INTEGRATING SUSTAINABILITY METRICS INTO DEVSECOPS: A RISK-BASED FRAMEWORK FOR GREEN SOFTWARE DEVELOPMENT

by

ASHWINI KUMAR RATH, MSc.

DISSERTATION

Presented to the Swiss School of Business and Management Geneva

In Partial Fulfillment

Of the Requirements

For the Degree

DOCTOR OF BUSINESS ADMINISTRATION

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

FEBRUARY 2025

INTEGRATING SUSTAINABILITY METRICS INTO DEVSECOPS: A RISK-BASED FRAMEWORK FOR GREEN SOFTWARE DEVELOPMENT

by

ASHWINI KUMAR RATH, MSc.

Supervised by

SAGAR BANSAL, DBA.

APPROVED BY:

dr. Anna Provodnikova

Dissertation Chair: Anna L. Provodnikova, PhD.

RECEIVED/APPROVED BY:

Admissions Director

Dedication

“To my beloved parents…”

Acknowledgements

First of all, I would like to express my deep appreciation to my supervisor, Dr. Sagar Bansal, for his guidance and encouragement throughout the whole process of conducting this research. I am pleased to acknowledge his feedback and advice which have been invaluable in the evolution and completion of this study.

I also thank the Swiss School of Business and Management Geneva faculty members for their knowledge and resources that helped me develop the skills required to finish this research.

I thank the industry professionals and organizations that participated in my research. The cooperation of these organizations and their willingness to be part of this study enhanced the quality of the findings.

I want to extend my gratitude to my colleagues at Batoi for their support throughout this research. They helped me to stay strong and continue working even during the most difficult times.

Finally, I would like to thank my family and friends for their support, understanding, and love. It is my great happiness to have had their support and belief in me, which has been a source of strength throughout the process. Thank you all.

ABSTRACT

INTEGRATING SUSTAINABILITY METRICS INTO DEVSECOPS: A RISK-BASED FRAMEWORK FOR GREEN SOFTWARE DEVELOPMENT

ASHWINI KUMAR RATH
FEBRUARY 2025

Dissertation Chair: Anna L. Provodnikova, PhD.

Software development and IT operations often contribute to environmental impact in ways that are frequently overlooked. As industries move toward sustainability, integrating green computing into DevSecOps workflows becomes essential. However, existing approaches lack structured methods to measure and mitigate environmental effects within software development, deployment, and management.

This study introduces two structured frameworks that embed sustainability metrics and risk evaluation within DevSecOps, helping organizations lower energy consumption, improve resource efficiency, and maintain security without compromising agility. A mixed-methods research design was employed, incorporating both qualitative interviews and quantitative surveys to identify and assess sustainability indicators in DevSecOps adoption.

The findings indicate a widespread awareness of green computing, yet a lack of standardized methodologies across software firms, including developer organizations and system integrators. By addressing this gap, the study provides actionable strategies for embedding sustainability into continuous integration, testing, and deployment processes.

By bridging theoretical research with industry applications, this work equips organizations with measurable tools to align software engineering practices with sustainability goals. Ultimately, it advances academic discourse while offering practical insights for companies integrating environmental responsibility into DevSecOps.

Keywords: DevSecOps, Green Computing, Sustainability, Software Development, Environmental Metrics, Framework Integration.

TABLE OF CONTENTS

List of Tables x

List of Figures xi

CHAPTER I: INTRODUCTION 1

1.1 Research Background 1
1.2 Problem Statement 7
1.3 Research Objectives 9
1.4 Research Questions 10
1.5 Significance of the Study 11
1.5.1 Academic Contributions 11
1.5.2 Practical Contributions 12
1.6 Structure of the Thesis 13

CHAPTER II: REVIEW OF LITERATURE 15

2.1 Introduction 15
2.2 Green Computing 17
2.3 DevSecOps 26
2.4 Green Metrics 30
2.5 Case Studies and Frameworks 33
2.6 Gap Analysis 38
2.7 Summary of Findings 39
2.8 Implications for Our Research 40

CHAPTER III: METHODOLOGY 42

3.1 Research Design 42
3.2. Data Collection Methods 46
3.2.1 Qualitative Data Collection (Semi-Structured Interviews) 46
3.2.2 Quantitative Data Collection (Structured Surveys) 47
3.2.3 Integration of Qualitative and Quantitative Methods 49
3.2.4 Tools and Technologies Used 49
3.3 Data Analysis Techniques 50
3.3.1 Qualitative Data Analysis 50
3.3.2 Quantitative Data Analysis 51
3.3.3 Integration of Qualitative and Quantitative Results 52
3.4 Ethical Considerations 54
3.4.1 Informed Consent 55
3.4.2 Confidentiality and Anonymity 56
3.4.3 Data Security 56
3.4.4 Avoidance of Bias 56
3.4.5 Adherence to Institutional Guidelines 57

3.4.6 Cultural and Contextual Sensitivity 57
3.4.7 Participant Well-Being 57
3.5 Limitations of the Study 58

CHAPTER IV: RESULTS 60

4.1 Introduction 60
4.2 Findings from Research 60
4.2.1 Qualitative Insights 61
4.2.2 Quantitative Insights 65
4.2.3 Combined Inferences 70
4.3 Benchmarking and Evaluation 71
4.3.1 Benchmarking Methodology 71
4.3.2 Industry Performance vs. Research Findings 74
4.3.3 Key Insights from Benchmarking 77
4.3.4 Summary of Benchmarking Insights 77
4.4 Case Study: Implementation of Green Metrics in DevSecOps 78
4.4.1 Case Study Overview 78
4.4.2 Implementation of Green Metrics Framework 78
4.4.3 Quantitative Impact of Implementation 81
4.4.4 Key Learnings from the Case Study 82
4.4.5 Summary of Case Study Findings 83
4.5 Summary of Findings 83
4.5.1 Key Takeaways from Research Findings 83
4.5.2 Case Study Insights: The Impact of Green Metrics Integration 84
4.5.3 Opportunities for Improvement 85
4.5.4 Conclusion 87

CHAPTER V: DISCUSSION 88

5.1 Introduction 88
5.2. The Need for Structured Green Metrics 89
5.3 The Green Metrics Framework (GMF) 90
5.4 Core Green Metrics 91
5.5 The Risk-Maturity Assessment Framework 94
5.5.1 Core Components of RMAF 95
5.5.2 Alignment with DevSecOps 97
5.5.3 Application and Benefits of RMAF 97
5.5.4 Comparative Analysis: RMAF vs. Existing Models 98
5.5.5 Challenges in Implementing RMAF 99
5.6 Comparison with Prior Work 100
5.6.1 Green Metrics Framework vs. Existing Green Computing Models 100

5.6.2 Risk-Maturity Assessment Framework (RMAF) vs. Existing Maturity Models 101
5.6.3 Novel Contributions of GMF and RMAF 102
5.7 Challenges and Opportunities 103
5.7.1 Challenges in Implementing Sustainability in DevSecOps 103
5.7.2 Opportunities for Advancing Sustainability in DevSecOps 104

CHAPTER VI: CONCLUSION AND RECOMMENDATIONS 107

6.1 Summary of Findings 107
6.1.1 Key Findings 107
6.1.2 Contributions to Research and Practice 108
6.2 Practical Recommendations 109
6.2.1 Recommendations for Organizations 109
6.2.2 Recommendations for Policymakers and Regulators 110
6.2.3 Recommendations for Practitioners and DevSecOps Teams 110
6.3 Future Research Directions 111
6.3.1 Expanding the Scope of Green Metrics in DevSecOps 112
6.3.2 AI and Automation for Sustainability Monitoring 112
6.3.3 Economic Impact of Green DevSecOps Adoption 113
6.3.4 Policy and Regulatory Frameworks for Green DevSecOps 113
6.3.5 DevSecOps Culture in Sustainability Adoption 113
6.4 Closing Remarks 114

APPENDIX A: INITIAL INTERVIEW GUIDE 116

APPENDIX B: INITIAL SURVEY QUESTIONNAIRE TO UNDERSTAND INTEGRATING OF GREEN METRICS INTO DEVSECOPS 119

APPENDIX C: SUSTAINABILITY AND DEVSECOPS MATURITY SURVEY 123

APPENDIX D: EXISTING METRICS AND MAPPING METRICS TO GOALS 128

APPENDIX E: SCORING MODEL FOR SUSTAINABILITY MATURITY AND RISK IN DEVSECOPS 136

REFERENCES 145

LIST OF TABLES

Table 3.1a: The flow of our research 45

Table 3.2a: The data collection process 47

Table 3.2b: Sampling and distribution for the structured surveys 48

Table 4.2a: Synthesis of the coding process in Grounded Theory, showing how particular concepts are categorized and then connected to form a cohesive theory. 63

Table 4.2b: The levels of adoption of sustainability into software processes in organizations. 68

Table 4.3a: The benchmarking analysis compared industry leaders in sustainability with the findings from our research. 75

Table 4.3b: Key Performance Indicators (KPI) Dashboard 76

Table 4.4a: Green Metrics Implementation Roadmap. 79

Table 4.4b: Quantitative Impact of Green Metrics Implementation. 81

Table 4.5a: The impact of green metrics integration in the case study. 85

Table 5.3a: The sustainability challenges in DevSecOps and solutions with the GMF. 90

