# SusDevOps: Promoting Sustainability to a First Principle in Software Engineering

Author: **Istvan David** (McMaster University, Canada)

## ABSTRACT

Sustainability is becoming a key property of modern software systems. While there is a substantial and growing body of knowledge on engineering sustainable software, end-to-end frameworks that situate sustainability-related activities within the software delivery lifecycle are missing. In this article, we propose the SusDevOps framework that promotes sustainability to a first principle within a DevOps context. We demonstrate the lifecycle phases and techniques of SusDevOps through the case of a software development startup company.

*Actionable insights software practitioners will get from this article:*

- *Understanding how and when sustainability practice interfaces with software development and operation is key to the efficient delivery of sustainable software. The SusDevOps framework situates these three practices with respect to each other.*
- *Feedback from operation should be processed through models and KPIs of sustainability, from which requirements are derived to implement the right trade-off between technical and sustainability goals. The SusDevOps lifecycle model aligns these concerns with the traditional DevOps process.*
- *While we use a simplified case to demonstrate the utility of the SusDevOps framework, it can accommodate a wide range of sustainability-related goals and KPIs.*

## 1. Introduction

Estimates show that the Information and Communications Technology (ICT) sector currently contributes to about 2–4% of global CO2 emissions and this number is projected to increase to about 14% by 2040 [1]. To follow suit with the rest of the global economy, the ICT sector should—directly or indirectly—decrease its CO2 emissions by 42% by 2030, 72% by 2040, and 91% by 2050.[1]

These numbers must concern software practitioners for a number of reasons.

First, the nature of user requirements is changing. While sustainability-related user requirements are not quite mainstream currently, sustainability is shaping up to become *the* non-functional requirement of the 21st century [2]. The expectation that users and organizations will not only reward but demand efforts toward sustainability has been identified as the top “global megatrend” by the International Council on Systems Engineering (INCOSE) recently[2],

[1] https://www.itu.int/en/mediacentre/Pages/PR04-2020-ICT-industry-to-reduce-greenhouse-gas-emissions-by-45-percent-by-2030.aspx

[2] https://www.incose.org/publications/se-vision-2035

necessitating a radically new approach to the engineering of software and software-intensive systems.
Second, even if a software company embraces the idea of developing sustainable software, the lack of software delivery frameworks that can accommodate sustainability goals quickly becomes a show-stopper. It is not easy to relate software functionality to sustainability goals along a software development lifecycle. When should sustainability requirements be addressed? How should they be prioritized? What techniques and tools can be used in support of systematic decision-making?

There is a substantial and rapidly growing body of knowledge on engineering sustainable software, with high-quality and actionable methods and techniques [3, 4]. This body of knowledge is ready to be put to use. To achieve this, we need end-to-end frameworks that promote sustainability to a first principle, instead of treating it as a quality metric.

In this article, we propose such a framework, called SusDevOps. As the name suggests, SusDevOps builds on the established software development and operations practices of DevOps [5] and aligns sustainability practices ("Sus") with them. The framework defines a holistic software development lifecycle model to weave these concerns into a coherent unit.

## Sidebar – DEVOPS

DevOps [5] is the collection of values, principles, practices, and tools that narrows the gap between developing ("Dev") and operating ("Ops") software systems. By promoting rapid iterative and incremental practices, DevOps increases the efficiency of delivery. While terminology is not standardized, the main stages of DevOps are well-understood by practitioners. These stages are the following.
PLAN. This stage includes the definition of process metrics for management and the definition of technical requirements for the software.
CREATE (or CODE). Development of the software system.
VERIFY (or TEST). Quality assurance, testing, validation, and verification.
PACKAGE. Package configuration, release staging, and release approvals.
RELEASE. Moving the software into production, including coordination, fallbacks, and recovery.
CONFIGURE. Preparation and provisioning of the infrastructure that hosts the software service.
MONITOR. Keeping track of functional and non-functional metrics, such as performance, response time, resource utilization, and end-user experience.
Feedback from monitoring is considered in the next planning phase of the next deliverable.

