Editor: **Christof Ebert**
Vector Consulting Services
christof.ebert@vector.com

Green IT and Green Software

Roberto Verdecchia and Patricia Lago, Vrije Universiteit Amsterdam
Christof Ebert, Vector Consulting Services
Carol de Vries, PhotonDelta

From the Editor

Ecologic behavior is the need across the world to mitigate the impacts of climate change. Software and IT play a pivotal role toward ecologic behaviors for many reasons. Being aware that IT systems alone already consume 10% of global electricity, the leading software practitioners must embark on green IT and green coding. Read in this article about hands-on guidance on how you can contribute toward more ecologic software. I look forward to hearing from you about this column and the technologies that matter most for your work.—*Christof Ebert*

SOFTWARE AND IT usage are continuously growing to keep our society active and manage our individual lives. But as they grow, their energy demand is exploding. By 2030, data centers alone will already consume some 10% of the global electricity.[1] Including the Internet, telecommunications, and embedded devices, the energy consumption will be one-third of the global demand. Understanding that end users only consume what we offer, it is the community of software developers who must become active in ecologic behaviors. Green IT is the call of today. Each single line of code that we develop today may still be running years from now on zillions of processors, eating energy and contributing to global climate change.

Green IT and green coding describe a paradigm switch in which software engineers, developers, testers, and IT administrators can make their solutions and services more energy efficient. Every single software person can contribute. In this article, we provide hands-on guidance on how to reduce the energy waste of your software and thus contribute to more ecologic behaviors.

Green IT

With the introduction of high-bandwidth data transfers, affordable data plans, the generalized migration of software applications and data management to the cloud, the wide usage of streaming services, and, obviously, the many embedded computers in our everyday lives, digital infrastructures are experiencing an ever-growing demand for energy. While digital transformation looks impressive from an economic perspective, it has its downside on the ecologic footprint of these businesses.

An immediate action is to adopt more renewable energy. Energy-hungry companies such as Microsoft, Google, and Amazon are currently investing in water energy, for example, to cool their data centers; solar energy; and wind farms. Many companies engage in trading CO_2 certificates to give a green color to the energy waste of their data centers. But renewable energy only "cures" the symptoms. It does not really solve reducing the need for energy.

Digital Object Identifier 10.1109/MS.2021.3102254
Date of current version: 22 October 2021

To address the root causes, we need green IT, such as lower energy consumption in data centers and for cloud services, as well as green code, which addresses how to efficiently organize the software itself.

Green IT provides solutions to sustainably reduce the energy needs of data centers and cloud services. As an example, the Lower Energy Acceleration Program (LEAP) has been launched to explore alternative solutions toward a sustainable growth of the data-center industry. As one of our goals, we want to develop a technology landscape for energy-efficient digital infrastructures.

As a first step, we conducted a series of interviews and focus groups with various digital infrastructure stakeholders, like data centers, cloud service providers, and cloud customers from both the public and private sectors. The participants were asked 1) to indicate and describe the solutions that in their experience would contribute to the sustainability of digital infrastructures, and 2) to map each solution on different temporal horizons according to the related perceived level of readiness in terms of widespread adoption and full impact. The resulting temporal horizons are:

- horizon 1 (H1): State of the art, that is, solutions for today
- horizon 2 (H2): Within the next four to six years, that is, solutions for the near future
- horizon 3 (H3): Beyond six years, that is, further evolution.