Table 5.4a: The three-phase implementation model of the GMF. 93

Table 5.5a: The maturity levels as defined in RMAF. 96

Table 5.5b: The RMAF maturity levels and DevSecOps principles. 97

Table 5.5c: A comparison of RMAF with other maturity models. 99

Table 5.6a: A comparison of GMF with existing sustainability models. 101

Table 5.6b: A comparison analysis of RMAF with other maturity models. 101

Table E.1: Example Weight Distribution 138

Table E.2: Structured Scoring Model with Sustainability. 144

LIST OF FIGURES

Figure 1.1a: Components that drive Application Lifecycle Management (ALM) 2
Figure 1.1b: Different Metrics for DevSecOps..... 5
Figure 1.2a: Green Computing and DevSecOps - Challenges..... 8
Figure 2.1a: Impact of the green metrics on the software development..... 15
Figure 2.2a: Dimension of sustainability..... 18
Figure 2.2b: Illustration of green computing and its scope within the sustainability study..... 26
Figure 2.3a: Illustrating Major Challenges for DevSecOps..... 28
Figure 2.4a: The Key Functions of the Green Metrics 31
Figure 2.5a: Approaches to develop and implement green metrics in IT..... 37
Figure 3.1a: Different phases of research 43
Figure 3.1b: Advantages of mixed-method design. 45
Figure 3.2a: The structured survey - the constituents and the quantifiable data to be captured. 48
Figure 3.3a: Illustration of why the integrative analysis approach has been employed. 54
Figure 3.4a: Various ethical considerations in our research. 55
Figure 4.2a: Critical themes for the integration of green computing practices into DevSecOps workflows..... 64
Figure 4.2b: Size Distribution of Organizations 66
Figure 4.2c: Industry Sector Distribution 66
Figure 4.2d: Role Distribution of Respondents 67
Figure 4.2e: Challenges in Integrating Green Metrics..... 68
Figure 4.2f: Resources Needed to Overcome Challenges 69
Figure 4.2g: Observed Impacts from Green Metrics Integration..... 70
Figure 4.2h: Combined inferences from our qualitative and quantitative studies. 71
Figure 4.3a: Illustration for the benchmarking process in our study 72
Figure 4.3b: Comparison of environmental efficiency metrics. 76
Figure 4.4b: The impact of green metrics implementation, showing changes in server utilization and carbon footprint. 82
Table 4.5b: Key Findings Matrix. 86

Figure 4.5a: Sustainability Impact Assessment chart. 86
Figure 5.1a: The key challenges while integrating green computing practices into DevSecOps workflows..... 89

CHAPTER I:

INTRODUCTION

1.1 Research Background

Rising concern about environmental sustainability on the one hand and increasing dependence on digital systems on the other has led to efforts to investigate the impact of digitalization on the environment and how to make it sustainable. The current pace of digitalization and a firm reliance on IT infrastructure has led to considerable energy consumption, thereby causing environmental degradation. For instance, data centers worldwide currently contribute to around 1% of global electricity usage as per the International Energy Agency (IEA), with estimates placing it between 1-1.5% of total electricity consumption worldwide. (Spencer and Singh, 2024). Green computing has emerged as a response, emphasizing energy efficiency, reduction in carbon emissions, and e-waste management. While organizations have adopted green computing at the hardware level - such as optimizing data centers - sustainable practices in software development, particularly within the DevSecOps framework, remain underdeveloped.

Meanwhile, a significant development in software process management has occurred when the security practices are shifted to the beginning and embedded throughout the development process instead of being an activity after software deployment. Tortoriello (2022) emphasizes the need to reexamine Application Lifecycle Management (ALM), particularly its evolutionary aspects. The core parameters that regulate ALM are governance, development, and operation, which include all software activities, as illustrated in figure 1.1a.

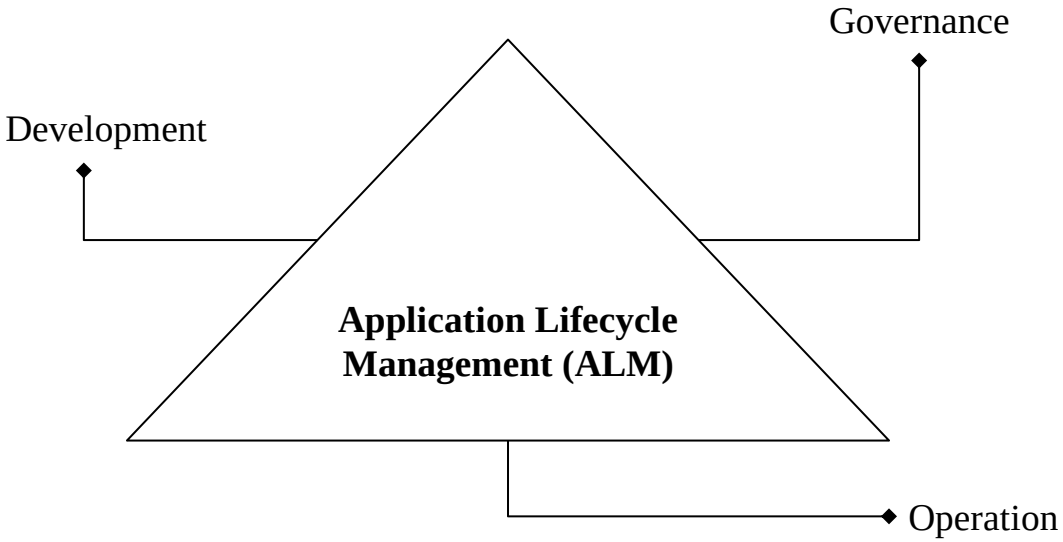

Figure 1.1a: Components that drive Application Lifecycle Management (ALM)

Mann and Maurer (2015) have also highlighted the agile development process and practices. Later, the evolution of software processes and methodologies gave rise to DevOps and DevSecOps with 54 peer-reviewed studies published between 2011 and June 2020 (Rajapakse et al., 2022). Industrial acceptance increased with more focus on the speed of software making and the complexities involved.

It is to be noted that the software sustainability is not limited to the development phase but extends to maintenance and deployment. It includes strategies for reducing the energy consumption of software running on servers and end-user devices and implementing green IT policies in software maintenance and operational procedures.

Traditional security practices, often implemented late in the development process, can hinder the agility of DevOps. Azad and Hyrynsalmi (2023) conducted a systematic literature review to identify critical success factors (CSFs) for DevOps projects, categorizing them into technical, organizational, and social and cultural dimensions.

DevOps evolves into DevSecOps by integrating security into each step of the software development lifecycle. This is in contrast to the conventional practices where security considerations were often addressed late in the development process. DevSecOps includes security from the beginning and encourages collaboration between development, operations, and security teams. It emphasizes collaboration, automation, and continuous feedback across all stages of software development. This approach includes practices like Continuous Integration and Continuous Deployment (CI/CD), enhances collaboration, accelerates delivery times, and optimizes resource utilization (Koskinen, 2019; Ahmed, 2019; Rath, 2024).

Green Computing, or ecologically sustainable computing, is a set of practices and methodologies designed to minimize the environmental or ecological impact of computing technologies (Vikram, 2015). It encompasses various activities and strategies focused primarily on energy efficiency, reducing carbon footprints, and managing e-waste (electronic waste). Energy efficiency in green computing involves optimizing hardware and software operations to consume the least energy possible while retaining performance. Reducing the carbon footprint concerns lowering the overall greenhouse gas emissions associated with computing activities - from data center operations to the end-user device. E-waste management addresses the responsible disposal and recycling of electronic components and devices, recognizing the harmful environmental effects of electronic waste (Sagar and Pradhan, 2021).

In software development and IT operations, one of the primary challenges, especially with respect to sustainable computing, is the need for standardized metrics and benchmarks for sustainability. Measuring and comparing the environmental impact of software products and IT practices is difficult. Furthermore, more comprehensive tools and methodologies are needed to improve the sustainability of software systems. Of course,

quickly changing technology landscape and many disruptive innovations that are taking place in the IT sector do offer considerable challenges too.

One a different note, there is a considerable dependence on energy resources with an increasing demand for the large-scale use of data and cloud-based services. This has put onus on adhering to sustainable practices, and need for greater awareness and education about the importance of sustainable computing among all stakeholders.

Despite the recognition of the importance of the sustainable practices in software processes, the adoption has yet to catch up. There can be several reasons for this, but the primary reason is the lack of standardized metrics and frameworks measure and guide the environmental impact of software development effectively.

Moreover, integrating sustainability into software development often needs balancing performance with ecological considerations. It includes developing energy-efficient algorithms, using sustainable software design principles, and implementing tools that measure software's environmental impact throughout its lifecycle. For DevSecOps, these practices must be integrated into the CI/CD pipelines without impacting operational efficiency.

Recent studies show the possibility of incorporate energy-aware programming and resource optimization tools in CI/CD pipelines (Sallou, Cruz and Durieux, 2023; Rajapakse et al., 2022). Tools like EnergiBridge facilitate cross-platform energy measurements that can be integrated into DevSecOps workflows; thus, we shall be able to to monitor and minimize energy consumption during automated builds and deployments (Sriraman and Raghunathan, 2023).

Metrics are important of for providing quantifiable insights into security, performance, and operational efficiency. The key metrics of DevSecOps are illustrated in Figure 1.1b.