The original principles of DevOps are nowadays widely adopted, and different augmentations have been developed to extend the scope of DevOps. Such augmented approaches include DevSecOps [6], which emphasizes security ("Sec") in the software delivery process; BizDevOps [7], which focuses on connecting business operations ("Biz") to software delivery; and most recently, MLOps [8], that applies DevOps principles to machine learning ("ML").

We aim to provide useful information to at least three groups of practitioners. Software engineers who wish to anticipate how sustainability skills will disrupt their current practice so that they can anticipate the skills required for modern software engineering. Architects and product owners, and other principal designers and owners of delivery processes, so that they can understand how to extend delivery processes to incorporate activities that support sustainability ambitions. And decision-makers who identified the need for sustainability and are looking to revamp their operations for sustainability-first software practices.

### Sidebar – SOFTWARE SUSTAINABILITY

Sustainability is usually characterized by three complementary dimensions, first defined in the Brundtland report.[3] Economic sustainability is concerned with the financial viability of a software product; environmental sustainability focuses on reduced ecological impact, such as energy consumption; and social sustainability promotes the elevated utility of software for humans. Penzenstadler and Femmer [9] define a fourth dimension, technical sustainability, focusing on the prolonged service time of software systems, chiefly supported by proper evolution methods. All of these four dimensions must be considered to achieve sustainability in software systems.
Software engineers and practitioners are typically focused on *technical* sustainability, i.e., evolvability and maintainability, as these concerns are closest to software code and decisions can be made in the scope of a software engineer's scope of authority [10]. However, there is an emerging awareness of *environmental sustainability*, such as the energy consumption of software, as resource-intensive methods, such as blockchain and large-scale machine learning become parts of modern software systems. Tools and methods in support of energy assessment of software [11] and energy-aware design are increasingly easier to incorporate into everyday software engineering practices.

## 2. The SusDevOps framework

SusDevOps is a software delivery framework that treats sustainability as a first principle of software. As the name suggests, the framework builds on well-established DevOps practices of integrated software development ("Dev") and software operation ("Ops") and extends them with the practice of sustainability ("Sus"). To keep software delivery agile despite the distinguished role of sustainability, SusDevOps defines a lifecycle model (Figure 1) that integrates sustainability-related activities with the software development and delivery activities of traditional DevOps.

In SusDevOps, sustainability is not a metric anymore, it is a goal that drives product design and delivery. This is an important improvement over current software delivery frameworks. For example, in traditional DevOps, sustainability is approached as a quality metric of the software [4]; and while BizDevOps [7] improves over this by treating sustainability as a business goal rather than a technical quality metric, it fails to connect sustainability to software requirements.

[3] https://sustainabledevelopment.un.org/content/documents/5987our-common-future.pdf

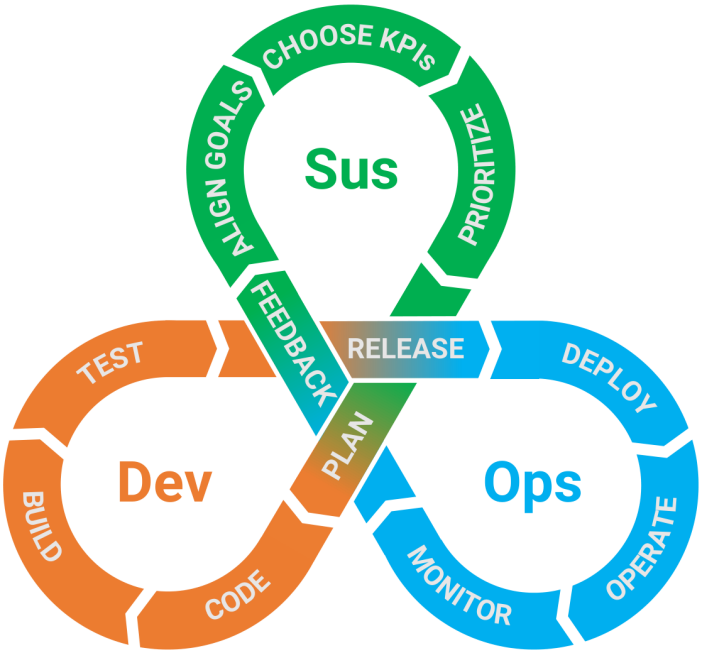



Figure 1: The SusDevOps framework.

