Potentials of Green Coding - Findings and Recommendations for Industry, Education and Science

Dennis Junger [1], Max Westing [2], Christopher Freitag [1], Achim Guldner [2], Konstantin Mittelbach [1], Sebastian Weber [2], Stefan Naumann [2], Volker Wohlgemuth[1]

Abstract: Progressing digitalization and increasing demand and use of software cause rises in energy- and resource consumption from information and communication technologies (ICT). This raises the issue of sustainability in ICT, which increasingly includes the sustainability of the software products themselves and the art of creating sustainable software. To this end, we conducted an analysis to gather and present existing literature on three research questions relating to the production of ecologically sustainable software ('Green Coding') and to provide orientation for stakeholders approaching the subject. We compile the approaches to Green Coding and Green Software Engineering (GSE) that have been published since 2010. Furthermore, we considered ways to integrate the findings into existing industrial processes and higher education curricula to influence future development in an environmentally friendly way.

Keywords: Green Coding, Potentials of Green Coding, Curricula, Software Engineering, Literature Analysis, Sustainable, Engineering, Curricula, Education, Higher Education

1 Introduction and Motivation

Sustainability, as relating to ICT, in general, and software development and engineering in particular, is a research field of growing interest [An22]. While sustainability considerations in ICT have focused on hardware for the longest time, software is the driver of hardware energy consumption [Gu17]. Additionally, it affects the hardware life-cycle, e. g. as ever more demanding software systems quickly outgrow older hardware [Wi95]. Access to models, methods, and tools for GSE is elementary to enable sustainable software production. Furthermore, it is essential to make future IT professionals and researchers aware of ICT's sustainability aspects and equip them with the skills to consider them in their work adequately. Thus, in this literature analysis, we'd like to present available Green Coding practices and information on how educators can incorporate them into their teaching.

Sustainability is a complex topic that is commonly subdivided into dimensions or pillars. Within Green Coding, we focus on the environmental aspect of sustainability, dispensing with the other pillars, like ecological, social, and economic sustainability. Of course, these

[1] HTW Berlin, University of Applied Sciences, Industrial Environmental Informatics Unit, Wilhelminenhofstr. 75A, 12459 Berlin, Germany {dennis.junger|christopher.freitag|volker.wohlgemuth}@htw-berlin.de, konstantin.mittelbach@student.htw-berlin.de

[2] Institute for Software Systems, Environmental Campus Birkenfeld, Trier University of Applied Sciences,55765 Birkenfeld, Germany {m.westing|a.guldner|seb.weber|s.naumann}@umwelt-campus.de

are also important and are introduced for the field of software engineering e. g. in Calero et al. [CMP21].

Research on sustainability in ICT is done under several terms, varied in definition and scope [CP17], e. g. Green IT/ICT, Green by IT/ICT, Sustainable Software, etc. The terms 'Green IT' and 'Green By IT' separate the field into two areas. In this regard, we designate ICT that aims to have a positive impact on sustainability in other areas (e. g. smart grids, building automation, environmental management information systems (EMIS) like life-cycle assessment software (LCA), etc.) as 'Green By IT', whereas 'Green IT' deals with making ICT itself more sustainable (e. g. follow-the-sun load shifting, backward compatible software, designing energy- and resource-efficient hardware, etc.). We consider 'Green Coding' as part of 'Green IT', concerned with the sustainability of the software itself, focusing on methods and tools that can be leveraged during development to impact software sustainability in all of its life-cycle phases (development, usage, and disposal phase).

2 Research Questions and Method

The search and analysis is divided into three research questions (RQ). For RQ1 *"What concepts of programming software in an environmentally friendly way exist in general for software development?"*, Green Coding practices and tools outlined in the academic literature are gathered and presented. This should provide an overview to IT professionals approaching the topic of sustainable software development, aide them to navigate the field, and help them find Green Coding practices and tools applicable to their work. RQ2 *"What concepts of environmentally friendly software engineering processes are already in place in the industry?"* is intended to yield information on Green Coding practices used by software developers. We searched for case studies or white papers on implementing Green Coding practices, and their impact on software performance and sustainability metrics. Methods and tools from industry and community sources are also presented here. RQ3 *"How can Green Coding concepts be implemented into the current software development processes and the curricula of existing study programs?"* explores how Green Coding practices can be communicated to future IT professionals, to establish competence in using available tools and methods. To reduce the complexity of RQ3 and more precisely match the target audiences, RQ3 is divided into the following two sub-questions. RQ3-1 focuses on the software development process, pointing out established software development methods, models, processes, and tools. RQ3-2 focuses on curricula of existing study programs to generate knowledge about teaching approaches for future IT professionals. As most software developers have an academic background, this analysis focuses on implementing aspects of Green IT into curricula of higher education institutions, since over 75 % of software developers attended a college [St22].