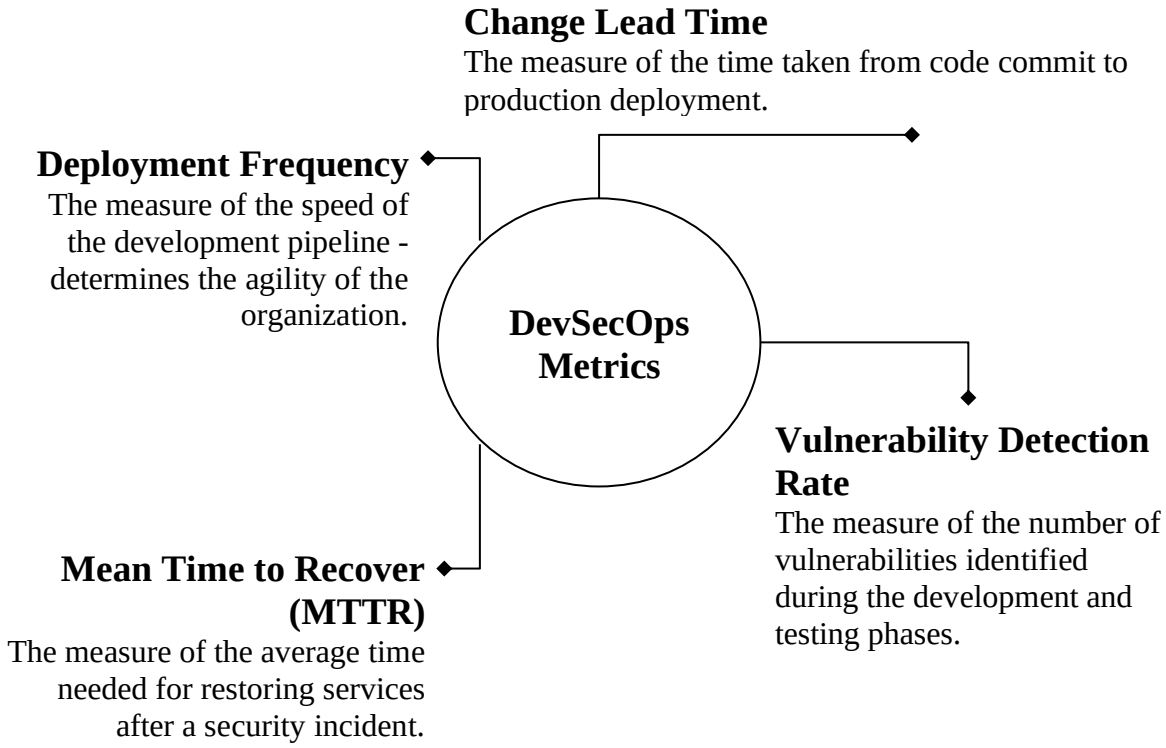

Figure 1.1b: Different Metrics for DevSecOps

Deployment Frequency reflects an organization's agility. It measures how frequently code is deployed to production or updates are released. For example, if an application is deployed 25 times a month, the deployment frequency is 25 deployments/month.

$$Deployment\ Frequency\ = \frac{Number\ of\ Deployments\ in\ a\ Period}{Total\ Time\ of\ the\ Period}$$

Mean Time to Recover (MTTR) is the average time needed to restore services after a security incident or failure. For example, if you had three incidents with recovery times of 2, 2.5, and 1.5 hours, the MTTR would be 2 hours.

$$MTTR = \frac{Sum\ of\ Recovery\ Times\ for\ All\ Incidents}{Number\ of\ Incidents}$$

Vulnerability Detection Rate is the number of vulnerabilities identified during the development and testing phases. For example, if we find 50 vulnerabilities out of 1000 tests conducted, the rate would be 0.05. The value may also be expressed in percentage; in this case, it is 5%.

$$Vulnerability\ Detection\ Rate = \frac{Number\ of\ Detected\ Vulnerabilities}{Total\ Number\ of\ Tests\ or\ Code\ Units\ Scanned}$$

Change Lead Time is the time spent between code commit and it's successful deployment in production. For example, if a commit is made at 10:00 AM and the code is deployed at 2:00 PM, the Change Lead Time is four hours.

$$Change\ Lead\ Time = Deployment\ Time - Code\ Commit\ Time$$

It is to be noted that there are other metrics in use; for example, Change Failure Rate (Number of Failed Deployments / Total Deployments), Security Automation Coverage (Automated Security Checks / Total Security Checks), Mean Time to Detect (Sum of Detection Times / Number of Incidents Detected), and Compliance Score (Compliant Processes / Total Processes.

Despite these advancements, the literature (as we shall see in detail in Chapter 2: Literature Review) reveals a major gap in systematic integrations of green metrics into DevSecOps workflows. Metrics like deployment frequency and Mean Time to Recover (MTTR) are widely used to assess security and efficiency, but the comparable metrics for

environmental performance (e.g., carbon footprint per deployment, server utilization efficiency, etc.) are rarely applied (Carlos-Eduardo et al., 2022; Bogdanović et al., 2023). This oversight leads to failure of organizations fail achieving sustainability, and consequently, hinders the development of standardized.

Studies also show that integrating sustainability into software engineering can lead to cultural shifts within organizations, and require different stakeholders (e.g., developers, operations teams, and sustainability officers) to collaborate (Ibrahim, Sallehudin and Yahaya, 2023). In short, we must address the challenges such as resistance to change, the lack of tools for green computing, and the absence of well-defined sustainability metrics.

Our research fills these gaps by proposing two frameworks that aligns DevSecOps principles with green computing goals. It aims to enable organizations to balance security, operational efficiency, and environmental impacts by appropriately integrating green metrics into DevSecOps processes.

It must be noted that we use phrases like green metrics, environmental sustainability metrics, ecological sustainability metrics, sustainability metrics interchangeably. It is the same for green computing, environmentally sustainable computing, ecologically sustainable computing, sustainable computing.

1.2 Problem Statement

DevSecOps has created a paradigm shift in how we do software development and management while incorporating security in the process. It boosts agile and iterative workflows with increased operational efficiency. On the other hand, the environmental impact of IT operations, including software development, has come under scrutiny. Despite integrating security and performance metrics within DevSecOps frameworks, we have largely ignored environmental considerations or have done some ad hoc progress only.

Organizations have made progress in greening hardware and infrastructure, but software development workflows (particularly where DevSecOps has been employed) have lagged in incorporating sustainability metrics. Figure 1.2a illustrates the major hurdles when efforts are made to integrate green metrics into DevSecOps.

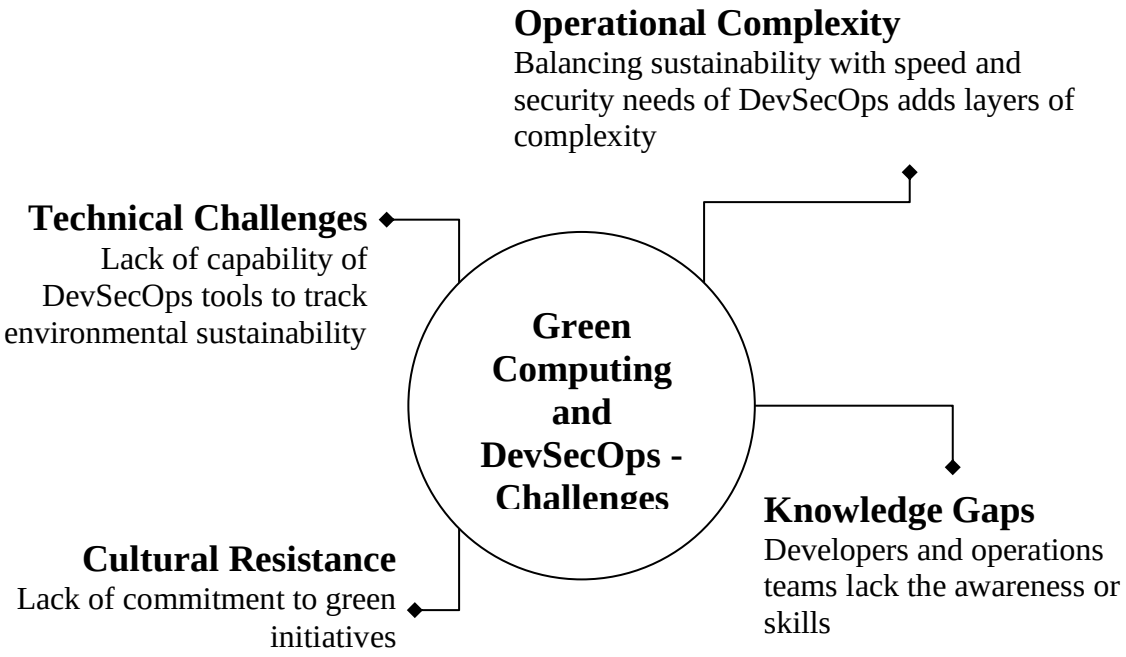

Figure 1.2a: Green Computing and DevSecOps - Challenges

Discussing this further, the existing DevSecOps tools are not designed to track environmental sustainability, making integration complex and resource-intensive; thus, it adds to Technical Challenges. Also, there is a broad lack of commitment to green initiatives in the software process, which follows from the prioritization of speed over sustainability by organizations - a Cultural Resistance. This is compounded by the challenge that arises from the lack the awareness or skills on the part of developers and operations teams who are needed to implement green practices effectively within CI/CD pipelines and other

workflows - this is Knowledge Gaps. Lastly, the need for a balance between sustainability efforts and the speed DevSecOps adds another layer of complexity, Operational Complexity, to already fast-paced and iterative processes, as highlighted by Rajapakse et al., 2022.

1.3 Research Objectives

Our research aims to integrate DevSecOps practices and green computing by proposing two comprehensive frameworks with a set of metrics and quantification of maturity and risk in the process. Our research objectives are as follows:

1. Investigate the Current State of Green Computing in Software Development: We shall examine existing practices and tools for sustainability in software process and identify the extent of adoption of green computing principles in DevSecOps workflows.
2. Identify Key Sustainability Metrics Relevant to DevSecOps: We shall explore and define measurable indicators (metrics) with a focus on aligning environmental sustainability in DevSecOps processes.
3. Develop a Framework for Integrating Sustainability Metrics into DevSecOps: We shall design an actionable framework that integrates green metrics with DevSecOps workflows.
4. Evaluate the Effectiveness of the Proposed Frameworks: We shall assess the impact of quantifying environmental footprint while maintaining software security and operational efficiency in practical scenarios.
5. Promote Industry Adoption of Green Metrics: We shall furnish practical guidelines and recommendations for industry professionals for Green DevSecOps.