The key sustainability-related activities of SusDevOps are the following.

- **Align goals**. First, sustainability goals are formulated based on the feedback from operations and are aligned with the goals of the product. Sustainability goals are more often contradicting than technical requirements and the resolution of these contradictions requires a dedicated activity. This requires a holistic, system-level view, as sustainability goals touch upon technical aspects and business operations as well. In the case study, for example, environmental sustainability was articulated as a key expectation from the users. Aligning it with the technical parameters of the software is not trivial and requires understanding the impact environmentally-friendly software design has on functional and extra-functional properties.
- **Choose KPIs**. After sustainability goals have been articulated, they need to be properly underpinned by measurable key performance indicators (KPIs). KPIs are indicators focusing on the aspects of software systems that are most critical for the success of the company [12]. However, establishing the right KPIs is challenging. In a recent article, Fatima et al. [13] elaborate in detail on these problems and provide practitioners with a template-based tool. In the case study, CO2 emission has been chosen as a heuristic for environmental sustainability. This can be now tied to the energy consumption of the software.
- **Prioritize goals**. Finally, priorities among goals are set. Typically, companies will face trade-off questions between sustainability and revenue, performance, etc. Dedicating a specific step to resolve these questions is necessitated by the potential involvement of higher-level decision-makers.

Some of the important benefits of the SusDevOps framework are the following.

- **End-to-end understanding**. The framework aligns elementary software delivery activities with sustainability activities. This allows stakeholders along the software delivery process (including IT departments) to understand when and how to deal with sustainability, what input to gather for sustainability-related decision-making, and which sustainability goals to translate to software requirements.
- **Actionability**. The framework defines activities for the sustainability practice, and for each activity, it recommends techniques and tools. This lowers the barriers to adopting SusDevOps and simplifies aligning it with business processes.
- **Agility**. The framework promotes systematic yet rapid sustainability-related decision-making in the iterative-incremental DevOps context. This allows for reacting to change—which is very much pronounced in end-user sustainability requirements—more efficiently.

The sequential alignment of the Sus and Dev practices might challenge the agility of SusDevOps. The guiding principle of SusDevOps is that requirements are first investigated through the filter of sustainability before reflecting on their technical aspects. These ideas have been recognized in sustainable software engineering, especially in the context of requirements elicitation, e.g., by Becker et al. [14]. Promoting sustainability to such a prominent position, and treating it as a principle practice along with development and operations, fosters a culture shift in which new techniques and tools in support of delivering sustainable software emerge naturally. Iterations between the Sus and Dev practices before releasing the product might be still introduced if needed, just like iterations are frequent in the Dev phase in traditional DevOps before releasing the product.

In the following, we demonstrate the sustainability-related activities of the framework through an illustrative case.

# 3. Illustrative case

We use the example of a software development startup company to illustrate the usage of the SusDevOps framework in a typical business-to-consumer (B2C) setting.
The company develops a software product of which the key feature is high-precision simulation driven by pre-trained AI models. User satisfaction is mainly influenced by the precision of the product. Being a startup, the company uses the majority of their revenue to regularly improve their computing capacity to provide better precision through better-trained AI models.
After the user base grew large enough, the company noticed that user satisfaction is also influenced by the perceived environmental sustainability of the software, such as energy efficiency. Given the B2C model, revenue might be significantly impacted by emerging user needs, leading to less liquidity to improve the product.

The main challenge in this case is to stay agile in the delivery process, that is, to maintain the right velocity while reacting to sustainability needs as they emerge. These needs have to be assessed systematically, trade-offs between technical and sustainability goals need to be

identified, those trade-offs have to be mapped onto the functionality of the software through requirements, etc. Traditional software engineering delivery processes might struggle to handle this challenge efficiently due to the convoluted and vague notion of sustainability.

However, these challenges can be addressed through a number of activities along the lifecycle defined by the SusDevOps framework. In the following, we briefly demonstrate these activities.

