Leveraging Artificial Intelligence for Sustainable Software Maintenance: A Case Study Approach

Chairul Rizal[1*], Erni Marlina Saari[2]
[1]Universitas Pembangunan Panca Budi, Medan, Indonesia
[2]Universiti Pendidikan Sultan Idris, Tanjong Malim-Perak Darul Rizduan, Malaysia

Keywords:
Artificial Intelligence; Sustainable Software Engineering; AI-Driven Software Optimization; Green Computing and Software Sustainability; Case Study.
.

***Correspondence Address:**
chairulrizal@dosen.pancabudi.ac.id

Abstract: In the era of rapid technological development, software maintenance has become a major challenge, especially regarding efficiency and sustainability. Many companies need help in effectively managing the software lifecycle while considering the environmental impact and resources used. This research explores how artificial intelligence (AI) can be utilised to improve sustainability in the software maintenance process. Through a case study approach, this research examines the implementation of AI in software maintenance in several technology organisations. The research methodology combines qualitative and quantitative approaches, where data is collected through in-depth interviews, observations, and document analyses, as well as efficiency measurements through the AI algorithms used. The results showed that the application of AI in software maintenance not only improved efficiency in identifying and fixing bugs but also significantly reduced energy consumption and computing resource usage. The case study also revealed that AI can help predict maintenance needs proactively, thereby reducing the frequency of human intervention that requires more energy. Thus, this research concludes that the integration of artificial intelligence in software maintenance makes a positive contribution to sustainability, both in economic and environmental terms. The recommendation for software developers is to further adopt AI technologies in the maintenance process to improve long-term operational sustainability.

INTRODUCTION

In the rapidly evolving digital age, software has become the foundation for many industry sectors, government and everyday life. However, behind these advancements come significant challenges related to the environmental and social impacts of the technologies we use. In this context, sustainability in software development is becoming an increasingly pressing issue. Sustainability in software development encompasses efforts to minimise the negative impacts of software on the environment and society, as well as ensuring that software can operate efficiently and survive in the long term. This involves efficient use of energy resources, reduced carbon emissions, wiser management of the software lifecycle, and ensuring that the software is inclusive and supports social justice.

Amid growing global awareness of the climate crisis, the information technology sector is under the spotlight for its contribution to global carbon emissions. With the growing number of cloud-based applications, energy-intensive data centres, and IoT devices, the software industry needs to adopt practices that support sustainability. Sustainability is not only a moral responsibility but also a strategy to maintain business competitiveness in a market that increasingly demands green practices.

To achieve sustainable software development, it is important for education in this field to adapt and integrate sustainability concepts into its curriculum. Education plays a key role in shaping the mindset and skills of future generations of software developers. Without a strong understanding of the importance of sustainability, graduates may not be prepared to face increasingly complex environmental and social challenges.

Integrating sustainability concepts in software development education teaches students about environmentally friendly development techniques and instils ethical values and social responsibility. This includes an understanding of the software lifecycle, from design, and implementation, to maintenance and disposal, as well as the long-term impact of software on society and the environment.

In addition, education that integrates sustainability can foster innovation in software development, by inspiring students to seek more efficient and environmentally friendly solutions. It also opens up new opportunities in the job market, where skills in sustainable software development are increasingly valued by industries moving towards a green economy.

The main problem underlying this research is the lack of attention to the concept of sustainability in software development education. Although there is a growing awareness of the importance of sustainability in the technology industry, the integration of sustainability principles into the software development education curriculum is still limited. Many current education programmes focus more on the technical aspects and functionality of software without regard to wider environmental and social impacts. As a result of this lack of focus on sustainability, graduates of software development programmes are often not equipped with the necessary knowledge and skills to develop environmentally and socially sustainable software. This can lead to the development of software that is energy inefficient, generates large amounts of e-waste, and exacerbates social inequities in access to technology. This problem statement confirms that there is an urgent need to integrate the concept of sustainability into software development education. As such, this research will focus on how to design an educational approach that not only teaches technical skills but also equips students with a deep understanding of the importance of sustainability in software development.