Our research aims to enhance the theoretical understanding and practical application of sustainability in DevSecOps. It will also offer actionable insights for organizations that aspire to balance environmental responsibility with the speed and security of modern software processes.

1.4 Research Questions

The following research questions are presented to address the gaps in integrating green computing into DevSecOps and to achieve the objectives of this research:

1. What are the current challenges in integrating green computing practices into DevSecOps frameworks? This question seeks to identify technical, cultural, and organizational barriers that hinder the adoption of sustainability metrics in DevSecOps workflows.
2. Which sustainability metrics are most relevant for measuring environmental impact within DevSecOps processes? This question aims to explore and define key green metrics, such as energy consumption, carbon footprint, and e-waste management, that align with the principles of green computing.
3. How can green metrics be systematically integrated into DevSecOps workflows without compromising security or operational efficiency? This question focuses on designing a framework that balances sustainability goals with the speed and agility demanded by modern software development practices.
4. How does the integration of green metrics impact the overall efficiency, security, and sustainability of software development processes? This question evaluates the effectiveness of embedding sustainability metrics into DevSecOps, examining the benefits and trade-offs through practical application and case studies.

5. What strategies can facilitate the widespread adoption of green metrics in DevSecOps across industries? This question discusses different practical solutions and organizational best practices that can promote the integration of sustainability into DevSecOps and expedite adoption.

1.5 Significance of the Study

It is an urgent necessity to integrate green computing practices into DevSecOps processes. It will balance environmental sustainability with modern software development, IT operation, and security demands. This study is meaningful for both academia and industry, and offers valuable contributions in the following ways.

1.5.1 Academic Contributions

Advancing Theoretical Understanding: The study fills the gap in the literature and expands the theoretical foundations of sustainable software development by systematically examining the intersection of green computing and DevSecOps. Also, the research aims to introduce new insights into how green metrics can be aligned with DevSecOps practices, advancing the discourse on sustainability in IT workflows.

Development of Frameworks: The proposed frameworks aim to provide academic benchmarks for integrating sustainability metrics into software development workflows.

Stimulating Future Research: Our research offers a foundation for future academic inquiry, encourages researchers to investigate into the practical implementation of green computing practices in other iterative and security-focused software methodologies and IT processes.

1.5.2 Practical Contributions

Enabling Organizations to Meet Sustainability Goals: The research findings provide actionable tools to organizations to reduce their carbon footprint, optimize resource utilization, and manage electronic waste.

Facilitating Green Computing Adoption Industry-Wide: Our research offers a set of guidelines and metrics that can be employed in various organizational and technological scenarios. It will promote a widespread integration of sustainability practices into software engineering.

Improving Cost Savings and Operational Efficiency: There will be greater possibility for organizations adopting the framework to achieve energy efficiency, reduce waste, and lower operational costs; and can create a compelling business case for green computing.

Strengthening Regulatory and Social Responsibility Alignment: Our research aims to help organizations mitigate compliance risks, enhance their corporate reputation, and ensure that their social responsibility efforts are in line with prevailing regulatory requirements.

Promoting Collaboration Between Teams: Our research emphasizes the importance of cross-functional collaboration among development, security, operations, and sustainability teams. It encourages shared responsibility for environmental stewardship within organizations.

Our study contributes to broader global efforts to combat climate change by addressing the environmental challenges of IT. Our findings support organizations in their effort to align with sustainability objectives of the United Nations Sustainable

Development Goals (SDGs), such as Goal 12 (Responsible Consumption and Production) and Goal 13 (Climate Action).

To summarize, our study help bridge the gap between theoretical investigation and practical relevance. The structured approach for embedding green computing practices into DevSecOps workflows delivers actionable insights to benefit academics, industry, and society at large and facilitates a sustainable future for software engineering.

1.6 Structure of the Thesis

This thesis is organized into six chapters as below; offers a comprehensive exploration of integrating green computing practices into DevSecOps frameworks:

Chapter 1: Introduction - This chapter gives an overview of our research, such as its background, problem statement, objectives, research questions, and significance. It establishes the foundation for understanding the gap between green computing and DevSecOps and outlines the scope and purpose of the research.

Chapter 2: Literature Review - This chapter explores the existing literature on DevSecOps, green computing, and sustainability metrics. It identifies current trends, challenges, and gaps in integrating sustainability into software processes. It underscores the need for green metrics for DevSecOps and paves the way for devising the proposed framework.

Chapter 3: Research Methodology - In this Chapter, the mixed-methods approach for the study is discussed. The methods of data collection, such as qualitative interviews, quantitative surveys are elaborated along with the analysis techniques to synthesize results. The ethical considerations and the limitations of the study are also discussed.

Chapter 4: Results - The key sustainability metrics are identified from the findings and enunciated in the new framework. Case studies and benchmarking results that validate the framework and demonstrate its practical applicability are included.

Chapter 5: Discussion - The results are interpreted in the context of the research objectives and questions. The chapter discusses what the results imply for industries and academics and highlights their potential to advance sustainable software development. The challenges and opportunities of adopting green metrics in DevSecOps are also included.

Chapter 6: Conclusion and Recommendations - This chapter summarizes the results of the study. It highlights the research contributions to theoretical understanding and practical applications. It offers a set of actionable recommendations for organizations pursuing the adoption of green metrics in DevSecOps. Finally, future research directions are outlined to further the effort of maximizing operational efficiency and security through automation and an integrated approach to sustainability practices in software engineering.

CHAPTER II:

REVIEW OF LITERATURE

2.1 Introduction

This research follows Garousi, Felderer and Mäntylä (2022) for conducting Multivocal Literature Reviews (MLRs) in software engineering, integrating grey literature (e.g., blogs, videos, and white papers) alongside traditional academic sources. In their study, the authors outline the benefits of MLRs, including how this bridge the gap between academic research and industry practice. Figure 2.1a illustrates the literature review process and the flow followed iteratively to reach a comprehensive set of findings.

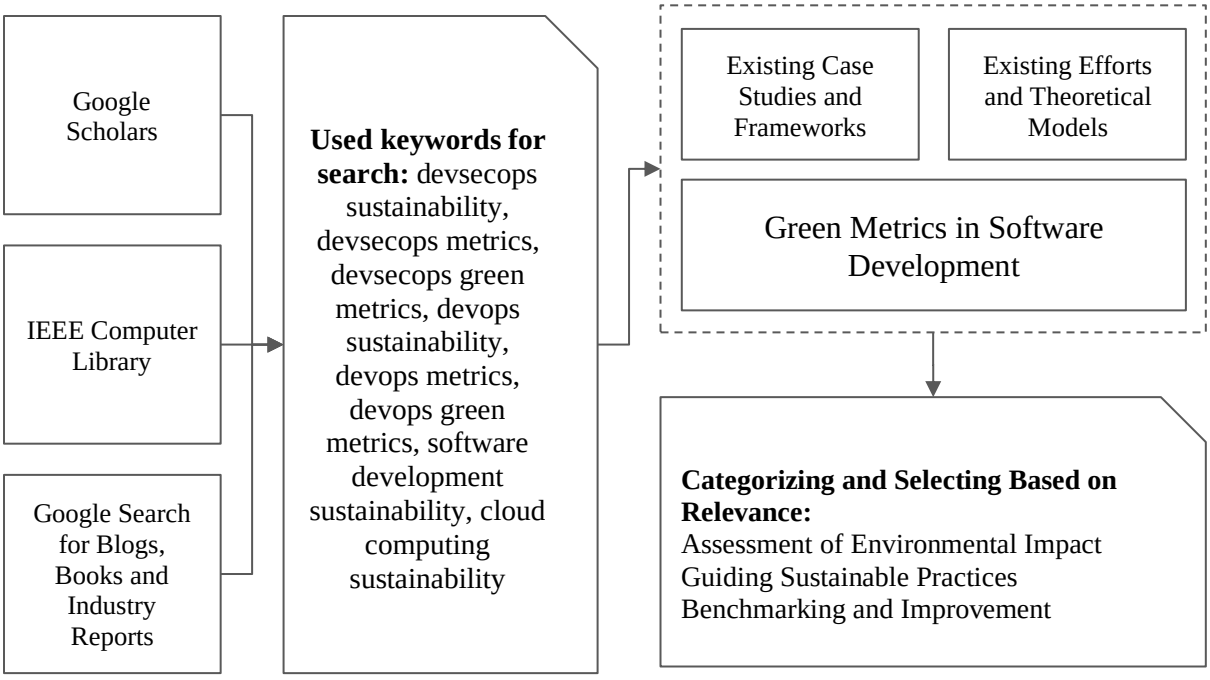

Figure 2.1a: Impact of the green metrics on the software development

We have adopted a methodological approach to analyze the impact of green metrics on software development. The primary data sources include scholarly databases

such as Google Scholar and IEEE Computer Library and broader searches encompassing blogs, books, and industry reports. For the literature review, we have used relevant keywords such as *DevSecOps sustainability*, *DevSecOps metrics*, *DevOps green metrics*, *software development sustainability*, and *cloud computing sustainability*, etc. as mentioned in Figure 2.1a. This search identified the existing case studies, frameworks, theoretical models, and specific metrics relevant to software development. We have also employed AI tools like SciSpace (typeset.io) to look for more associated articles and Paperpile App (paperpile.com) to archive and sort reviewed articles. We have then categorized the findings based on their significance, particularly in three critical areas: assessing environmental impact, guiding sustainable practices, and benchmarking improvements.