## Monitor (Ops) and Feedback (Ops-to-Sus)

Monitoring is a DevOps activity that helps keep track of functional and non-functional metrics. In SusDevOps, this activity also includes keeping track of sustainability properties (e.g., the energy consumption of software), as well as the satisfaction of end-users. In anticipation of emerging sustainability-related needs of users, companies can run market studies and use the feedback functionality of applications to gather valuable leads. In our case, a market study finds that users value perceived sustainability highly.

This information is fed back to the development team to support the planning of the next release.

## Align goals

In the first sustainability activity, goals are formulated based on the feedback and are aligned with the goals of the product. Causal loop diagrams (CLD) are an appropriate formalism to capture system dynamics. A practical and accessible overview of using CLDs in software engineering has been provided by Penzenstadler et al. [3].
As the name suggests, CLDs aim to visualize the causal relationships between system variables. System variables might influence each other through positive or negative causal loops, meaning that two variables change in the same or opposite direction, respectively.
Figure 2a shows the original system before feedback. *Precision* is a key influencing factor of *User satisfaction*: the higher the precision, the higher the user satisfaction. This causal relationship is shown by the link between the variables denoted by a + sign. By the same logic, higher *User satisfaction* leads to higher *Revenue* which, in turn, allows the company to further increase the *Precision* of the software. This leads to a *reinforcing loop* (denoted by “R”) of *Technical performance*. Reinforcing loops are associated with exponential increases and decreases—in this case, with a rapid increase in revenue. Of course, hardware limitations and market saturation keep the system in check, and variables plateau at some point.

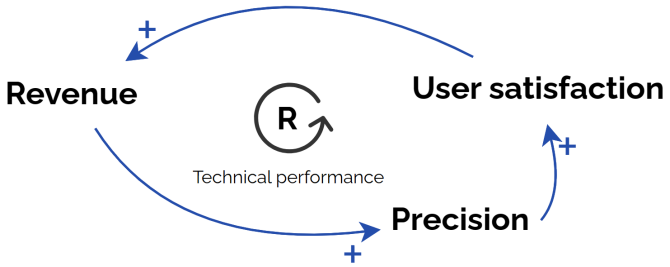
Aligning the new goal means adding *Perceived sustainability* to the system in order to form loops in later steps. *Perceived sustainability* positively influences *User satisfaction*. To form loops and to further detail the diagram, we need more detailed variables.

## Choose KPIs

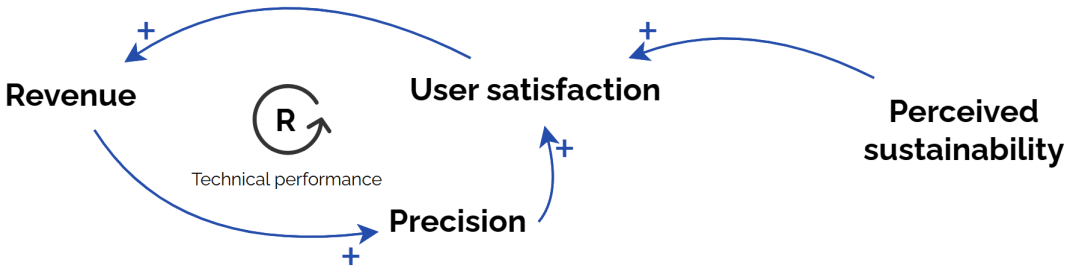
After the new goal has been articulated, it has to be properly elaborated by choosing the right KPIs. KPIs are indicators focusing on the aspects of software systems that are most critical for the success of the company [12]. However, establishing the right KPIs is challenging. In a recent article, Fatima et al. [13] elaborate in detail on these problems and provide practitioners with a template-based tool.