We also identified four types of solutions, namely, technical, social, and environmental solutions, and paradigm shifts. An overview of the solution landscape is presented in Figure 1 and described in further detail in Verdecchia et al.[2]

Short-term solutions are readily available for adoption. In Figure 1, we group them as short-term horizon H1. Horizon H1 namely includes moving to the cloud (relocating data, computational, and software capabilities from on-premise locations to the cloud). With the popularization of cloud-native and serverless applications, resources are accessed on demand, profiting from hyperscale data-center energy-efficiency optimizations. Such solutions entail the use of green energy sources (such as solar farms), heuristics for hyperscale hardware management (for example, reuse of dissipated heat), deployment of domain-specific hardware [artificial intelligence (AI) accelerators], and tight collaborations between software and hardware companies to develop dedicated hardware components (integrated infrastructures). H1 also sees the rise of energy-aware software optimizations, such as

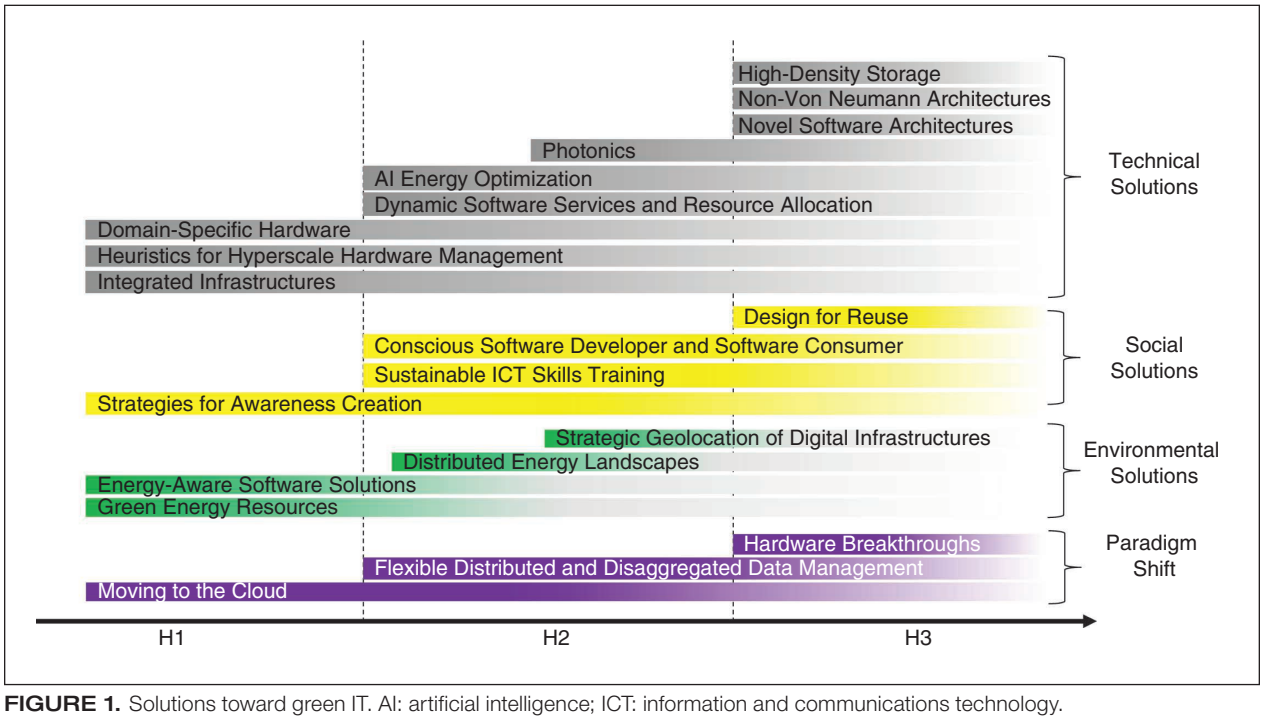

FIGURE 1. Solutions toward green IT. AI: artificial intelligence; ICT: information and communications technology.

GREEN TESTING

Testing consumes not only most of the time and effort in a software project, but it also heavily contributes to energy waste. Testers know that it is not the amount of testing but the quality of testing that matters. Still, in our consulting practice we see many test teams that pile up test cases without any underlying test strategy. With one client, we reworked a test strategy that was built upon a huge amount of brute-force testing toward intelligent testing.