This systematic approach helps our review capture a balanced perspective, emphasize the role of green metrics in sustainable software practices, and boost the potential to address environmental challenges in the digital era. Moreover, the categorization serves as a basis for exploring the integration of (ecological) sustainability principles into software processes and for evaluating its practical implications.

In our review, we shall present the studies and their findings in a particular order. First, we shall discuss studies relevant to substantiating green computing needs and then examine how different studies look at sustainability in general and green computing in particular. We shall then discuss DevSecOps advancements in software practices and highlight the sustainability perspectives. Later, we shall present studies about green metrics and DevSecOps before moving to gap analysis and our next step. It is to be noted that we shall frequently move between the considerations strictly about green computing in DevSecOps and the general scope of sustainability in software practices; again, it will help us ascertain overlapping studies and consequent insights for our purpose.

2.2 Green Computing

We shall review the state of studies in green computing in the ICT industry in general and software practices in particular. The reason for expanding our purview to the large state of affairs is to capture the essence due to the coinciding nature of software and other components of ICT activities. It is, in fact, more relevant due to the software-centric approach to digital technologies with the prevailing role of cloud computing worldwide (Rath, 2013).

Different studies have also discussed different characteristics of sustainability and how the environmental aspect is considered. For example, Gerostathopoulos, Raibulet and Lago (2022) propose an approach that leverages decision maps to systematically capture sustainability-relevant concerns, categorized into technical, economic, environmental, and social dimensions. Similarly, Garscha (2021) explores integrating sustainability considerations into agile frameworks and emphasizes the five dimensions of sustainability - environmental, economic, technical, social, and individual - and highlights the role of requirements engineering in shaping sustainable software systems.

Chitchyan et al. (2016) emphasize the need for systemic thinking in software design while investigating into sustainability in requirements engineering. They identify obstacles like the lack of education on sustainability, inadequate organizational support, and prioritization of immediate goals over long-term impact. Their study of sustainability encompasses environmental, economic, social, individual, and technical dimensions. Similarly, Chitchyan, Noppen and Groher (2015) examine the role of sustainability in Software Product Line Engineering (SPLE), and also explore its alignment with economic, technical, social, and environmental dimensions. They used a case study of the DiVA project and grounded theory analysis to identify key sustainability-related concepts, such as efficiency, productivity, adaptability, and tool support.

Thus, if we select the common components, we may represent sustainability with four dimensions, as illustrated in Figure 2.2a.

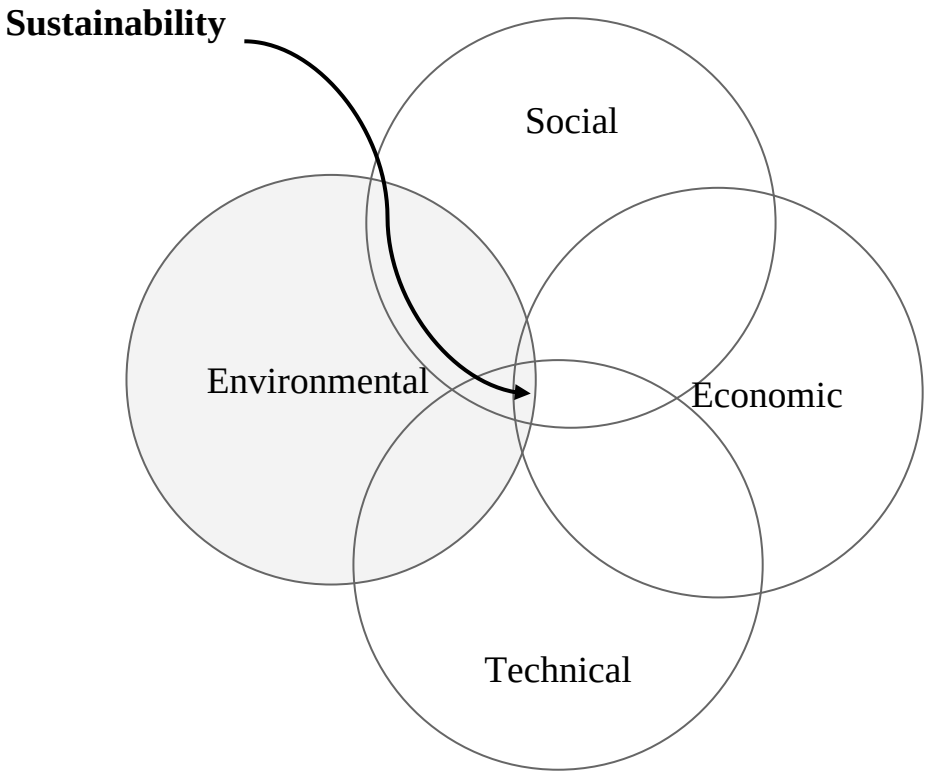

Figure 2.2a: Dimension of sustainability

Essentially, green computing can be understood as environmental sustainability. Wolfram, Lago and Osborne (2017) provide a systematic mapping study on the definition and application of sustainability in software engineering. The study goes further and categorizes sustainability concerns across the software lifecycle; it emphasizes areas such as software requirements, design, and runtime, with a predominant focus on environmental sustainability. The authors also propose the emerging need to embed sustainability into software engineering practices.

Harmon and Auseklis (2009) state that the efficient use of computing resources to enhance performance while minimizing environmental impact. They note that power consumption by IT departments constitutes up to 50% of the organization's energy costs - a major financial and ecological hurdle for adopting sustainable IT practices. They advocate integrating green computing strategies - such as power management, virtualization, improved cooling technologies, recycling, and electronic waste disposal - into IT services to meet sustainability objectives. They also identify a gap in understanding the strategic benefits of sustainable IT services, particularly in creating customer, business, and societal value, and propose a set of principles to guide sustainable IT service design.

Gmach et al. (2012) explore the integration of demand-side and supply-side energy management for sustainable data centers. Their approach involves a detailed energy profiling method that models power demand using workload simulators and incorporates dynamic supply sources like photovoltaic arrays and municipal solid waste facilities. The study stresses using metrics such as CO2 emissions, water consumption, and operational costs to assess sustainability.

Saha (2018) highlights the growing importance of green computing in mitigating environmental challenges that arise due to rapid industrialization and technological advancements. The study emphasizes designing, manufacturing, and using computing resources in environmentally sustainable ways; also focuses on energy efficiency, recyclability, and reducing hazardous materials and underscores the potential of green IT practices to address global environmental concerns and the need for collaboration among stakeholders. Even earlier literature focused on the direction - Schopf (2009) examines sustainability within the National Science Foundation's Office of CyberInfrastructure (OCI), emphasizing the importance of creating reusable, reliable, and long-lasting software and services to support computational science.

Erdélyi (2013) examines the role of software in enhancing the eco-sustainability of IT systems, and emphasizes that software behavior significantly influences hardware energy consumption. The study identifies factors in green software development, including energy-efficient algorithms, resource optimization, and sustainable coding practices. Fakhar et al. (2012) explore energy conservation in software systems through green computing strategies implemented at the software level.

Groher and Weinreich (2017) investigate sustainability in software development through interviews with team leads from Austrian companies. They highlight a factor of the narrow understanding of sustainability among practitioners, focusing mainly on technical aspects like maintainability and extensibility.

In an earlier study, Albertao et al. (2010) propose a methodology to assess the sustainability performance of software projects by introducing a comprehensive set of sustainability metrics. These metrics include environmental, economic, and social aspects, focusing on properties such as modifiability, reusability, portability, and supportability. They emphasize iterative improvement by assessing metrics at the end of each release cycle, which enables targeted sustainability goals.

Andrikopoulos et al. (2022) conducted a systematic mapping study to explore the intersection of sustainability and software architecture. The study identifies gaps in the coverage of sustainability dimensions (technical, environmental, economic, and social) and architecture lifecycle activities. The technical dimension gets more importance, while social and environmental dimensions remain underexplored. Furthermore, the study states the need for comprehensive approaches integrating all sustainability dimensions across the software lifecycle. This work contributes to the growing recognition of sustainability as a critical software quality.

Condori-Fernandez and Lago (2018) examine the role of quality requirements in contributing to software sustainability across technical, economic, social, and environmental dimensions. Through a survey-based study involving software architects, requirements engineers, and sustainability experts, they identify key quality attributes like modifiability, recoverability, and satisfaction as significant contributors to sustainability. Landauer (2006) introduces a framework, called *Wrapping Architectures*, to enhance the long-term sustainability of complex systems.

Haron et al. (2015) highlight the role of software reusability in promoting green computing within Malaysia's IT sector. Similarly, a study by Venters et al. (2018) evaluates software sustainability from the point of view of software architecture. Lago (2019) proposes *Decision Maps*, which frame architectural concerns across technical, economic, social, and environmental sustainability dimensions, as a tool to embed sustainability in software architecture design. Catolino (2020) employs mixed-method approach to examine how *code smells* (the issues within software codebase while it gives correct output) evolve over time and impact project sustainability.

Spencer and Singh (2024) opine that the increasing demand for AI services could significantly escalate electricity consumption by data centers, which currently account for about 1-1.5% of the global value. The authors emphasize the necessity for energy-efficient technologies and sustainable practices to mitigate potential environmental impacts. Lange, Pohl and Santarius (2020) recommend targeting energy-reducing effects while mitigating energy-increasing mechanisms associated with digitalization.

Jayalath et al. (2019) review of green cloud computing, focusing on adopting green-computing attributes and implementations by leading cloud service providers. The study examines how cloud computing, as a more efficient alternative to traditional data centers, contributes to environmental sustainability. In fact, there have been studies on cloud

adoption independent of sustainability perspective (Rath et al., 2012), but then points to the importance of looking at the role of cloud computing that has brought large-scale transformation in software-centric view and environmental outlook. However, the authors note that the energy consumption of cloud data centers remains a significant concern due to the reliance on non-renewable energy sources.