The analysis bases on an exploratory literature search using known contributions as well as querying Google Scholar, Digital Bibliography & Library Project (dblp), and the university libraries of the Trier University of Applied Sciences and the Industrial Environmental

Informatics Unit of the HTW Berlin – University of Applied Sciences for the terms 'sustainable software development processes', 'sustainability as non-functional requirement', and 'development of curriculum'. We limited the results to literature published since 2010 and excluded papers referring to sustainability in a different context than defined above (i. e. not focusing on the environmental dimension of sustainability), as well as papers from non-computer science fields (e. g. results about green (building) code). This was decided based on the title, research area, and abstract. We had to exclude some literature that would likely have been applicable, due to access restrictions.

We selected the search strings for the RQs with the intention of acquiring literature summary sources (i. e. previous literature analysis or introductory texts), concrete measures and guidelines, and literature on implementations of green coding practices in software development and education. Keywords to identify case studies, models, best practices, guidelines, processes, and curricula were added to identify the state of the art in Green Coding. The search strings are available in the supplementary material repository[3].

3 Processed Sources and Findings

3.1 RQ1 - What concepts of programming software in an environmentally friendly way exist for software development in general?

Energy consumption at the hardware level is well-understood. However, the impact of software on energy consumption has only been regarded in the past decade and has gained more attention since about 2015. There are models for estimating or measuring the energy consumption of software, and tools based on these models that seem suitable to inform software developers about energy optimization potentials in their code such as [As21; Hi14; Hö13; Ka09; Kh22; Le15]. These exist for a range of technologies. Henderson et al. developed a framework to make the climate impacts of machine-learning research more visible [He20]. Furthermore, hardware-based power measurement tools like RAPL[4] or nvidia-smi[5] exist, that can be used to quickly assess the energy consumption of individual hardware components.

A common theme in many studies is that there rarely is an optimal solution concerning energy- and resource consumption [Pe17]. Some studies like [Ja16] compare hardware-based measurements with simultaneously obtained estimations and find that there are sometimes large (over 60 % in this case) differences in the measurements vs. the estimations due to unexplained energy overheads. This demonstrates the complexity of software sustainability and highlights why IT professionals need to be able to assess which measures are suited for

[3] `https://doi.org/10.5281/zenodo.7920569`

[4] `https://01.org/blogs/2014/running-average-power-limit-%E2%80%93-rapl` and `https://cdrdv2.intel.com/v1/dl/getContent/671200` [2023-04-28]

[5] `https://developer.nvidia.com/nvidia-system-management-interface` [2023-04-28]

a project. The steps required to improve upon a given piece of source code's induced energy consumption are also not clear. Some work provides concrete hints [Hö14] or details where energy hot spots occur in the source code [No15; Ve18; Wa12].

Considering the underlying technologies, Kreten [Kr22] describes methods for energy-efficient container technologies. There is research on software sustainability along the product life-cycle: when designing the software architecture (e. g. [Ch19]), during development (e. g. [Hö14; Wa12]), and maintenance (e. g. [Bu17; Ma21; Sa16]). Most of these approaches rely on energy measurements and a broad conceptual basis, such as [Hi14; Ke13; Na11], who define models, process artifacts and evaluation methods for green software engineering processes, not including specific Green Coding methods such as coding guidelines or using specific architectures and technologies. Several studies focus explicitly on applications for mobile devices (e. g. [Ba22; Hi14; Sa16]), due to the impact of energy consumption on battery life, with Anwar et al. providing an overview of available tools for the development of green android apps [An21]. Advances have been made in the development and application of standards for sustainable software products and eco-labels [Ke18; Na21]. However, no sustainability standards applying explicitly to software development have been identified by standards organizations.

3.2 RQ2 - What concepts of environmentally friendly software engineering processes are already in place in the industry?

Few case studies on software sustainability practices in the industry are available. Calero et al. [Ga21] performed an in-depth analysis of software sustainability on a set of personal health record software products. They propose recommendations and measured their impact on software energy consumption. Heldal et al. [He23] identify a need for sustainability competency in IT professionals in the industry. Lammert et al. [La22] found that software engineers are usually not equipped with the training and tools to fulfill this need. In addition to scientific papers, industry and developer communities have also produced methods and tools for sustainable software engineering, some of which are not (yet) published. These often build on the software measurement approaches mentioned in RQ1.