In our case, Figure 2b is extended by two KPIs, shown in Figure 2c. To approximate environmental sustainability, the CO2 emission of computation is used as a KPI. CO2 emission impacts perceived sustainability negatively, hence, a negative relationship is drawn in the causal loop diagram. To approximate precision, the number of floating-point operations (FPO) can be used as a KPI. Common choices are the cumulative energy consumption and cumulative power consumption of computation as well, but the relationship between energy, power, and run time is ambiguous. Power (SI-unit: watts) is the amount of work (SI-unit: joules) over time. Very simplistically, we say *power = work / time*. Conversely, *energy = power × time*. Reducing computational resources results in lower power consumption, but prolongs computation time; thus, due to the latter equation, the change in energy is unclear. Adding more resources to reduce computation time will result in shorter computation time but higher power consumption and again, the change in energy is unclear. FPO calculates the amount of work needed for computation directly and is thus tied to the amount of energy consumed [15], and in the long term, to CO2 emissions required to produce this energy. This now allows for establishing two new links.

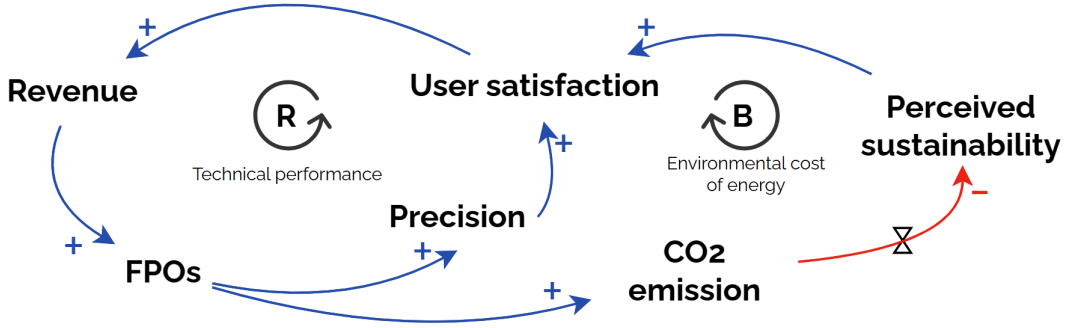
First, the link between Revenue and Precision is replaced by a link between Revenue to FPO, and FPO to Precision. The effect of influences does not change, as every link remains a positive causal link: with higher revenue, the company can afford more FPOs, leading to higher precision, and increasing user satisfaction which, in turn, leads to increased revenue—forming a reinforcement loop. Second, a new link is placed between FPO and CO2 emission. The link is a positive causal link: the higher the FPO, the higher the CO2 emissions are. This now forms a new balancing loop (denoted by “B”). That is, with the increasing CO2 emission, Perceived sustainability decreases, eventually leading to lower Revenue, lower number of FPOs, and lower CO2 emissions. Eventually, the loop will find a balance among the variables and stabilize.




*(a) Original understanding before feedback.*




*(b) Updated understanding after feedback.*




*(c) New KPIs (FPO and CO2 emission) and new links forming a new loop.*

Figure 2. Evolving understanding of sustainability factors visualized in a causal loop diagram as the chosen tool in this case study.

## Prioritize

In the final activity of the sustainability practice of SusDevOps, priorities among goals are set. The company faces an important question: what is the right trade-off between precision and perceived sustainability? This question emerged solely because the previous activity identified a balancing loop in the system. From Figure 2c we know that both precision and CO2 emissions can be expressed in relation to the number of FPOs a computation requires to achieve its goals. The challenge is that parameter sensitivity between precision and CO2 emission is not known unless empirical data is collected first. In alternative terms, it is not known how much precision and CO2 emission will decrease by a unit of decrease of FPO. Evidence from the AI domain [15] suggests that by just sacrificing 0.5% of precision, FPO can decrease by as much as 30-35%. It is still not entirely clear how this decrease in FPO will impact CO2 emissions, but surely, it is a step in the right direction. Further experiments can shed light on this relationship in the context of the specific software product.

Factors that commonly influence prioritization are user needs, business goals, corporate values, and in some cases, laws and regulations. Specifically, corporate values influence the leverage point a company chooses to influence the ecosystem their software product is a part of. For example, following the leverage point clusters of Penzenstadler et al. [3], a company might not even wait for emerging user needs, but take a proactive stance and choose the highest leverage points to *change the intent of the system and stakeholders*. This could be achieved by

raising sustainability awareness, for example, through gamification that rewards more energy-efficient usage of the software.