Moreover, green computing extends to developing green policies, and implementing environmentally friendly IT practices, i.e., using renewable energy sources for managing data centers and promoting energy-efficient hardware and software. This approach considers the entire lifecycle of IT products and services with an aim to reduce their environmental impact from production to disposal. Raja (2021) investigates the significance of green computing within the IT sector, emphasizing the imperative of enhancing energy efficiency in computing technologies. The study opines the swift adoption of green information technology (IT) as a formal organizational policy, extending beyond environmental strategies to encompass the holistic development of society. The study of Singh (2015) on green computing strategies and challenges states the critical importance of adopting eco-friendly practices in the production and utilization of advanced technologies.

Harmon and Demirkan (2011) discuss the evolving concept of sustainable IT, emphasizing its transition from product-focused *Green IT* initiatives aimed at cost and energy efficiency to a broader alignment with corporate sustainability (CS) and corporate social responsibility (CSR). They opine that sustainable IT holds potential beyond ecological benefits; addresses economic, legal, and social dimensions of CSR. Ono, Iida and Yamazaki (2017) suggest integrating digital technologies like cloud and AI into corporate strategies to drive environmental transformation. This approach illustrates how ICT can bridge industries and promote sustainable development across sectors.

Ibrahim, Sallehudin and Yahaya (2023) investigate the intersection of software development waste reduction and environmental sustainability through the application of Lean principles, referred to as *Green Lean*. Their study emphasizes that software development, being a complex socio-technical activity, often involves activities that do not add value to the customer or user, identified as waste. By adopting Lean methodologies, which focus on waste reduction and efficiency enhancement, the authors propose that software development processes can become more environmentally sustainable.

Anwar and Pfahl (2017) conducted a systematic mapping study to explore the role of software analytics in fostering green software engineering. Their research highlights the use of software analytics techniques, such as statistical analysis, text mining, and pattern detection, to enhance energy efficiency in software development. Despite advancements, only 11 out of 50 reviewed studies utilized software analytics for greener software practices, emphasizing the need for automated tools and metrics to bridge the gap between energy consumption and other quality attributes. The study identifies gaps in high-level software design and energy-aware maintenance and proposes these as future directions for green software engineering.

Tee, Abdullah and Abdullah (2015) conduct a systematic literature review on green software development (GSD) within collaborative knowledge management (KM) environments. The study identifies key focus areas, including energy efficiency and sustainable software lifecycle practices, while emphasizing the role of KM in managing and sharing GSD knowledge. Their findings highlight the limited scope of current GSD research and the need for integrating KM techniques to enhance knowledge sharing and environmental practices in software development. The study endorses development of frameworks incorporating KM to bridge knowledge gaps and sustain best practices for greener software.

Bambazek, Groher and Seyff (2022) investigate the integration of sustainability considerations into agile software development, particularly within the Scrum framework. Their survey study among practitioners reveals a high potential for addressing sustainability impacts through agile processes despite a lack of shared understanding of sustainable software. The study identifies opportunities for assessing sustainability during key Scrum events (agile methods), such as product backlog refinement and sprint reviews. It highlights the importance of team involvement and stakeholder collaboration.

As mentioned by Calero, Moraga and García (2022), two key perspectives in the intersection of software and sustainability are increasingly recognized in the field such as Sustainability IN Software (Software Sustainability - SOS) and Sustainability BY Software (Software as Part of Sustainability - SAPOS). Sustainability IN Software focuses on making the software itself sustainable by minimizing resource consumption during development and management and maximizing its longevity. The emphasis is on the sustainability of the software development process using energy-efficient coding practices, minimizing resource usage, and considering how software can be maintained over time without excessive energy costs. On the other hand, Sustainability BY Software views software as a tool for achieving sustainability in other areas. In this view, software is developed sustainably and contributes to the sustainability of different systems and processes, i.e., software helps monitor and reduce energy consumption in smart grids or optimize resource allocation in supply chains; they play a crucial role in promoting overall sustainability. Bash et al. (2023) emphasize a fundamentals-based framework to address sustainability through supply-demand principles, focusing on minimizing resource consumption and optimizing operational efficiencies.

SOS focuses on making software itself sustainable; emphasizes efficient use of resources, thereby minimizing negative environmental impacts, while SAPOS highlights

software’s role in addressing broader sustainability goals. The authors specify key relationships by aligning these perspectives with the United Nations Sustainable Development Goals (SDGs). They suggest that sustainability principles must be integrated into software development practices, and there must be more awareness among developers and users about the environmental and societal impacts of software. They also state the importance of embedding sustainability into software engineering as both a process and a product focus.

These perspectives highlight software's dual role in sustainability efforts, both as an entity that needs to be made sustainable and as a powerful tool for enabling broader sustainability goals. We shall use the term sustainable computing or green computing interchangeably and shall take Sustainability IN Software into account for all our purposes. The sole reason for this choice is to focus our energy on the aspects that impact the ecology due to different computing processes and practices.

The review uncovers various ways contemporary studies use sustainability - environmental, economic, technical, and social. Figure 2.2b illustrates the broad understanding of sustainability from different perspectives and where the current research focuses.

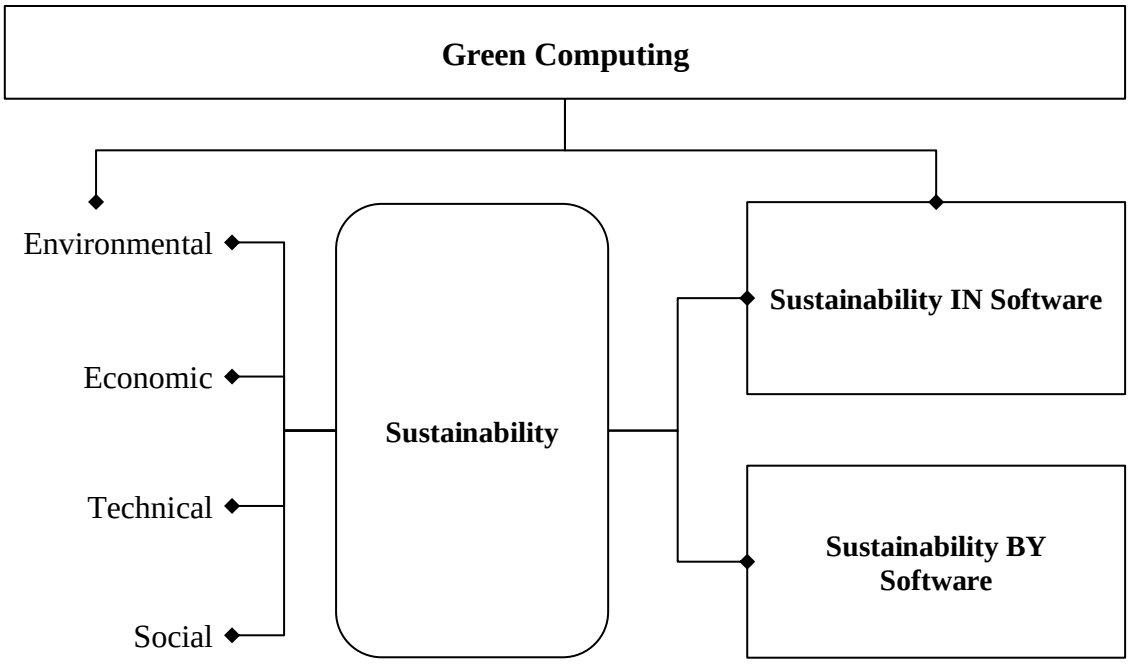

Figure 2.2b: Illustration of green computing and its scope within the sustainability study.

In the next section, we review DevSecOps from the standpoint of Application Lifecycle Management, the evolution of DevOps into DevSecOps, and the metrics used in DevSecOps.

2.3 DevSecOps

Chen and Suo (2022) explore the integration of security within the software development lifecycle through DevSecOps, emphasizing its role in embedding security measures at every stage - from initial design to deployment. The study introduces an integrated DevSecOps platform to enhance enterprise research and development efficiency and automating security protections. By making security a shared responsibility across development, security, and operations teams, the platform seeks to streamline processes and bolster the overall security posture of applications and infrastructure.

The key principles of DevSecOps include automation, continuous integration (CI), continuous deployment (CD), and proactive security measures. By combining automated security checks, vulnerability scanning, and real-time monitoring, DevSecOps reduces the likelihood of security breaches while maintaining the iterative nature of modern software development (Rath, 2024). Bermon Angarita, Fernández Del Carpio and Osorio Londoño (2022) underline collaboration and automation to streamline the software development lifecycle, where metrics play a key role in the quality assurance of software. Roche (2013) examines the integration of DevOps principles into quality assurance (QA) practices. The study points out the evidence that DevOps removes traditional silos between teams of developers, QA, and operations. This promotes a culture of collaboration and shared responsibility. Key advancements include process standardization, automation of testing, and the use of metrics for real-time decision-making.

Tortoriello (2022) advocates for automated security tools like SAST and DAST, apart from cultural and organizational changes, to enhance collaboration among teams for development, operations, and security. Mack (2023) emphasizes continuous security at speed to meet modern software demands. Rahul, Kharvi and Manu (2019) highlight the importance of incorporating static and dynamic security analyses into the development cycle to address vulnerabilities without compromising the speed of DevOps processes. Arseneault et al. (2022) highlight the critical role of Integration and Test teams in ensuring product quality and stability amidst changing schedules and requirements.