In addition to the blue angel for software products and the criteria developed in [Ke18] and the proposed measurement setups using external power meters like [Ju22], internal loggers (like the aforementioned RAPL and nvidia-smi or eBPF[6]) in combination with resource loggers can be useful to assess the impacts of a software product under development. Here, it is necessary to access the data provided by the hardware drivers. Exemplary tools are the Green Metrics Tool[7], Code Carbon[8] Hubblo RAPL exporter[9] and Scaphandre[10].

[6] `https://ebpf.io/` [2023-04-28]
[7] `https://github.com/green-coding-berlin/` [2023-05-09]
[8] `https://github.com/mlco2/codecarbon` [2023-05-10]
[9] `https://github.com/hubblo-org/windows-rapl-driver` [2023-05-09]
[10] `https://github.com/hubblo-org/scaphandre` [2023-05-09]

Many available guidelines and best-practice recommendations are in the form of blog posts or lists from developer communities, consulting firms and businesses[11]. Regardless of the correctness of the information contained in this content, they often do not provide sources for their recommendations.

3.3 RQ3-1 How can Green Coding concepts be implemented into the current software development processes?

Heldal et. al. [He23] identify that several factors motivate organizations to act sustainably, primarily beneficial effects on their business. We identified the following approaches to implementing the findings into the software development process: In addition to the cost savings from more efficient hardware and energy use, the reduction of direct and indirect environmental impacts must be incentivized, e. g. through environmental guidelines and environmental regulations. Ultimately, environmental seals, labels, and awards can provide a competitive advantage. For example, an eco-label is a clear advantage for public procurement in Germany, as it can be part of the specification for a product [Sc22]. However, even without regulations, it can be assumed that a green impact will benefit from transparency.

Additionally, we considered where the knowledge could be implemented in companies, leading to the question of how the knowledge is transferred to the corresponding destinations. In this context, training and education are fundamental anchor points for transfer and implementation [Sa21]. The conceptual and technical basics of Green Coding or Green IT, in general, should be conveyed. Training courses and workshops must bring this knowledge into the companies. These training courses can be given or initiated externally by consultants and governmental actors. However, they can also be quickly and widely disseminated, e. g. through e-learning platforms.

For integrating the fundamentals of Green Coding into the software development process, reference models, like the Greensoft Model [Di13], describe ways to implement them into software development processes like Scrum [Na11]. Investing in (the development of) tools and frameworks for implementing more energy-saving code or adapting the software development process is indispensable to implement the path most effectively. Only by making the consumption transparent for developers, improvements are possible.

3.4 RQ3-2 How can Green Coding concepts be implemented into the curricula of existing study programs?

To answer this question and develop recommendations, our first step was to research existing courses and programs of universities and institutes of higher education. However, the

[11] e. g. `https://www.ibm.com/cloud/blog/green-coding`, `https://www.suso.academy/en/2023/03/13/green-coding-the-5-most-important-basics-for-sustainable-software-development-with-code-examples/`, or `https://geekflare.com/green-coding/` [2023-06-27]

search was more challenging than it initially seemed. Most universities have printed the specializations of the study subjects in module handbooks or make the course information with their disciplines available only after an account has been created in an application management system. This influenced our search, which we mainly conducted via online media limited to English publications and announcements. In addition to sources from cooperating universities, we consulted job platforms such as LinkedIn[12] and institutions known in the Green Coding community. Currently, courses designed for GSE or Green Coding are mainly available in Europe. Table 1 lists the universities and universities of applied sciences in Europe and the US, which we found that teach related courses.

School	**Country**	**School**	**Country**
University of Bristol	England	LUT University	Finland
Hasso Plattner Institut (HPI)	Germany	HTW Berlin	Germany
Umwelt-Campus Birkenfeld (UCB)	Germany	Amsterdam School of Data Science	Netherlands
DELFT University of Technology	Netherlands	Kharkiv Aviation Institute*	Ukraine
Royal Institute of Technology (KTH)	Sweden	Mälardalen University	Sweden
University of Gothenburg	Sweden	Carnegie Mellon University	USA
The University of California Berkeley	USA		

*current status unknown

Tab. 1: Universities teaching Green Coding and GSE

To be able to give recommendations for the implementation of Green Coding and GSE, in addition to the search terms listed in the supplementary material, search terms for the creation of curricula are added to the queries.