Previous testing methods followed the traditional scheme of collecting test cases. The low-level code testing used a Jenkins pipeline with automatic unit test cases. Above that was an integration test, which ensured that functions and interfaces worked properly. The system test was a more experience-driven set of test clusters for base functions, UX, and networking to other components of a yet bigger system. So far, so unstructured.

All three test layers were basically collections of test cases, which were automated as much as possible, but not controlled with test methods, such as coverage, risk orientation, heat maps, change logs, and traceability for regression testing. The test share compared to the total project effort was growing by the year from initially 30% to more than 50% when we started. The test effort in such a brute-force scheme is highly correlated with test energy consumption, which was visible, not the least in the amount of cooling necessary in summer for their on-premise test equipment.

As a first step, we introduced test methodologies and systematically structured the test process. For instance, we introduced a static code analysis to identify hot spots that needed more low-level testing. Test cases and test management were instrumented to identify test effectiveness and test efficiency. This allowed us to substantially reduce test cases by removing redundancies. At the same time, the test effectiveness was improved by targeting previously uncovered functions and code segments. Test-driven methods were introduced to start from the beginning of a change or new functionality with testability in mind. Test-driven development (TDD) facilitates not only a basic set of unit test cases, but also ensures better code quality. Test-driven requirements engineering (TDRE) specifies functions as test cases. Both TDD and TDRE facilitate a much more efficient regression testing, which, with many changes over the project time and product lifetime, saves lots of effort—and energy.

In a next step we introduced risk-oriented testing. In a distributed system, undesirable behavior and malfunctions can arise because there has been a software change at one point that breaks through to other components. This raises numerous questions: How can the function of a system be ensured if changes take place in the subcomponents? How can the safety and reliable behavior be guaranteed if software changes are made to individual components during operation? Unlike brute-force testing with its low efficiency, risk-oriented testing evaluates the dependencies of functions and code, both internally, that is, as white-box testing, as well as externally, as black-box testing. For example, an external dependency is a critical functionality, such as billing of a finance service or a safety-critical function in a robot.

Finally, we introduced intelligent testing with AI to support validation. There are many AI-based test methods, ranging from rule-based systems, fuzzy logic, and Bayesian nets to the multiple neural network approaches of deep learning. The potential for intelligent testing is manifold: On a system level there are questions on which test cases must be executed, and to what extent? Intelligent validation, such as cognitive testing, helps in selecting and creating test cases. In a first step, an assistance functionality is set up to identify priorities in an existing set of use cases. As a result, the validation expert can test more quickly and with a better coverage of situational relevant scenarios. On the level of component or module testing, it is also required to identify relevant cases. This can range from a simple support on how to feed the system with scenario-based data generation and check on the outputs with a test oracle, toward complex algorithms that automatically create test cases based on the code or user interface.

As a result, the test effort was reduced, while at the same time the test effectiveness was improved. Given the high amount of test equipment and its steady energy consumption, we achieved a double-digit reduction in electrical energy.

energy-efficient use on demand and energy-aware management of idle servers. From a social perspective, communication raises awareness on the environmental impact of personal digital infrastructure usage.

Solutions for the Near Future

For the near future, we see our second paradigm shift (Figure 1, H2), namely flexible distributed and disaggregated data management. Supported by steady advancements in communication technologies and the growing affordability of computational power, a vast number of computational tasks will elastically move as near as possible to both the consumer premises and the increasingly ubiquitous intelligent systems, (for example, via decentralized computing) and distributed networks of computational nodes. Solutions of H2 reflect the second paradigm shift and entail the appearance of distributed energy landscapes (prosumers), dynamic software service and hardware resource allocation, and the strategic positioning of digital infrastructures near to their end users to ensure high bandwidth, keeping the communication-energy consumption low. (See also "Green Testing.")