Sustainability in software development refers to practices and principles that ensure that the development process is not only efficient and effective but also environmentally, socially and economically responsible. In this context, there are several key principles underlying sustainability, including the integration of environmentally friendly practices, inclusive development, and the application of methodologies that support continuous improvement.

One of the key principles of sustainability is the application of a process improvement-orientated software development methodology. The Capability Maturity Model Integration (CMMI) model is one approach that can be used to improve software development processes in organizations, including small companies that often face challenges in development management (Heristian & Erawati, 2019). CMMI helps organizations to assess and improve their development practices, to produce higher quality and sustainable software (Widodo, 2016). In addition, methodologies such as DevOps and Continuous Integration/Continuous Deployment (CI/CD) also contribute to sustainability by improving collaboration between teams and accelerating development cycles, which in turn reduces waste and improves responsiveness to user needs (Taju, 2023; Toba et al., 2022).

The principle of sustainability also includes security aspects in software development. Secure Software Development Life Cycle (SDLC) is an approach that integrates security into every phase of software development, from analysis to maintenance (Hasan et al., 2021). By priority security, developers can prevent vulnerabilities that could harm users and the environment, thus supporting the long-term sustainability of the resulting software products.

In addition, sustainability in software development is also related to measuring the quality and effectiveness of the developed software. Methods such as Function Point Analysis (FPA) can be used to evaluate the complexity and quality of the software, which is important to ensure that the final product meets the set standards and can function properly in the long run (Parlika, 2023; Wicaksono et al., 2021). By taking proper measurements, organizations can identify areas that need improvement and implement the necessary changes to improve the sustainability of their products.

Finally, sustainability in software development also involves the involvement of stakeholders and users in the development process. An approach that involves feedback from users and collaboration with various parties can ensure that the software developed not only meets technical needs but also considers social and environmental impacts (Perwitasari & Irwansyah, 2021). Thus, sustainability in software development does not only focus on technical aspects but also broader social and environmental responsibilities.

One widely used approach is Agile, which emphasises flexibility and collaboration in software development. Agile approaches, including Extreme Programming (XP), have proven to be effective in coping with rapid and unexpected changes in requirements, which often occur in software development projects (Dewi et al., 2018; Gunawan et al., 2020). In an educational context, the application of Agile Project Management to the development of real projects, such as the e-musrenbang system, demonstrates how students can learn to adapt to change and work in dynamic teams (Dewi et al., 2018). In addition, the use of XP in web-based application development also gives students hands-on experience in applying Agile principles, such as continuous testing and iterative development (Gunawan et al., 2020; Supriyatna, 2018).

On the other hand, traditional approaches such as Waterfall are still used in educational contexts, especially for projects that have clear and stable specifications. This method offers a more organized and easy-to-understand structure, making it suitable for students who are just starting in software development (Mahendri, 2023). However, this approach is often considered less flexible than Agile, especially in the face of rapidly changing requirements (Bolung & Tampangela, 2017).

In addition, model-based approaches, such as Software Development Life Cycle (SDLC) and prototyping, are also important in software development education. The

SDLC model provides a systematic framework for software development, which helps students understand each phase in the development process (Paksi et al., 2023; Budi et al., 2017). The use of prototype models in software development also allows students to better capture user needs through direct interaction and rapid feedback (Perwitasari & Irwansyah, 2021). This is particularly important in an educational context, where an understanding of user needs can improve the quality of the final product.

The importance of software quality measurement is also a focus in software development education. Methods such as Function Point Analysis are used to evaluate and measure software quality, which helps students understand the importance of quality in software development (Parlika, 2023). By understanding how to measure and evaluate software, students can be better prepared to face challenges in the professional world.

Finally, knowledge-based approaches, such as the use of the Analytic Hierarchy Process (AHP) in software application selection, demonstrate the importance of data-driven decision-making in software development education (Setiadi, 2023). This approach trains students to analyze and evaluate various options before making a decision, which is an essential skill in the software industry.

One relevant theory is the Theory of Constructivism, which emphasises that knowledge is built through experience and social interaction. In the context of software development, this approach can be applied by encouraging students to engage in real projects that focus on sustainability, such as the development of applications that support natural resource management or applications that promote environmental awareness. In this way, students not only learn about technology but also understand the social and environmental impacts of the products they develop.