### Plan (Sus-to-Dev)

To conclude the sustainability-focused part of the SusDevOps process, plans are formulated based on aligned, elaborated, and prioritized goals. In a typical software engineering setting, this mainly means formulating requirements or change requests for the software [15]. This activity also includes formulating engagement plans with users to advocate change and bring users on board with the new, more sustainable product.

Eventually, the cycle is completed with the traditional DevOps activities of software development and operation.

## 4. Challenges in adopting SusDevOps

Here are some of the key challenges practitioners should anticipate when adopting SusDevOps.

- **Proper Monitoring and Feedback mechanisms should be identified.** The lack of proper mechanisms to gauge end-users' sustainability needs challenges the agility of the delivery process. This might slow down the software delivery process and threaten user privacy. In B2B settings, this challenge can be addressed by the product owner working closely with clients. B2C settings, however, might necessitate innovative ways to engage with users.
- **New competencies need to be developed.** While sustainability expertise might be brought in as a service, it is more *sustainable* for a company to develop internal skills. This entails expanding the skillsets of business analysts, product owners, and software engineers with the foundational concepts, models, techniques, and tools that help along the sustainability practice of SusDevOps. In the case study, each sustainability-related step was executed by these internal professionals.
- **Stakeholder buy-in is paramount.** As sustainability transcends the software's purpose [14], it requires active support from all stakeholders in a company. However, the highly stratified nature of sustainability (i.e., having different meanings at different levels of organizational hierarchy) challenges such efforts. To simplify this challenge, Becker et al. [14] recommend minimizing the number of involved stakeholders in sustainability decisions and focusing on those who have influence.
- **Knowledge management is key.** Sustainability goals are inherently interdisciplinary and multisystemic: their sound interpretation is possible only by investigating them from multiple angles. Even the simple illustrative case in this article touched upon business goals, technical goals, and environmental goals. Thus, facilitating processes to continuously maintain the map of sustainability goals is key to having the proper understanding of the sustainability of the developed software within its extended socio-technical context.

- **Watch out for accidental greenwashing.** Greenwashing is the deceptive strategy of magnifying a company's sustainability efforts. Greenwashing can be accidental too, for example, by focusing on a single environmental attribute while ignoring others.[4] Understanding sustainability goals and their alignment with the overall product strategy, and subsequently choosing the right KPIs is a key moment in making sound sustainability decisions and by that, avoiding accidental greenwashing. In a recent article, Fatima et al. [13] provide actionable pointers and a template-based tool for developing KPIs. It is important that adopters of SusDevOps embrace continual improvement. Especially in convoluted problems, such as the sustainability of software, the best KPIs can be developed over time.

To help address these challenges, Table 1 provides examples of techniques and tools in support of SusDevOps.

Table 1. Techniques and example tools in support of the *Sustainability* practice of SusDevOps.

| Activity | Techniques | Example tools |
|---|---|---|
| Monitor and Feedback | User surveys, focus group interviews | Built-in survey functionality |
| Align goals | System dynamics, Causal loop diagrams | Visual Paradigm[5], Miro[6] |
| Choose KPIs | KPI templates | Fatima et al. [13] |
| Prioritize goals | Market dynamics analysis, empirical studies, quantitative simulations | Vensim[3] |
| Plan | Requirements engineering and validation | Requirements templates, issue tracking systems |

## 5. Conclusion

In this article, we proposed a novel, sustainability-first software development and delivery framework, SusDevOps. The framework defines an integrated lifecycle model based on well-established DevOps principles. Through the case of a software development startup company, we demonstrated the utility of the proposed framework, its alignment with standard software development and operation processes, and recommended techniques and tools to cover the various sustainability-related stages of the lifecycle.

[4] https://www.un.org/en/climatechange/science/climate-issues/greenwashing
[5] https://www.visual-paradigm.com/
[6] https://miro.com/

Of course, SusDevOps is no silver bullet and might not serve as a blueprint for every software project, but it provides a reasonably general, yet applicable framework to organize team effort around delivering sustainable software. It is important that adopters maintain a pragmatic standpoint and implement SusDevOps through techniques and tools that fit their goals, skills, and organizational capabilities.