The ever-growing energy consumption of AI-based systems will be mitigated in the distributed paradigm of H2 with the rise of federated learning algorithms, affordable AI chips, and data usage plus compression strategies. Energy-aware software optimizations will continue to prosper, with consolidated tactics for software energy efficiency, seamless virtualization of hardware components, and energy-driven workload optimizations. From a social perspective, the growing demand for sustainable information and communications technology (ICT) skills will require the training of new professional profiles. The awareness that emerged in H1 will develop in H2 into conscious software developers and consumers, who will feel personally accountable for their actions and hence develop and demand more sustainable solutions. Unlike H1, where the average lifecycle expectancy of digital infrastructure hardware components equals approximately two years, H2 is also expected to witness a growing trend of design for hardware reuse and lifecycle management practices.

> Ecologic behaviors and climate protection are not a threat, but a great business opportunity.

Further Evolution

As we look further into the future, and uncertainty grows, we can foresee the energy efficiency of digital infrastructures being supported by our last paradigm shift, namely novel hardware breakthroughs. While already supporting to a large extent the distributed and disaggregated paradigm, photonic technology will shape itself in H3 into integrated photonics, mitigating the culmination of microelectronics computational advancements. In H3, non-von Neumann architectures (such as neuromorphic computing) will also become available for adoption, allowing one to carry out complex computations at a fraction of their current energy cost. By making use of electron spins or relations between protons and electrons, high-density storage solutions will also become in use during H3. From a software perspective, to support the drastic hardware changes of H3, software systems will require novel software architectures to best fit software artifacts to the new underlying hardware.

Green Software

Now is the time that IT practitioners include green software techniques into their own practices. Here we will provide some hands-on guidance for actually implementing green IT. Table 1 provides an overview with reference points. Software practitioners can contribute in many ways to accelerate the transition to an energy-efficient digital infrastructure.

Move to the Cloud

Cloud services typically decrease energy consumption compared to on-premise data centers because they have less overhead and more efficiency in scalability. Depending on the sources, between 80 and 90% of businesses have already migrated to the cloud.[3,4] This significant investment does not mean that "the job is done" (also given that energy efficiency has hardly ever been a concern); rather, it reveals that effective and technically sustainable migration demands major rearchitecting.

It calls for the following concrete actions: When replacing on-premises

Table 1. Examples of short-term solutions and related energy impacts.

H1 Solution Type	Example of Specific Technique (If Applicable)	Observed Energy Impact	Source
Domain-specific hardware	Utilizing AI accelerators	2.5% improvement of power performance every year	IBM, 2021. "Introducing the AI chip leading the world in precision scaling. https://www.ibm.com/blogs/research/2021/02/ai-chip-precision-scaling
Strategies for awareness creation	Providing actionable guidelines for data-center operators to configure power management settings	10–13% energy savings	Netherlands Enterprise Agency, 2020. "Happy Flow manual 1.0. Energy-efficient design of data centers using power management and virtualization." https://amsterdameconomicboard.com/app/uploads/2020/10/Happy-Flow-Manual-LEAP.pdf
Energy-aware software optimizations	Shutting down idle servers	84% of energy reduction	Asperitas, 2017. "The datacentre of the future." https://www.asperitas.com/whitepapers/the-datacentre-of-the-future
Heuristics for hyperscale hardware management	Replacing fan-based with liquid-based cooling	6–45% reduction of IT energy footprint	Rais et al., 2018. "Energy-saving potential of separated two-phase thermosiphon loops for data center cooling," *Journal of Thermal Analysis and Calorimetry.*
Green energy resources	Replacing brown with green energy resources	100% consumed energy produced by renewable energy resources	Google, 2021. "Cloud Sustainability." https://cloud.google.com/sustainability
Moving to the cloud	Migrating from on premises to the cloud	64% of energy reduction	Accenture, 2020. "Cloud computing and sustainability: The environmental benefits of moving to the cloud. https://www.accenture.com/us-en/insights/strategy/green-behind-cloud

functionality with cloud-native services, the services selection should be informed by transparent indicators (which must be provided by the cloud service providers) for the related energy and resource efficiency. Also, when reengineering and rearchitecting applications for the cloud, practitioners should include software tactics to manage data more efficiently to decrease the necessary resources and thus energy for data storage, computing resources to process data, and networking resources to transfer data among cloud, edge, and customer sides. Examples of such energy-efficient data management tactics may address dataflow optimization, data deduplication, and smart data compression based on frequency of use. Serverless computing and function-as-a-service facilitate seamless scalability of resource consumption. Data-usage profiling and AI/machine learning techniques for profiling applications can discover data-usage patterns and adjust performance toward more efficiency.