The main objective of this research is to develop an educational approach that integrates sustainability principles.

RESEARCH METHODS

This research uses a mixed methods approach, which combines elements from both qualitative and quantitative research. This approach was chosen to use the strengths of both methods in identifying the needs, challenges and opportunities in sustainability integration in software development education. This is a phase of the study that can be seen in Figure 1 below.

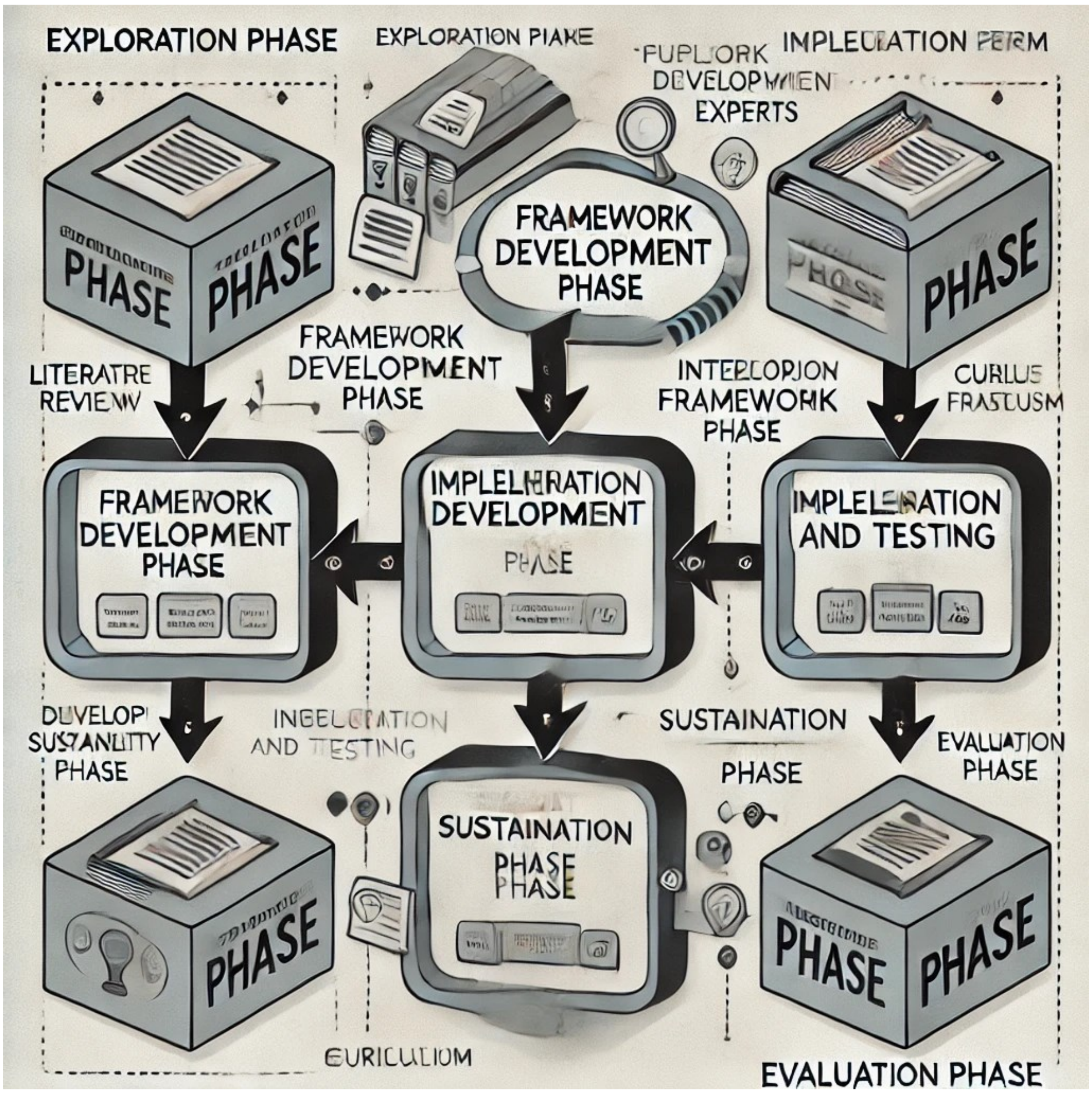

Figure 1. Research Phase

Exploration Phase:

- Collected literature related to sustainability in software development and education.
- Conduct expert interviews and analyse existing software development curricula to identify gaps and needs.

