

Towards a Green Quotient for Software Projects

Rohit Mehra[†], Vibhu Saujanya Sharma[†], Vikrant Kaulgud[†], Sanjay Podder[‡], Adam P. Burden^{*}

[†]Accenture Labs, India

[‡]Accenture, India

^{*}Accenture, USA

{rohit.a.mehra,vibhu.sharma,vikrant.kaulgud,sanjay.podder,adam.p.burden}@accenture.com

ABSTRACT

As sustainability takes center stage across businesses, green and energy-efficient choices are more crucial than ever. While it is becoming increasingly evident that software and the software industry are substantial and rapidly evolving contributors to carbon emissions, there is a dearth of approaches to create actionable awareness about this during the software development lifecycle (SDLC). *Can software teams comprehend how green are their projects?* Here we provide an industry perspective on why this is a challenging and worthy problem that needs to be addressed. We also outline an approach to quickly gauge the “greenness” of a software project based on the choices made across different SDLC dimensions and present the initial encouraging feedback this approach has received.

KEYWORDS

Sustainable Software Engineering, Green Software, Carbon Emissions, Software Metrics

1 INTRODUCTION

Multiple studies have estimated that the internet and communications technology industry currently accounts for 2-7% of the global greenhouse gas emissions, and is expected to increase to 14% by 2040¹. These rapidly growing emissions can have a disastrous impact on the sustainability of our environment. A major share of these emissions can be attributed to the design, development, and distribution of software systems and the corresponding infrastructure required to run them. The software development process itself, and the choices/decisions that a project team makes during a typical SDLC, can have a major impact on the greenness (energy consumption and carbon emissions) of the software system.

Although there exists some body of knowledge on software sustainability, it is often overlooked and hence, its industry adoption is still limited [2]. For example, multiple recent studies concerning professional software developers have reported software quality, bug resolution, effective collaboration, etc. as their top priorities, but the lack of sustainability-related questions and priorities poses a major cause for concern [3, 8]. In our experience, we believe that this limited adoption can be attributed to the following challenges:

- **Lack of Awareness:** Despite being a first-class non-functional requirement, software sustainability is under-represented in the current software engineering curriculum [9]. While the focus is predominantly on software performance, quality, security, usability, etc., software sustainability is rarely taught as an area. Moreover, we are witnessing a dearth of such courses/trainings in the open-source community as well. This acts as a major challenge for an environmentally inclined practitioner (developer, tester, program manager,

etc.) who wants to adopt sustainable practices in her project, but is unaware of a starting point.

- **Intrusive Approaches:** Existing approaches require in-depth access to software code, project artifacts, deployment environment, etc., to gather relevant information and suggest optimizations. This intrusive approach becomes a challenge for organizations due to adherence to multiple security, privacy, and compliance-related protocols, and hence acts as an initial barrier to adoption. The problem further amplifies for a software services organization where these artifacts are predominantly owned by the clients.
- **Siloed Focus Areas:** The majority of research in this area focuses on a very specific/isolated part of the larger problem, without usually considering respective tradeoffs (impact on other parts of the project). For example, studying the effects of design patterns on energy consumption [7]. In some cases, introducing a design pattern might increase the energy consumption of the underlying code, but removing it might have a negative impact on other software engineering dimensions like maintainability, reusability, etc. Moreover, these existing approaches tend to focus more on the technological aspects of the software project, leaving behind other project aspects such as process, team, collaboration, etc., which are equally responsible for generating emissions. This restricts the practitioner from gauging a holistic 360° view of the sustainability of a software project - *as a whole*.
- **Lack of Green Metrics in Practice:** In the current literature, there is a dearth of standardized metrics to evaluate/predict the greenness of a particular software system, especially during the design phase [4]. It is a major challenge for the practitioners (especially architects) to proactively comprehend the impact of their design choices on the overall greenness of the software project and the to-be-developed software system. For example, how does choosing a white background color for the mobile application impacts the greenness of the software system? While there exist certain green metrics to evaluate this once the software has been developed, but in most industrial cases, it becomes an effort- and cost-intensive process to then make amends.
- **Hidden Impact:** Unlike seeing carbon reductions from other activities like driving a car, it is a challenge for the practitioners to comprehend the potential benefits of a recommendation in terms of just reduction in emissions. This further limits the adoption. The eventual goal of any research in this area should be to convey the insights in a manner that highlights the potential impact in a relatable way (e.g., correlating potential emission savings with operational cost savings, or the number of rural households that can be lighted) and nudges the practitioner in adopting that recommendation (e.g., using gamification techniques).

¹<https://c2e2.unepdtu.org/wp-content/uploads/sites/3/2020/03/greenhouse-gas-emissions-in-the-ict-sector.pdf>

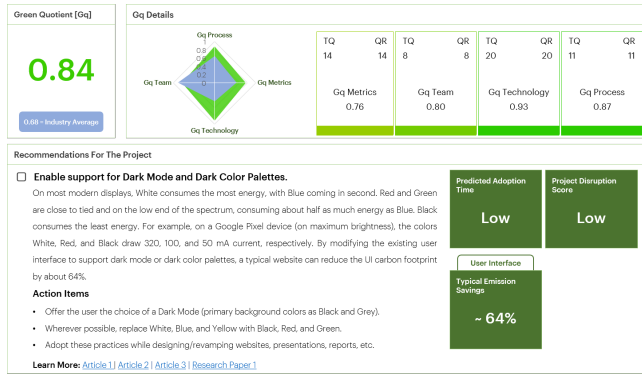


Figure 1: A slice of the report generated by PGQ approach.