Ecology by Design

Designers determine the ecologic footprint of their software. As shown in Figure 1, we will witness major paradigm shifts that demand an increased distribution of applications, data, and computing resources among local, edge, and remote cloud premises. To be ready for this shift, practitioners should already start to rearchitect their products with flexibility in mind. Embedding flexibility in software engineering, as has never been done before, allows

us to develop future-proof (technically sustainable) software products as well as take advantage of the promises of smart data staging and computational offload for delivering energy efficiency. As a designer, ensure that new software versions do not require more computing power, hard-disk space, or bandwidth than before. This is easily feasible by optimizing algorithms. A simple checkpoint is that previous core functionality remains available on older hardware. As a developer for embedded software or Internet of Things (IoT) components, for example, massively deployed software components residing close to energy-hungry hardware, there are several best practices for green software, that is, avoid always-on, choose efficient algorithms, use pulse- rather than continuous connectivity (which also better protects your systems for cybersecurity), and reduce unnecessary precision and storage demands.

Green User Experience

Users of software dominate the consumption because they scale eventually to the millions. Imagine the simple move from receiving high-resolution advertisement movies toward ad blocking. This is under the control of each user and will save on both bandwidth and energy consumption, in data centers, switches, and definitely in each device, where less GPU power would be necessary. User experience (UX) matters, and it can be made adjustable for less consumption. As a UX designer, make sure that users can configure software according to their individual needs, such as night mode, movie blocking, reduced-resolution movies, and so on. UX should clearly show the power usage of the respective software. Create awareness by showing with red/amber/green how much energy is currently consumed. This can be linked to, for example, processer usage. Make users able to manage the energy consumption of hardware as efficiently as possible, including providing energy-saving modes to easily switch down hardware when not used. This can even be automated by tracking usage and bringing unused processes to a sleep mode or simply terminating them.

> Creating energy-aware software and infrastructures calls for the much more explicit and pervasive inclusion of contextual information in design decision making.

Blending Software and Its Context

Creating energy-aware software and infrastructures calls for the much more explicit and pervasive inclusion of contextual information in design decision making. The context involves planning for known changes, adapting to unknown ones, and including systemic perspectives like different sectors, such as the energy and ICT sectors; stakeholders, such as the priorities, investments, and constraints of organizations; and systems, such as the role of data centers for hyperscaling and municipalities for microclouds and energy ecosystems in built environments.

Let us look to blockchain technology as an example. Blockchains are increasingly used to establish electronic contracts, such as those in finance transactions, replacement parts delivery in industry and aerospace, medical treatments, and so on. Yet their energy hunger is growing in parallel. The major critique of any distributed ledger technology is its waste of energy in maintaining the steadily growing transaction history.[5] For instance, the Bitcoin currency gets more value primarily because it needs more computer power, which wastes energy. Blocks are validated using a proof-of-work riddle. Only the first peer to solve it is rewarded, which has created a race for IT power to solve the riddles. Miners build clusters of specialized, optimized hardware. These clusters require power to operate and to cool down. The energy needed just to maintain the Bitcoin currency is estimated to be beyond that of Switzerland.[1] As a software engineer, try to avoid energy-hungry blockchain transactions and move toward alternative distributed consensus techniques.