Framework Development Stage:

- Based on the findings from the exploratory stage, develop a framework that integrates sustainability into the software development curriculum.
- This framework will include teaching methodologies, teaching materials, and evaluation

strategies.

Implementation and Pilot Phase:

- Implement the proposed framework in several educational institutions as a pilot project.
- Collect data through surveys and observations during the implementation process.

Evaluation Stage:

- Analyse the data collected from the implementation to evaluate the effectiveness of the framework.
- Provide recommendations based on the research findings for further improvement and adoption.

Validity and Reliability

To ensure the validity and reliability of the research results:

- Triangulation: Using multiple data collection methods (interviews, surveys, document analysis) to confirm findings.
- Validity Testing: Using content and construct validity tests to ensure that the research instruments measure what they are supposed to measure.
- Reliability Testing: Conducting reliability tests to ensure consistency of results from the research instruments used.

RESULTS AND DISCUSSION

The findings from the literature review indicate a growing recognition of the importance of sustainability in software engineering education. However, significant challenges remain in effectively integrating these principles into existing curricula. One of the primary barriers is the lack of awareness and understanding of sustainability concepts among educators and students alike (Chitchyan et al., 2016). Many software engineering programs continue to prioritize technical skills over sustainability considerations, leading to a disconnect between education and the realities of sustainable software development.

To address these challenges, several strategies have been proposed. First, incorporating sustainability topics into core software engineering courses can raise awareness and foster a culture of sustainability among students (Penzenstadler & Fleischmann, 2011; Cai, 2010). This can be achieved through the development of new course modules that focus on sustainable design principles, green software practices, and the ethical implications of software development. For instance, integrating case studies that highlight successful sustainable software projects can provide students with practical examples of how sustainability can be achieved in real-world scenarios (Palacin-Silva et al., 2018; Penzenstadler et al., 2012).

Second, project-based learning (PBL) approaches can be employed to engage students in hands-on sustainability projects. PBL encourages students to work collaboratively on real-world problems, allowing them to apply theoretical knowledge to practical situations (Bielefeldt, 2013). By focusing on sustainability challenges, such as energy-efficient software design or the development of socially responsible applications, students can develop critical thinking and problem-solving skills essential for future software engineers (Penzenstadler, 2013).

Moreover, partnerships with industry stakeholders can enhance the relevance of sustainability education. Collaborating with companies that prioritize sustainable practices can provide students with exposure to current industry trends and challenges. Such partnerships can also facilitate internships and co-op programs that allow students to gain practical experience in sustainable software development (Seyff et al., 2021; Naumann et al., 2014).

The literature also emphasizes the importance of continuous feedback and iterative learning in sustainability education. Agile methodologies, which prioritize customer-centric approaches and frequent feedback loops, can be adapted to enhance sustainability practices in software development (Sriraman & Raghunathan, 2023; Rashid & Khan, 2018). By incorporating sustainability metrics into Agile practices, educators can help students understand the impact of their decisions on resource utilization and environmental outcomes.

Furthermore, the integration of sustainability into software engineering education should not be limited to technical skills alone. It is essential to cultivate a mindset that values sustainability as a core principle of software development. This can be achieved through interdisciplinary approaches that incorporate insights from environmental science, social sciences, and ethics into the software engineering curriculum (Nazir et al., 2020; Becker et al., 2015).

CONCLUSION

The integration of sustainability into software development education is not merely an option but a necessity in preparing the next generation of developers. As the software industry continues to evolve, educational institutions must adapt their curricula to include sustainability principles that address the environmental, social, and economic dimensions of software engineering. The findings from the literature review underscore the importance of raising awareness, developing practical strategies, and fostering partnerships with industry stakeholders to facilitate this integration. By implementing project-based learning, interdisciplinary approaches, and Agile methodologies, educators can equip students with the skills and mindset necessary to create sustainable software solutions. Ultimately, the goal is to cultivate a generation of software engineers who are technically proficient, socially responsible, and environmentally conscious. Through these efforts, the software industry can contribute to a more sustainable future, ensuring that technology serves as a force for good in society.