## Acknowledgments

The author would like to thank Dominik Bork (TU Vienna, Austria) for his feedback on this article.

## References

[1] L. Belkhir and A. Elmeligi, “Assessing ICT global emissions footprint: Trends to 2040 & recommendations,” Journal of Cleaner Production, vol. 177. Elsevier BV, pp. 448–463, Mar. 2018. doi: 10.1016/j.jclepro.2017.12.239.

[2] B. Penzenstadler, A. Raturi, D. Richardson and B. Tomlinson, "Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century" in IEEE Software, vol. 31, no. 03, pp. 40-47, 2014. doi: 10.1109/MS.2014.22

[3] B. Penzenstadler et al., "Software Engineering for Sustainability: Find the Leverage Points!," in IEEE Software, vol. 35, no. 4, pp. 22-33, July/August 2018, doi: 10.1109/MS.2018.110154908.

[4] P. Lago, S. A. Koçak, I. Crnkovic, and B. Penzenstadler, “Framing sustainability as a property of software quality,” Communications of the ACM, vol. 58, no. 10. Association for Computing Machinery (ACM), pp. 70–78, Sep. 28, 2015. doi: 10.1145/2714560.

[5] C. Ebert, G. Gallardo, J. Hernantes and N. Serrano, "DevOps," in IEEE Software, vol. 33, no. 3, pp. 94-100, May-June 2016, doi: 10.1109/MS.2016.68.

[6] J. Alonso, R. Piliszek and M. Cankar, "Embracing IaC Through the DevSecOps Philosophy: Concepts, Challenges, and a Reference Framework," in IEEE Software, vol. 40, no. 1, pp. 56-62, Jan.-Feb. 2023, doi: 10.1109/MS.2022.3212194.

[7] V. Gruhn and C. Schäfer, “BizDevOps: Because DevOps is Not the End of the Story,” Communications in Computer and Information Science. Springer International Publishing, pp. 388–398, 2015. doi: 10.1007/978-3-319-22689-7_30.

[8] D. Kreuzberger, N. Kühl and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," in IEEE Access, vol. 11, pp. 31866-31879, 2023, doi: 10.1109/ACCESS.2023.3262138.

[9] B. Penzenstadler and H. Femmer, “A generic model for sustainability with process- and product-specific instances,” Proceedings of the 2013 workshop on Green in/by software engineering. ACM, Mar. 26, 2013. doi: 10.1145/2451605.2451609.

[10] I. David and D. Bork, “Towards a Taxonomy of Digital Twin Evolution for Technical Sustainability,” Proceedings of the ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems Companion, MODELS 2023 Companion. IEEE, 2023.

[11] G. Kalaitzoglou, M. Bruntink, and J. Visser, “A Practical Model for Evaluating the Energy Efficiency of Software Applications,” Proceedings of the 2014 conference ICT for Sustainability. Atlantis Press, 2014. doi: 10.2991/ict4s-14.2014.9.

[12] D. Parmenter, Key performance indicators: developing, implementing, and using winning KPIs. John Wiley & Sons, 2015.
[13] I. Fatima, M. Funke, and P. Lago, “From Goals to Actions: Providing Guidance to Software Practitioners with KPIs”, Authorea Preprints, 2023.
[14] C. Becker, et al., "Requirements: The Key to Sustainability" in IEEE Software, vol. 33, no. 01, pp. 56-65, 2016. doi: 10.1109/MS.2015.158
[15] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green AI,” Communications of the ACM, vol. 63, no. 12. Association for Computing Machinery (ACM), pp. 54–63, Nov. 17, 2020. doi: 10.1145/3381831.

**Istvan David** is an Assistant Professor of Computer Science at McMaster University, Canada, where he leads the Sustainable Systems and Methods Lab. His research focuses on modeling and simulation of complex systems, with a particular interest in sustainability of software-intensive systems. He is active outside of academia as well, mainly in consulting companies along their digital transformation and sustainability transformation processes. Contact him at istvan.david@mcmaster.ca.