Competence

To move toward an energy-efficient cloud, practitioners must embed strategies for awareness creation for measuring, monitoring, and visualizing the energy footprint of the software they deliver. The software energy consumption must be measured, estimated, and predicted at all levels, from the used computing

resources like CPUs/GPUs, to the generated data traffic, and from the single cloud-native service and container, to whole systems and apps. Education and training programs must equip new talents and the existing workforce with the related know-how.

Ecologic Hardware

Continuously demanding, installing, and using the latest advertised highest-performing hardware highly contributes to both e-waste and energy consumption. When designing software, consider that it can also be deployed on previous hardware platforms. Aging is normal, but it should not be measured in months—but rather in several years. Embedded designers are aware of the problem and have the solutions. When they design a piece of software for avionics, industry automation, or simple upgradeable IoT devices, they decouple the software from the underlying hardware, thus allowing the hardware to be used for many years. Over-the-air upgrades can introduce new and fancy features, but also security patches and error corrections—without demanding new hardware. For instance, mobile vehicles, from automobiles to trains and airplanes, are typically run for years, if not decades, on the same platform by simply upgrading some controllers toward new hardware, while keeping the entire system active.

A car typically has a lifetime of 10 years, during which it will occasionally get new hardware devices if the previous set has reached end of life. Trains and other industry systems have a much longer lifetime. So why would we need to buy new consumer goods on a yearly base? To reduce your own personal ecologic footprint, consider buying refurbished equipment. It is a simple, yet effective, step for your own ecologic contribution.

Ecologic Behaviors Are the New Normal

Ecologic transformation is the next big wave of innovation after the digital transformation. Every business will evolve toward ecologic behavior. Investors are currently changing behaviors toward financing based on ecology and sustainability. They do that not only for ethical reasons, but simply because the economic risks of traditional business are too high. Another driver is the worldwide market changes in behaviors across countries. This currently still looks unbalanced; some parts of Europe are rather advanced, while other countries maintain old-fashioned, energy-wasting behaviors. Yet, this will change if people look toward not only low pricing, but also ecology and sustainability as a new normal. Buying power will have an increasingly greater impact on any business.

Behaviors matter for ecologic transformation. In one aspect, this means that we should evangelize in our respective spheres. During the past two years, online collaboration has exploded to mitigate the COVID-19 pandemic risks. Tools such as Skype, Teams, and Zoom have replaced the traditional meet and greet—at no cost and with big wins on efficiency and flexibility. Traveling suddenly has become a legacy with its high time and cost overheads, combined with its huge ecologic footprint.

> The Bitcoin currency gets more value primarily because it needs more computer power, which wastes energy.

INDUSTRY SURVEY

This column will further address major challenges in software and IT. With the current fast-changing software and IT landscape, you are invited to take part in a short survey on industry trends 2022. Your opinion as a domain expert and decision maker is valuable. Please give us 2 min of your time and answer a few brief questions. After the end of the survey, you will exclusively receive its analysis, enriched with the experience and hands-on advice from current projects. With some luck, you will also receive a free copy of the IEEE book *Global Software and IT*. We will not use your personal data any further. Here is the link to the industry survey: www.vector.com/trends-survey

While meeting people in real life is still a good way to build relationships, online meetings allow us to adjust instantaneously to specific needs. In a recent survey of Vector Consulting Services and *IEEE Software*, a vast majority (more than 90%) of respondents acknowledged the advantages of online meetings and trainings.[6]

> Over-the-air upgrades can introduce new and fancy features, but also security patches and error corrections—without demanding new hardware.

Digital transformation has moved behaviors from traditional business to IT-driven business. Yet is has the drawback of currently wasting huge amounts of energy. Green IT and green coding are the countermechanisms that address the root cause by reducing the energy consumption of our IT and software systems. A good example of how fast this can change involves our laptops. While the energy consumption had been stable for a single laptop over the past three decades, their power and performance had improved by the millions. This was achieved by using advanced hardware design and dynamically switching off those arts that are currently idle or unnecessary. Automotive vehicle software is another example of how green software manifests. It used to grow with almost a hundred controllers eating kilowatts of energy and now is decreasing due to intelligent cloud services and new architectures. The immediate benefits are obvious: an extended battery life and fewer cooling needs. Obviously, we will move to green IT for more than ecologic reasons, but ecology will accelerate the transformation. See our "Industry Survey" and participate.