REFERENCE

Hasan, M., Suhermanto, S., & Suharmanto, S. (2021). Keamanan sistem perangkat lunak dengan secure software development lifecycle. Jurnal Ilmu Komputer Dan Bisnis, 12(1), 88-101. https://doi.org/10.47927/jikb.v12i1.95

Heristian, S. and Erawati, W. (2019). Systematic literature review of software process improvement models in a small company. Cess (Journal of Computer Engineering System and Science), 4(2), 125. https://doi.org/10.24114/cess.v4i2.12695

Parlika, R. (2023). Pengukuran kualitas perangkat lunak website pendataan ekskul siswa menggunakan function point. Jurnal Ilmiah Informatika, 11(01), 1-14. https://doi.org/10.33884/jif.v11i01.5578

Perwitasari, A. and Irwansyah, M. (2021). Model prototipe dan analisis use case pada rekayasa kebutuhan perangkat lunak pengajuan dokumen kependudukan. Jurnal Edukasi Dan Penelitian Informatika (Jepin), 7(2), 175. https://doi.org/10.26418/jp.v7i2.47976

Taju, S. (2023). Mengakselerasi keterampilan rekayasa perangkat lunak: peranan devops, sdlc, dan ci/cd dalam meningkatkan kompetensi siswa smk n 1 pusomaen. SSJ, 2(1), 119-128. https://doi.org/10.31154/servitium.v2i1.24

Toba, H., Gautama, T., Narabel, J., Widjaja, A., & Sujadi, S. (2022). Evaluasi metodologi ci/cd untuk pengembangan perangkat lunak dalam perkuliahan. Jurnal Edukasi Dan Penelitian Informatika (Jepin), 8(2), 227. https://doi.org/10.26418/jp.v8i2.51992

Wicaksono, S., Valentina, I., Ekadana, F., & Chandra, M. (2021). Pengukuran kualitas perangkat lunak menggunakan function point analysis (studi kasus: fishbowl). Decode Jurnal Pendidikan Teknologi Informasi, 1(2), 43-49. https://doi.org/10.51454/decode.v1i2.8

Widodo, W. (2016). Evaluasi proses pengembangan perangkat lunak pada virtual team development menggunakan cmmi versi 1.3. Jurnal Informatika, 10(1). https://doi.org/10.26555/jifo.v10i1.a3345

Bolung, M. and Tampangela, H. (2017). Analisa penggunaan metodologi pengembangan perangkat lunak. Jurnal Eltikom, 1(1), 1-10. https://doi.org/10.31961/eltikom.v1i1.1

Budi, D., Siswa, T., & Abijono, H. (2017). Analisis pemilihan penerapan proyek metodologi pengembangan rekayasa perangkat lunak. Teknika, 5(1), 24-31. https://doi.org/10.34148/teknika.v5i1.48

Dewi, K., Ciptayani, P., & Wijaya, I. (2018). Agile project management pada pengembangan e-musrenbang kelurahan benoa bali. Jurnal Teknologi Informasi Dan Ilmu Komputer, 5(6), 723-730. https://doi.org/10.25126/jtiik.2018561143

Gunawan, R., Napianto, R., Borman, R., & Hanifah, I. (2020). Penerapan pengembangan sistem extreme programming pada aplikasi pencarian dokter spesialis di bandarlampung berbasis android. Format Jurnal Ilmiah Teknik Informatika, 8(2), 148. https://doi.org/10.22441/format.2019.v8.i2.008

Mahendri, R. (2023). Penerapan teknologi single page application (spa) pada aplikasi lelang barang secondhand berbasis website. Voteteknika (Vocational Teknik Elektronika Dan Informatika), 11(3), 240. https://doi.org/10.24036/voteteknika.v11i3.122337

Paksi, A., Hafidhoh, N., & Bimonugroho, S. (2023). Perbandingan model pengembangan perangkat lunak untuk proyek tugas akhir program vokasi. Jurnal Masyarakat