For future curricula, Drake and Reid [DR18] propose to, on the one hand, teach the big picture of a topic and, on the other hand, the main competencies. To convey this knowledge, technical, practical, and participatory interests should be addressed [FB06]. Mishra and Mishra [MM21] describe how a sustainable software engineering curriculum based on ACM/IEEE could be created. Since it is challenging to design entire curricula such as PERCCOM [Ko19], the first step should be to develop individual courses, workshops, discussion groups [MM11] or university initiatives like in the HAW Hamburg [Ei23]. This can be followed by a module like 'Software Engineering Sustainability', introduced at the Kharkiv Aviation Institute [TV19]. To identify the right target group for these modules, it should be attended by various similar courses because interdisciplinary knowledge is more important than ever in current times [Ra18]. To implement the new knowledge, the Industrial Environmental Informatics Unit at HTW Berlin developed and evaluated a prototypical course that includes the listed points to identify best practices and obstacles, conducted in the winter semester of 2022/23. As a part and also a by-product of the course and the measurement setup [Ju22] used for this purpose, the acquired knowledge was already passed on to students and colleagues. Thus, publications on the latest statistical analysis applications [Se23] and also concerning current AI environments in education [Bü23]. The contents and modalities are currently being published in [Ju23].

[12] `https://www.linkedin.com/` [2023-05-09]

4 Conclusion and Outlook

In this literature analysis, we collected results to classify the state of the art of Green Coding in software engineering and education. The goals were to enable stakeholders to realize the potential of Green Coding, whether well-researched or untapped, to provide them with a solid introduction to the field and possible approaches for their activities. A decade ago, the area of *Sustainable Software Engineering* was still in its infancy, as evidenced by a lack of case studies and evaluations [Pe14]. Today, we conclude that research in the field is steadily increasing and has made significant progress. Several measurement and evaluation systems for assessing software's energy- and resource consumption and -efficiency were developed. However, the low adoption rate of GSE methods still stands, as we could identify only a few case studies. Implementing aspects of sustainability, such as Green Coding techniques, into the training of future IT professionals is a crucial step to bringing sustainability into view for software producers while providing them with a workforce capable of implementing sustainability into their development process. Future work could be done by extracting practices contained in non-academic sources and finding academic papers supporting their impact on sustainability or attempting to prove some of the recommendations experimentally by applying them to a software product and tracking the impact of each recommendation. Surveys and expert interviews should be conducted to get a better overview of green coding in institutes of higher education and industry, as curricula may not be up-to-date and corporations do not always publish the results of their projects.

A logical consequence of this project is that this knowledge is listed and communicated in a way that is as easy to understand as possible. Besides already existing courses, like the course from the green software foundation [13], it is essential that these concepts are scientifically validated and distributed. Since the expertise in imparting knowledge lies with the institutes of higher education, the design for the course concept to impart this knowledge and practice needs to be expanded. Furthermore, we are currently extending the GSE approach to Artificial Intelligence and also planning to focus on a case study about the conception of a prototype measurement approach to fully virtualized AI laboratories.

Acknowledgements

This work was funded by the Internet Society Foundation[14] project "Potentials of Green Coding". We would also like to thank our partner, the German Informatics Society (Gesellschaft für Informatik e. V.), especially Elisabeth Schauermann and Carolin Henze[15], the assisting master students, and student assistants.

[13] `https://learn.greensoftware.foundation/` [2023-05-09]
[14] `https://www.isocfoundation.org/` [2023-05-09]
[15] `https://gi.de/en/aktuelles/projekte/en-green-coding` [2023-05-09]

References

[An21] Anwar, H. et al.: Tool Support for Green Android Development. In: Software Sustainability. Springer, pp. 153–182, 2021.

[An22] Andrikopoulos, V. et al.: Sustainability in Software Architecture: A Systematic Mapping Study, 2022, URL: https://arxiv.org/abs/2204.11657.

[As21] Aslanpour, M. S. et al.: WattEdge: A Holistic Approach for Empirical Energy Measurements in Edge Computing. In: Service-Oriented Computing. Springer, Cham, pp. 531–547, 2021.

[Ba22] Bangash, A. A. et al.: A Black Box Technique to Reduce Energy Consumption of Android Apps. In: 44th International Conference on Software Engineering: New Ideas and Emerging Results. ACM, Pittsburgh, Pennsylvania, pp. 1–5, 2022.