Too often, and wrongly, ecology and economy are perceived as antagonists. Ecologic behaviors and climate protection are not a threat, but a great business opportunity. Today, environmental protection is driven by technology, innovation, and smart IT. It is not the slogans alone, but we ordinary engineers who facilitate the ecology transformation by deploying green IT. The famous American politician and scientist

ABOUT THE AUTHORS

ROBERTO VERDECCHIA is a research associate at the Software and Sustainability group of Vrije Universiteit Amsterdam, Amsterdam, 1081 HV, The Netherlands. He can be contacted at https://robertoverdecchia.github.io or r.verdecchia@vu.nl.

PATRICIA LAGO is a full professor in software engineering at Vrije Universiteit Amsterdam, Amsterdam, 1081 HV, The Netherlands, where she leads the Software and Sustainability research group in the Computer Science Department. She is a cofounder of the Green Lab. She can be contacted at www.patricialago.nl or p.lago@vu.nl.

CHRISTOF EBERT is the managing director of Vector Consulting Services, Stuttgart, 70499, Germany. He serves on the editorial board of *IEEE Software* and is a Senior Member of IEEE. He can be contacted at https://twitter.com/christofebert or christof.ebert@vector.com.

CAROL DE VRIES is a program and technology manager at PhotonDelta, Eindhoven, 5656 AA, The Netherlands. His focus is on integrated photonics technologies and applications. Contact him at carol@photondelta.eu.

Benjamin Franklin once remarked: "If you fail to prepare, you prepare to fail." Green IT and ecologic business are today's answers toward saving the planet.

Acknowledgments

Some parts of this research received funding from the Netherlands Enterprise Agency Project "Energy Efficient Digital Infrastructures" (project RVO-TSE2200010) and support from the LEAP Initiative of the Amsterdam Economic Board.

References

1. S. Podder, A. Burden, S. Kumar Singh, and R. Maruca, "How green is your software?" Harvard Business Review, Sept. 2020. Accessed: Aug. 1, 2021. [Online]. Available: https://hbr.org/2020/09/how-green-is-your-software
2. R. Verdecchia, P. Lago, and C. de Vries, "LEAP technology landscape: Trends and scenarios," 2021. Accessed: Aug. 1, 2021. [Online]. Available: https://tinyurl.com/leapLandscape
3. J. Desjardins, C. Ang, and M. Lu, "Cloud computing growth," Visual Capitalist. Accessed: Aug. 1, 2021. [Online]. Available: https://www.visualcapitalist.com/cloud-computing-growth/
4. N. Galov, "25 Cloud computing statistics in 2020—Will AWS domination continue?" 2020. Accessed: Aug. 1, 2021. [Online]. Available: https://hostingtribunal.com/blog/cloud-computing-statistics/
5. C. Ebert, P. Louridas, T. M. Fernández-Caramés, and P. Fraga-Lamas, "Blockchain technologies in practice," *IEEE Softw.*, vol. 37, no. 4, pp. 17–25, July 2020. doi: 10.1109/MS.2020.2986253.
6. C. Ebert and B. Tavernier, "Technology trends: Strategies for the new normal," *IEEE Softw.*, vol. 38, no. 2, pp. 7–14, Mar. 2021. doi: 10.1109/MS.2020.3043407.
7. C. Ebert and R. Ruschil, "Test-driven requirements engineering," *IEEE Softw.*, vol. 38, no. 1, pp. 16–24, Jan. 2021. doi: 10.1109/MS.2020.3029811.

Digital Object Identifier 10.1109/MS.2021.3115655