shift-left as the lever for high impact of small actions was only possible within the DevOps team.

The key contributions *to practice* can be summarized by the following aspects:

- With DevOps a team setup to establish a holistic sustainability approach for their value stream respective product or service is given
- Organizations can initialize sustainability software engineering by offering adequate degrees of freedom to the DevOps team
- The main effects in sustainability come with a shift-left from run to build.
- In non-DevOps organizations an initial initiation in the ops-team can be limited by the silo border to the dev-team.

The key contributions *to theory* can be summarized by the following aspects:

- Identification that sustainability can become a quality characteristic of software
- Identification that a holistic shift-left approach is needed for sustainability impact
- Identification that the organizational refinement aspects and the software life-cycle have to be thought together.
- Identification that different agile organizational structures like dev-teams or DevOps-teams require different approaches for effective sustainability engineering

As summary about sustainable IT artifacts should be a shared responsibility between IT and workload owner: *IT responsibility is sustainable service delivery* and *user responsibility is sustainable service consumption*. This includes that the IT cares about a sustainable software development and operation and the user respective workload owner cares about a sustainable usage of the IT. A precondition respective assumptions is that the IT artifact itself is a valuable artifact within an overall sustainable business context.

Also keep in mind that currently often *sustainability is measured outside* – e.g. by an (external) audit - in the operating phase but mostly *sustainability is decided inside* IT during design and implementation in an earlier development phase. A potential measured derivations to the expectations cannot be “healed” at or after the point of the late measurement without costly and resource intensive refactoring or re-implementation of the software. Therefore, *software sustainability is a build-in quality characteristic*.

An investigation aspect for the future is how to facilitate holistic improvements of sustainability with separated dev and ops teams by establishing collaborative goals over the different teams. The Volkswagen Group IT still starts different new initiatives which foster green IT and sustainability. Based on these triggers further building blocks for sustainable software engineering will be developed in the near future. However, there is still a lot of work to do to develop skills and capabilities for sustainable software engineering with the proposed shift-left mindset in all the value stream teams.

References

1. Lozano, R.: Towards better embedding sustainability into companies' systems: an analysis of voluntary corporate initiatives. *J. Clean. Prod.* **25**, 14–26 (2012)
2. Jose, P.D., Saraf, S.: Corporate sustainability initiatives reporting: a study of India's most valuable companies. *Corporations Sustain.*, 49–88. Routledge (2017)
3. Ismail, N.B., Alcouffe, S., Galy, N., Ceulemans, K.: The impact of international sustainability initiatives on Life cycle assessment voluntary disclosures: the case of France's CAC40 listed companies. *J. Clean. Prod.* **282**, 124456 (2021)
4. ISO 14001:2015 Environmental management systems — Requirements with guidance for use
5. Naumann, S., Dick, M., Kern, E., Johann, T.: The greensoft model: a reference model for green and sustainable software and its engineering. *Sustain. Comput.: Inform. Syst.* **1**(4), 294–304 (2011)
6. Barroero, T., Motta, G., Durante, M.: Sustainable service level agreements. *IEEE SCC*, 679–684 (2011)
7. Macías, M., Guitart, J.: SLA negotiation and enforcement policies for revenue maximization and client classification in cloud providers. *Future Gener. Comput. Syst.* **41**, 19–31 (2014)
8. Sengupta, J., Singh, P., Suri, P.K.: Energy aware next fit allocation approach for placement of VMs in cloud computing environment. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) *FICC 2020*. AISC, vol. 1130, pp. 436–453. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39442-4_33
9. MacDonald, C.: Understanding participatory action research: a qualitative research methodology option. *Can. J. Action Res.* **13**(2), 34–50 (2012)
10. UN SDG: <https://sdgs.un.org/goals>. Access validated 18 Jul 2022
11. Behrendt, S., Göll, E., Korte, F.: Effizienz, Konsistenz, Suffizienz: strategieranalytische Betrachtung für eine Green Economy. Institut für Zukunftsstudien und technologiebewertung,

- IZT-Text 1–2018, ISBN: 978–3–941374–35–5 (2018). https://www.izt.de/fileadmin/publikationen/IZT_Text_1-2018_EKS.pdf. Access validated 08 Jun 2022
12. Belady, C., Rawson, A., Pflueger, J., Cader, T.: Green grid data center power efficiency metrics: PUE and DCIE. The green grid, 1–9 (2008)
 13. Koomey, J., Berard, S., Sanchez, M., Wong, H.: Implications of historical trends in the electrical efficiency of computing. *IEEE Ann. Hist. Comput.* **33**(3), 46–54 (2010)
 14. Plan-Build-Run approach: <https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/using-a-plan-build-run-organizational-model-to-drive-it-infrastructure-objectives>. Access validated 18 Jul 2022
 15. Poth, A., Nunweiler, E.: Develop sustainable software with a lean ISO 14001 setup facilitated by the efiS® Framework. In: Przybyłek, A., Jarzębowicz, A., Luković, I., Ng, Y.Y. (eds.) *LASD 2022. LNBIP*, vol. 438, pp. 96–115. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94238-0_6
 16. 1-hour project: <https://www.volkswagenag.com/en/sustainability/strategy-policy-engagement/engagement/project-one-hour.html>. Access validated 08 04 2022
 17. Go2zero: <https://www.volkswagenag.com/en/news/stories/2019/07/co2-getting-to-zero.html>. Access validated 18 Jul 2022
 18. Poth, A., Schubert, N., Riel, A.: Sustainability efficiency challenges of modern it architectures – a quality model for serverless energy footprint. In: Yilmaz, M., Niemann, J., Clarke, P., Messnarz, R. (eds.) *Systems, Software and Services Process Improvement: 27th European Conference, EuroSPI 2020, Düsseldorf, Germany, September 9–11, 2020, Proceedings*, pp. 289–301. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-56441-4_21
 19. Poth, A., Kottke, M., Heimann, C., Riel, A.: The EFIS framework for leveraging agile organizations within large enterprises. In: Gregory, P., Kruchten, P. (eds.) *XP 2021. LNBIP*, vol. 426, pp. 42–51. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88583-0_5
 20. Poth, A., Kottke, M., Middelhave, K., Mahr, T., Riel, A.: Lean integration of IT security and data privacy governance aspects into product development in agile organizations. *J. Univ. Comput. Sci. (JUCS)* **27**(8), 868–893 (2021)
 21. Poth, A., Widock, A., Henschel, A., Eissfeldt, D.: Foster sustainable software engineering awareness in large enterprises – from a cheat-sheet for technical and organizational indicators to dashboards, euroSPI’22, Springer, in publication process (2022) https://doi.org/10.1007/978-3-031-15559-8_5
 22. Poth, A., Levin, D-A., Rjfolli, O., Wanjetscheck, M.: Quality evaluation with the open-source quality-radar for a sustainable selection and use of FOSS components, euroSPI’22, Springer, in publication process (2022). https://doi.org/10.1007/978-3-031-15559-8_36

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter’s Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter’s Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