[Bu17] Bunse, C.: On the Impact of Code Obfuscation to Software Energy Consumption. In: From Science to Society, New Trends in Environmental Informatics - EnviroInfo 2017, Luxembourg, 13-15September 2017. Springer, pp. 239–249, 2017, URL: https://doi.org/10.1007/978-3-319-65687-8%5C_21.

[Bü23] Bültemann, M. et al.: Energy Consumption of AI in Education: A Case Study. In: 21. Fachtagung Bildungstechnologien (DELFI). Gesellschaft für Informatik e.V., Bonn, 2023.

[Ch19] Chowdhury, S. A. et al.: GreenBundle: An Empirical Study on the Energy Impact of Bundled Processing. In: 41st International Conference on Software Engineering. 2019.

[CMP21] Calero, C.; Moraga, M. Á.; Piattini, M.: Introduction to software sustainability. Software Sustainability/, pp. 1–15, 2021.

[CP17] Calero, C.; Piattini, M.: Puzzling out Software Sustainability. Sustainable Computing: Informatics and Systems 16/, pp. 117–124, 2017, ISSN: 2210-5379, URL: https://www.sciencedirect.com/science/article/pii/S2210537916301676.

[Di13] Dick, M. et al.: Green software engineering with agile methods. In: 2013 2nd International Workshop on Green and Sustainable Software (GREENS). Pp. 78–85, 2013.

[DR18] Drake, S.; Reid, J.: Integrated Curriculum as an Effective Way to Teach 21st Century Capabilities. Asia Pacific Jorunal of Educational Research 1/, pp. 31–50, Jan. 2018.

[Ei23] Eickstädt, E. et al.: Computer Science for Future - Sustainability and Climate Protection in the Computer Science Courses of the HAW Hamburg, 2023.

[FB06] Fraser, S. P.; Bosanquet, A. M.: The curriculum? That's just a unit outline, isn't it? Studies in Higher Education 31/3, pp. 269–284, 2006.

[Ga21] García-Berná, J. A. et al.: Energy efficiency in software: A case study on sustainability in personal health records. Journal of Cleaner Production 282/, p. 124262, 2021.

[Gu17] Guldner, A. et al.: Energy Consumption and Hardware Utilization of Standard Software: Methods and Measurements for SoftwareSustainability. In: Progress in IS. Springer, Aug. 2017.

[He20] Henderson, P. et al.: Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning. Journal of Machine Learning Research/, Arxiv: 2002.05651, Jan. 2020.

[He23] Heldal, R. et al.: Sustainability Competencies and Skills in Software Engineering: An Industry Perspective, 2023.

[Hi14] Hindle, A. et al.: GreenMiner: A Hardware Based Mining Software Repositories Software Energy Consumption Framework. In: Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, New York, USA, pp. 12–21, 2014.

[Hö13] Hönig, T. et al.: Proactive Energy-Aware System Software Design with SEEP. Softwaretechnik-Trends 33/2, pp. 6–7, 2013.

[Hö14] Hönig, T. et al.: Proactive Energy-Aware Programming with PEEK. In: 2014 International Conference on Timely Results in Operating Systems. USENIX, 2014.

[Ja16] Jagroep, E. A. et al.: Software Energy Profiling: Comparing Releases of a Software Product. In: 38th International Conference on Software Engineering Companion. 2016.

[Ju22] Junger, D. et al.: Conception and test of a measuring station for the analysis of the resource and energy consumption of material flow-oriented environmental management information systems (EMIS). In: EnviroInfo 2022. Gesellschaft für Informatik e.V., 2022.

[Ju23] Junger, D. et al.: Design and implementation of a lecture for teaching current Green Coding approaches and practices at the HTW Berlin. In: Informatik 2023. accepted for publication, Gesellschaft für Informatik e.V., Bonn, 2023.

[Ka09] Kansal, A. et al.: Joulemeter: Virtual Machine Power Measurement and Management, 2009, URL: `https://www.microsoft.com/en-us/research/project/joulemeter-computational-energy-measurement-and-optimization/`.

[Ke13] Kern, E. et al.: Green software and green software engineering–definitions, measurements, and quality aspects. In: First International Conference on Information and Communication Technologies for Sustainability (ICT4S2013), 2013b ETH Zurich. Pp. 87–91, 2013.

[Ke18] Kern, E. et al.: Sustainable software products—Towards assessment criteria for resource and energy efficiency. Future Generation Computer Systems/, pp. 199–210, 2018.

[Kh22] Khandelwal, S. et al.: Real Time Carbon Emissions Calculator for Personal Computers. In: Data Management, Analytics and Innovation. Springer, Singapore, pp. 651–663, 2022.