Pakalapati, Venkatasubbu and Sistla (2023) highlight the potential of Artificial Intelligence (AI) to revolutionize software development and to enhance security by automating threat detection, predicting vulnerabilities, and streamlining workflows. Guzman Camacho (2024) stresses automated threat detection, predictive analytics for vulnerability management, and intelligent automation as key applications of AI/ML within

DevSecOps. The author demonstrates how AI/ML-driven DevSecOps can improve organizational resilience, streamline processes, and mitigate evolving security threats. Petrović (2023) investigates using generative AI like ChatGPT for runtime DevSecOps log analysis, and compares it to traditional machine learning.

The growth and adoption of DevSecOps are not without challenges. As illustrated in Figure 2.3a, the following can be considered major challenges.

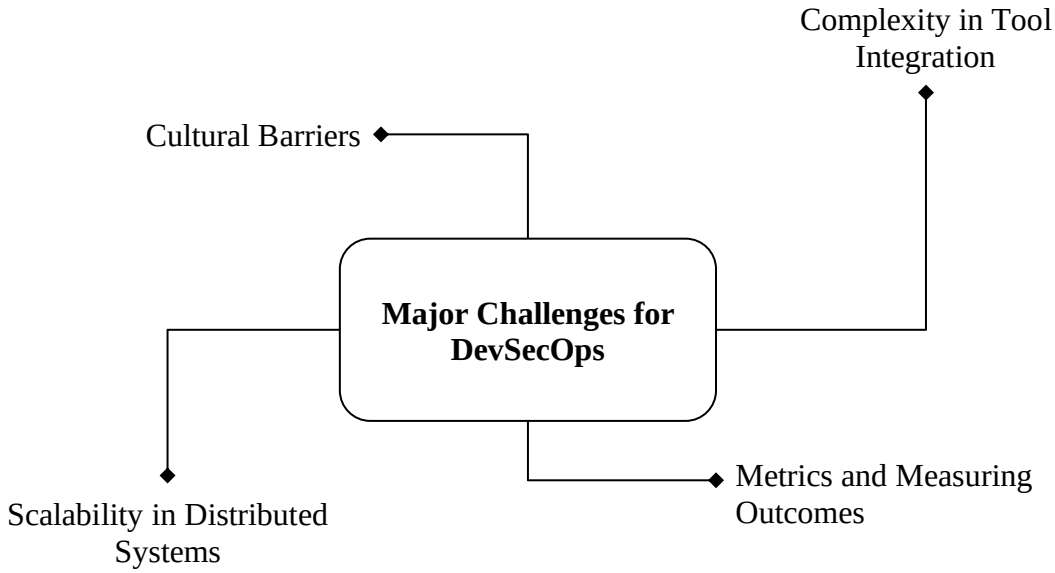

Figure 2.3a: Illustrating Major Challenges for DevSecOps

Tortoriello (2022) has discussed the resistance to change and the siloed nature of teams in traditional organizations that hinder the seamless integration of DevSecOps practices. The sheer number of tools required for DevSecOps, including CI/CD platforms, security scanners, and monitoring systems, makes integration challenging, particularly for small and medium-sized enterprises.

Rajapakse et al. (2022) conducted a systematic review to identify challenges and solutions associated with adopting DevSecOps. The study categorizes issues into four

key themes: tools, practices, infrastructure, and people. Key challenges include tool integration difficulties, limited automation of manual security practices, and inter-team collaboration barriers. The authors propose solutions such as developer-centric tools, automated vulnerability assessment practices, and frameworks for complex environments. Emphasizing the need for hybrid tools like Interactive Application Security Testing (IAST), the review highlights gaps in tool standardization and continuous security assessment.

Gupta (2022) presents a structured framework for DevSecOps adoption, emphasizing its significance in integrating security into the software development lifecycle. The study highlights DevSecOps as a cultural and operational shift that addresses security vulnerabilities by embedding security practices from the development phase through deployment.

The fourth factor is measuring the outcome of DevSecOps (with a set of clearly defined metrics).

The metrics are used to monitor progress and gain insights into the performance of the software delivery process, as per Bermon Angarita, Fernández Del Carpio and Osorio Londoño (2022). The most significant contributions of DevOps in terms of metrics are related to continuous integration, software design, and software testing, as outlined in Orozco-Garcés, Pardo-Calvache and Suescún-Monsalve (2022). In the context of DevSecOps, more than traditional metrics and evaluation methods are required to ensure software security. New security metrics must be defined based on multiple measures to increase reliability. The measurement of metrics in DevOps and DevSecOps can be done manually, through surveys, or automatically (Carlos-Eduardo et al., 2022; Wissam, Mallouli et al., 2020). The authors emphasize for an automatic and accurate measurement of DevSecOps metrics as it would be of great value for practitioners in improving software

delivery performance. Prates et al. (2019) conducted a multivocal literature review to identify key metrics for measuring the effectiveness of DevSecOps implementations; the study describes nine metrics, including defect density, defect burn rate, critical risk profiling, and top vulnerability types, which are essential for tracking vulnerabilities, prioritizing risks, and improving development processes.

2.4 Green Metrics

As discussed in the previous section, the metrics determine the performance of DevSecOps workflows. However, they largely ignore environmental considerations. As sustainability becomes a critical global concern, expanding these metrics to include green computing dimensions - such as energy consumption per deployment and carbon footprint - represents an essential step forward (Li and Zhou, 2011). The authors propose dynamic power management, application-aware energy policies, and low-power hardware designs to address the challenges of implementing sustainable computing practices. Moreover, the paper identifies the need for holistic approaches to integrate energy awareness into resource management across computing systems, advocating for collaborative efforts between hardware and software to achieve comprehensive energy efficiency. This work underscores the necessity of continued research to develop unified energy metrics, improve energy modeling, and optimize green networking protocols, paving the way for the next generation of sustainable IT systems.

This section reviews the existing efforts to integrate sustainability into IT practices, with a particular focus on software development and emphasizing sustainability in software. This perspective is particularly relevant to integrating sustainable practices within the DevSecOps framework.

First, let us discuss green metrics themselves. Green metrics in software development refer to quantifiable measures to assess and manage the environmental impacts of software systems. These are vital as the industry is increasingly scrutinized for its ecological footprint. In the software development, the green metrics have three key functions as illustrated in Figure 2.4a.

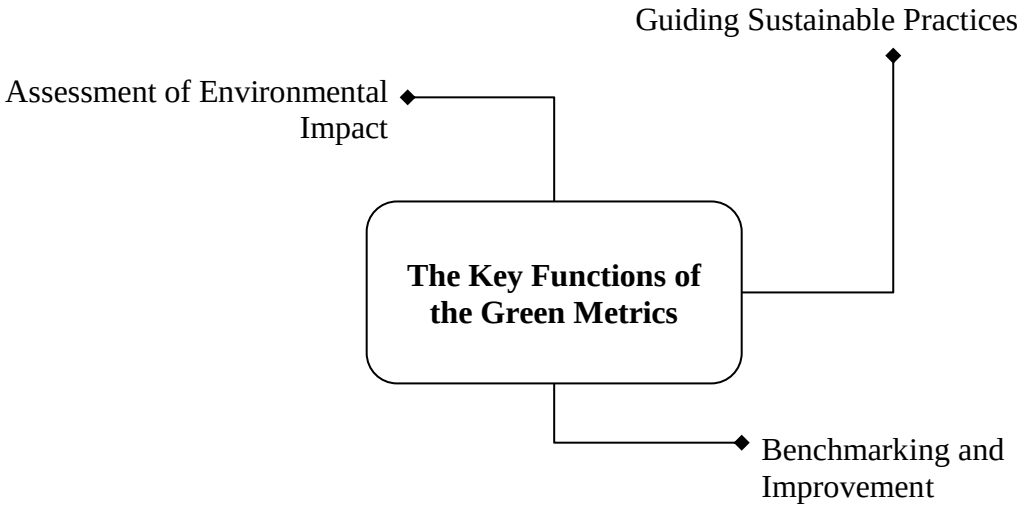

Figure 2.4a: The Key Functions of the Green Metrics

Assessment of environmental impact involves evaluating the energy efficiency, carbon footprint, and overall ecological impact of software products, from development to deployment and maintenance. We quantify the environmental aspects of software systems - guiding organizations in making more informed, eco-friendly decisions - termed as guiding sustainable practices. The third key function of green metrics is benchmarking and improvement, which helps identify sustainability improvement areas when we benchmark against industry standards or regulatory requirements.

The growing emphasis on sustainability in business and among consumers magnifies the relevance of green metrics. As environmental issues become more

prominent, the demand for transparency and accountability in software development practices increases. Welter and Benitti (2014) emphasize the significance of green computing in promoting environmental sustainability, and the need for tools, practices, processes, frameworks, and reference models. Similarly, Debbarma and Chandrasekaran (2016) highlight the importance of energy efficiency and reduced environmental impact.

Mourão, Karita and Machado (2018) stress the need for standardized metrics and methodologies to assess software sustainability. Mallouli et al. (2020) highlight the integration of automated and continuous measurement processes within the software development lifecycle. This approach prescribes structured metrics as defined by the SMM (Structured Metrics Meta-model) standard; thus integrates tools for automated vulnerability detection, secure coding, and anomaly detection. Moreover, the framework introduces real-time data analysis with AI/ML to predict risks, optimize system performance, and reduce energy consumption.

The current state of sustainable computing is characterized by growing awareness and incremental adoption of green practices within the IT industry. Freitag et al. (2021) highlight that while there has been progress, particularly in the hardware domain with energy-efficient data centers, the software development aspect of sustainable computing could be more advanced.

Bozzelli, Gu, and Lago (2019) identify and classify metrics for assessing software "greenness" in software engineering based on their systematic literature review. Analyzing 960 publications, they choose 23 primary studies, extracting 96 distinct green metrics. Their analysis shows that research on green software metrics has predominantly focused on energy consumption and with limited attention to other sustainability aspects. It emphasizes the need for comprehensive models to assess green software qualities beyond

mere energy efficiency, and also highlight a gap in the literature for evaluating the overall sustainability of software systems.