Informatika, 14(1), 70-79. https://doi.org/10.14710/jmasif.14.1.52752

Parlika, R. (2023). Pengukuran kualitas perangkat lunak website pendataan ekskul siswa menggunakan function point. Jurnal Ilmiah Informatika, 11(01), 1-14. https://doi.org/10.33884/jif.v11i01.5578

Perwitasari, A. and Irwansyah, M. (2021). Model prototipe dan analisis use case pada rekayasa kebutuhan perangkat lunak pengajuan dokumen kependudukan. Jurnal Edukasi Dan Penelitian Informatika (Jepin), 7(2), 175. https://doi.org/10.26418/jp.v7i2.47976

Setiadi, T. (2023). Penerapan analytic hierarchy process dengan metode penilaian tertimbang menggabungkan sistem hybrid knowledge based untuk pemilihan aplikasi. Jurnal Ilmiah Sistem Informasi, 2(3), 27-36. https://doi.org/10.51903/juisi.v2i3.794

Supriyatna, A. (2018). Metode extreme programming pada pembangunan web aplikasi seleksi peserta pelatihan kerja. Jurnal Teknik Informatika, 11(1), 1-18. https://doi.org/10.15408/jti.v11i1.6628

Chitchyan, R., Becker, C., Betz, S., Duboc, L., Penzenstadler, B., Seyff, N., … & Venters, C. C. (2016). Sustainability design in requirements engineering. Proceedings of the 38th International Conference on Software Engineering Companion. https://doi.org/10.1145/2889160.2889217

Penzenstadler, B. and Fleischmann, A. (2011). Teach sustainability in software engineering?. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&amp;T). https://doi.org/10.1109/cseet.2011.5876124

Cai, Y. (2010). Integrating sustainability into undergraduate computing education. Proceedings of the 41st ACM Technical Symposium on Computer Science Education. https://doi.org/10.1145/1734263.1734439

Palacin-Silva, M., Ahmed, S., & Porras, J. (2018). Infusing sustainability into software engineering education: lessons learned from capstone projects. Journal of Cleaner Production, 172, 4338-4347. https://doi.org/10.1016/j.jclepro.2017.06.078

Penzenstadler, B., Bauer, V., Calero, C., & Franch, X. (2012). Sustainability in software engineering: a systematic literature review. 16th International Conference on Evaluation &Amp; Assessment in Software Engineering (EASE 2012). https://doi.org/10.1049/ic.2012.0004

Bielefeldt, A. R. (2013). Pedagogies to achieve sustainability learning outcomes in civil and environmental engineering students. Sustainability, 5(10), 4479-4501. https://doi.org/10.3390/su5104479

Penzenstadler, B. (2013). Towards a definition of sustainability in and for software engineering. Proceedings of the 28th Annual ACM Symposium on Applied Computing. https://doi.org/10.1145/2480362.2480585

Seyff, N., Penzenstadler, B., Betz, S., Brooks, I., Oyedeji, S., Porras, J., … & Venters, C. C. (2021). The elephant in the room - educating practitioners on software development for sustainability. 2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS). https://doi.org/10.1109/bokss52540.2021.00017

Naumann, S., Kern, E., Dick, M., & Johann, T. (2014). Sustainable software engineering: process and quality models, life cycle, and social aspects. Advances in Intelligent Systems and Computing, 191-205. https://doi.org/10.1007/978-3-319-09228-7_11

Sriraman, G. and Raghunathan, S. (2023). A systems thinking approach to improve sustainability in software engineering—a grounded capability maturity framework. Sustainability, 15(11), 8766. https://doi.org/10.3390/su15118766

Rashid, N. and Khan, S. U. (2018). Agile practices for global software development vendors in the development of green and sustainable software. Journal of Software: Evolution and Process, 30(10). https://doi.org/10.1002/smr.1964

Nazir, S., Fatima, N., Chuprat, S., Sarkan, H. M., Nurulhuda, F., & Sjarif, N. N. A. (2020). Sustainable software engineering:a perspective of individual sustainability. International Journal on Advanced Science, Engineering and Information Technology, 10(2), 676-683. https://doi.org/10.18517/ijaseit.10.2.10190

Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N., … & Venters, C. C. (2015). Sustainability design and software: the Karlskrona manifesto. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. https://doi.org/10.1109/icse.2015.179