[Ko19] Kor, A.-L. et al.: Education in Green ICT and Control of Smart Systems : A First Hand Experience from the International PERCCOM Masters Programme. IFAC-PapersOnLine 52/9, 12th IFAC Symposium on Advances in Control Education ACE 2019, pp. 1–8, 2019, ISSN: 2405-8963, URL: https://www.sciencedirect.com/science/article/pii/S2405896319304495.

[Kr22] Kreten, S.: Modellbildung und Umsetzung von Methoden zur energieeffizienten Nutzung von Containertechnologien, PhD thesis, Trier University, 2022.

[La22] Lammert, D. et al.: Software Engineers in Transition: Self-Role Attribution and Awareness for Sustainability. In: Proceedings of the 55. Hawaii International Conference on System Sciences. Jan. 2022.

[Le15] LeBeane, M. et al.: Watt Watcher: Fine-Grained Power Estimation for Emerging Workloads. In: International Symposium on Computer Architecture and High Performance Computing. IEEE, 2015.

[Ma21] Mancebo, J. et al.: Does maintainability relate to the energy consumption of software? A case study. Software Quality Journal 29/1, pp. 101–127, 2021.

[MM11] Malik, A. S.; Malik, R. H.: Twelve tips for developing an integrated curriculum. Medical teacher 33/2, pp. 99–104, 2011.

[MM21] Mishra, A.; Mishra, D.: Sustainable Software Engineering: Curriculum Development Based on ACM/IEEE Guidelines. In: Software Sustainability. Springer, Cham, 2021.

[Na11] Naumann, S. et al.: The greensoft model: A reference model for green and sustainable software and its engineering. Sustainable Computing: Informatics and Systems 1/4, pp. 294–304, 2011.

[Na21] Naumann, S. et al.: The eco-label blue angel for software - Development and components. In: Advances and New Trends in Environmental Informatics: Digital Twins for Sustainability. Springer, pp. 79–89, 2021.

[No15] Noureddine, A. et al.: Monitoring energy hotspots in software. Automated Software Engineering 22/3, pp. 291–332, Oct. 2015.

[Pe14] Penzenstadler, B. et al.: Systematic mapping Study on Software Engineering for Sustainability (SE4S). ACM International Conference Proceeding Series/, May 2014.

[Pe17] Pereira, R. et al.: Energy Efficiency across Programming Languages: How Do Energy, Time, and Memory Relate? In: SIGPLAN International Conference on Software Language Engineering. ACM, Vancouver, Canada, pp. 256–267, 2017.

[Ra18] Ramirez-Mendoza et al.: Engineering Education 4.0: proposal for a new Curricula. In: IEEE Global Engineering Education Conference (EDUCON). Pp. 1273–1282, 2018.

[Sa16] Sahin, C. et al.: How does code obfuscation impact energy usage? Journal of Software: Evolution and Process/, Jan. 2016.

[Sa21] Saraiva, J. et al.: Bringing Green Software to Computer Science Curriculum: Perspectives from Researchers and Educators. In: ITICSE. ACM, 2021.

[Sc22] Schneider, T.: Rechtsgutachten umweltfreundliche öffentliche Beschaffung - Aktualisierung 2022, Accessed: 2023-04-28, 2022.

[Se23] Seegert, T. et al.: Energy and resource comparison of current applications with a focus on statistical analyses and evaluations using the example of Matlab and R/RStudio. In: EnviroInfo 2023. accepted for publication, Gesellschaft für Informatik e.V., Bonn, 2023.

[St22] Statista: Worldwide levels of formal education for software developers, `https://www.statista.com/statistics/793568` [2/24/2023], 2022.

[TV19] Turkin, I.; Vykhodets, Y.: Software Engineering Sustainability Education in Compliance with Industrial Standards and Green IT Concept. In: Green IT Engineering: Social, Business and Industrial Applications. Springer, pp. 579–604, 2019.

[Ve18] Verdecchia, R. et al.: Code-Level Energy Hotspot Localization via Naive Spectrum Based Testing. In: Advances in Environmental Informatics: Managing Disruption, Big Data and Open Science. Springer, 2018.

[Wa12] Wang, S. et al.: Safari: Function-level power analysis using automatic instrumentation. In: 2012 International Conference on Energy Aware Computing. IEEE, pp. 1–6, 2012.

[Wi95] Wirth, N.: A Plea for Lean Software. Computer 28/02, pp. 64–68, 1995.