In the next section, we'll discuss the potential impact on the carbon emissions of the software projects, if they can be effectively guided towards making green decisions.

2 INDUSTRY POTENTIAL OF MAKING GREEN DECISIONS

Throughout the SDLC, practitioners are faced with multiple decision points that guide the software development process going forward. For example, choice of programming language, choice of a third-party library, choice of a cloud provider, etc. If effectively guided, these decision points reserve the potential for making a software project - *greener*. For example:

- **Programming Language:** By selecting Java over Python for developing a software system, typically about 97.4% energy savings can be achieved [6]. For a large-scale software, used by millions/billions of users, this might correspond to major operational cost savings (in terms of reduced data center energy consumption), in addition to being pro-environment. Moreover, if a greener choice is not selected initially, the cost, effort, and potential emissions of code refactoring at a later stage might be exceptionally high.
- **UI Color Scheme:** By choosing dark color schemes for developing a mobile application, energy consumption of the smartphone's display can be reduced by an average of 64%, as compared to lighter color schemes [1].
- **Collaboration:** While collaborating remotely, by simply switching from a video meeting to an audio meeting (by turning off the camera), a typical team can reduce the carbon footprint of such meetings by about 96% [5]. This is even more important during the current pandemic scenario when the majority of such meetings are happening remotely.

3 PROJECT GREEN QUOTIENT

To overcome these aforementioned challenges, we have started to explore a questionnaire-based approach, *Project Green Quotient (PGQ)*, that enables a practitioner to quickly gauge the overall greenness of a software project in a non-intrusive way while receiving relevant advisory for further in-depth investigations into different non-green aspects of the project. *PGQ* spans multiple project aspects such as technology, process, metrics, and team. The intent is to be the first and overarching step in a software project's journey towards being green. The approach relies on a novel metric *Green Quotient*, that quantifies the overall greenness of a software

#	Question	Feedback (%)	
		No	Probably Yes
Q1	Did the assessment raise awareness, curiosity, and interest for further deep dive into the adoption of green practices?		100
Q2	Are these insights and recommendations helpful in incorporating green practices in the project?	33.33	66.66
Q3	Did the assessment enable the team to think of new ideas regarding the greenness of the project?		100
Q4	Did the green quotient metric (and its breakdown) allow the team to gauge the overall sustainability of the project?	16.66	83.33
Q5	According to you, will a similar assessment be helpful for other project teams seeking to adopt green practices in their projects?		100

Table 1: Descriptive analysis of the qualitative feedback received from six early adopters of the PGQ approach.

project on a scale of 0 to 1 (0 representing non-adoption of any green practices and 1 representing adoption of all green practices that the approach has been trained on). Moreover, to increase adoption, the approach also highlights the potential benefits in terms of operational cost savings, emissions reduction, etc. which can be realized by adopting the system-generated recommendations. Figure 1 showcases an early snapshot of our prototype implementation.

To understand the applicability and effectiveness of our approach, we recruited six different software projects from our delivery centers and asked them to take the PGQ assessment for their respective projects. Post completion of the assessment and generation/walkthrough of reports, the project teams were asked to complete a small qualitative feedback assessment. The results of the study are showcased in Table 1 and appear very promising.

4 CONCLUSION

Here we introduced the challenges, potential impact, and ongoing research on gaining a holistic perspective on the adoption of green decisions/practices by a software project. While the challenges and potential impact are aimed at fostering further research in this important area, the ongoing research is aimed at industry practitioners who have a desire/mandate to incorporate sustainability practices into their projects but are unaware of a starting point.

REFERENCES

- [1] P. Dash and Y. C. Hu. 2021. How Much Battery Does Dark Mode Save? An Accurate OLED Display Power Profiler for Modern Smartphones. In *Proceedings of the 19th International Conference on Mobile Systems, Applications, and Services (MobiSys)*.
- [2] J. A. Garcia-Berna, J. Carrillo de Gea, B. Moros, J. L. Fernández-Alemán, J. Nicolás, and A. Toval. 2018. Surveying the Environmental and Technical Dimensions of Sustainability in Software Development Companies. *Applied Sciences* (2018).
- [3] H. Huijgens, A. Rastogi, E. Mulders, G. Gousios, and A. V. Deursen. 2020. Questions for Data Scientists in Software Engineering: A Replication. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.
- [4] P. Lago, Q. Gu, and P. Bozzelli. 2014. *A systematic literature review of green software metrics*. VU Technical Report.
- [5] R. Obringer, B. Rachunok, D. Maia-Silva, M. Arbabzadeh, R. Nateghi, and K. Madani. 2021. The overlooked environmental footprint of increasing Internet use. *Resources, Conservation and Recycling* 167 (2021).
- [6] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva. 2017. Energy Efficiency across Programming Languages: How Do Energy, Time, and Memory Relate?. In *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017)*.
- [7] C. Sahin, F. Cayci, I. L. M. Gutiérrez, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh. 2012. Initial explorations on design pattern energy usage. In *2012 First International Workshop on Green and Sustainable Software (GREENS)*.
- [8] V. S. Sharma, R. Mehra, and V. Kaulgud. 2017. What Do Developers Want? An Advisor Approach for Developer Priorities. In *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*.
- [9] D. Torre, G. Procaccianti, D. Fucci, S. Lutovac, and G. Scanniello. 2017. On the Presence of Green and Sustainable Software Engineering in Higher Education Curricula. In *Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials (SECM '17)*.