2.5 Case Studies and Frameworks

Quite a few case studies and frameworks have been proposed towards the integration of environmental considerations into IT processes and practices. There have been many directions that these studied have taken. For example, Le et al. (2016) investigate the relationship between architectural decay and the sustainability of software systems, emphasizing the role of *architectural smells* such as unused interfaces and sloppy delegation.

We cite a few region-centric case studies here. Riekstin et al. (2016) propose a framework integrating real-time differentiated life-cycle assessment models to calculate carbon emissions and energy consumption across distributed ICT networks worldwide. In another effort, Ahmaro, Bin Mohd Yusoff and Abualkishik (2014) highlight a growing commitment among Malaysian IT firms to enhance energy efficiency and reduce environmental impact through sustainable computing strategies.

Tjoa and Tjoa (2016) emphasize the dual nature of ICT's impact - enabling sustainability through optimization (positive) and electronic waste (negative). Philipson (2011) introduces a framework and emphasizes the importance of measurement and benchmarking in driving effective Green ICT strategies. Agarwal, et al. (2015) highlight the importance of adopting environmentally friendly computing practices to reduce energy consumption and minimize environmental impact. The study identifies energy-efficient hardware utilization, optimization of software algorithms for lower power usage, and the implementation of virtualization techniques to enhance resource efficiency. The authors advocate for more awareness and widespread adoption of green computing within the IT

industry. Muthu, Banuroopa and Arunadevi (2019) emphasize the importance of sustainability in software engineering and propose a model to extend traditional lifecycle frameworks incorporating metrics to evaluate ecological, economic, and social impacts. The authors highlight the need for sustainable approaches in requirement study, architecting, coding, testing, and implementation. Ruokolainen and Kutvonen (2012) propose a holistic approach to service ecosystem engineering, incorporating analysis, design, and governance to enhance stakeholder collaboration and system viability.

Jimenez, Calero and Moraga (2022) introduce a tool to assess the incorporation of software sustainability actions in the CSR (Corporate Social Responsibility) initiatives of software development companies and highlight gaps in the implementation of software sustainability practices. Uddin and Rahman (2012) propose a comprehensive Green IT framework to enhance energy in large-scale data centers. The framework applies green metrics like Power Usage Effectiveness (PUE), Data Center Efficiency (DCE), and Carbon Emission Calculator to assess and benchmark the performance of individual data center components. Wang, Palanisamy and Xu (2020) present a sustainability-aware resource provisioning framework to reduce the total carbon footprint of data centers.

Winters (2018) emphasizes the importance of sustainable software practices in environments with extensive dependencies and long-term maintenance requirements. Alharthi, Spichkova and Hamilton (2018) introduce a web-based tool to evaluate software across five sustainability dimensions, such as individual, social, technical, economic, and environmental. Ibrahim, Sallehudin and Yahaya (2023) underscore the idea of reducing waste and enhancing sustainability in software processes.

Penzenstadler et al. (2021) address the need for sustainability impact assessments in software development at various stages of a product's lifecycle. They highlight the potential for regulatory integration of sustainability assessments in IT, akin to practices in

civil engineering, to ensure responsible development of digital solutions. Albertao et al. (2010) highlight the feasibility of incorporating sustainability considerations into software engineering processes. Ferri, de Barros and Brancher (2011) propose a sustainability framework integrating IT governance, green computing principles, and virtualization technologies to promote eco-sustainability in software development. Lami and Buglione (2012) propose a process-centric approach to measure software sustainability and introduce a new process group (Sustainability Management, Sustainability Engineering, and Sustainability Qualification processes) under the ISO/IEC 12207 framework.

Wlodarczyk and Rong (2010) raise concerns about the sustainability impacts of the integration of cloud computing and the cyber-physical space. Chang, Wills and De Roure (2010) examine cloud business models and their impact on sustainability. Their work offers practical tools for organizations adopting cloud computing along with sustainable growth.

Kipp and Jiang (2011) introduce a set of green metrics to identify energy inefficiencies within applications and offer guidance for improving design and execution to enhance energy efficiency. Kienzle, Strooper and Viller (2016) propose a comprehensive framework for integrating sustainability into the software development lifecycle and emphasize the importance of environmental, economic, and social impacts at each stage of the software process. Their study highlights the need for embedding sustainability as a core consideration. Jindal and Gupta (2012) state that adopting green computing strategies will address ecological concerns apart from offering economic benefits. Haugsvær (2023) introduces methodology to reduce the carbon footprint of web products by integrating sustainability principles into every phase of the development lifecycle.

Bang et al. (2013) identify tools and methodologies that contribute to modern web application development. Bogdanović et al. (2023) explore the role of DevOps in sustainable enterprise development. They present case studies of DevOps practices that

successfully aligned with sustainability goals. Zohaib (2023) presents a decision-making framework for sustainable DevOps. This framework acts as a conduit between rapid development methodologies and environmental considerations.

Studies like those by Sriraman and Raghunathan (2023) have highlighted the importance of sustainability in software engineering and DevOps practices. They propose a framework to enhance sustainability practices across the software lifecycle. Sallou, Cruz and Durieux (2023) provide practical tools and assessment criteria for implementing green metrics in software development. Dahab et al. (2022) propose a framework for green software measurement and use machine learning techniques to assess and refine green software metrics continuously. This approach indicates a growing field of green software engineering that is moving towards better measurement.

The review suggests that it is important to develop a comprehensive set of green metrics that can be readily integrated into existing IT workflows, and these metrics must provide measurement of energy consumption and carbon footprint apart from taking into account factors like resource utilization, waste generation, and software product lifecycle impacts. Figure 2.6 outlines different aspects (approaches) to develop and implement green metrics in IT; so, also in DevSecOps.

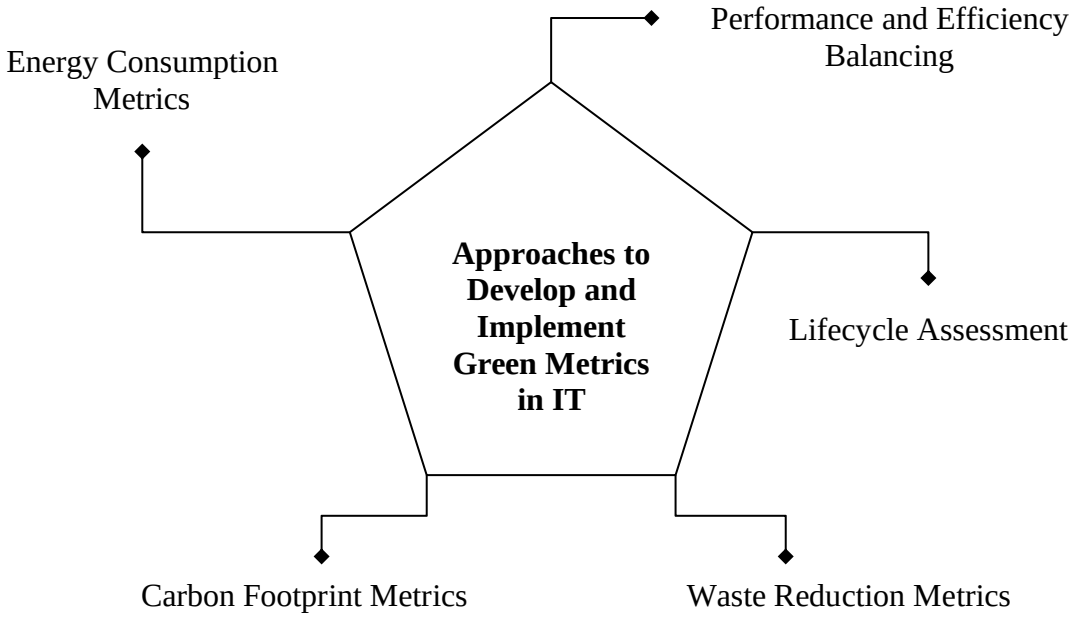

Figure 2.5a: Approaches to develop and implement green metrics in IT

The *Energy Consumption Metrics* measure the software's energy consumption during operation. It includes developing tools and methodologies for assessing the energy efficiency of code, algorithms, and software architectures. The *Lifecycle Assessment* evaluates the environmental impact of software from development to end-of-life. The *Carbon Footprint Metrics* calculates carbon footprints of software products that involve estimating the greenhouse gas emissions associated with the entire software lifecycle. The *Waste Reduction Metrics* focus on reducing electronic waste through efficient coding practices, extending the lifespan of software, and promoting recyclability and reuse in software development. The *Performance and Efficiency Balancing* considers the balance between software performance and environmental efficiency.

2.6 Gap Analysis

Our review of the existing literature in the intersection of sustainable computing, green metrics, and DevSecOps reveals critical gaps, and suggests the need for further research. While awareness about green computing is increasing, integrating these practices within the DevSecOps framework is still nascent. The reviewed literature indicates advancements in green metrics related to energy consumption, carbon footprint, and waste reduction. Still, these are generally not designed with the unique dynamics of DevSecOps in mind. . It is to be noted that there are many existing approaches providing qualitative assessments or broad guidelines without offering quantifiable metrics to systematically integrate into DevSecOps processes. It limits the potential for targeted improvements in sustainability practices within DevSecOps.

Moreover, there needs to be more clarity between the development of green metrics and their practical implementation in the fast-paced, security-focused environment of DevSecOps. The current green metrics predominantly focus on the the operational phase, with less attention given to integrating these metrics throughout the CI/CD pipeline, which is central to DevSecOps.

Our research addresses these gaps by developing and integrating quantitative green metrics for DevSecOps. We focus on creating a comprehensive set of metrics in evaluating environmental impacts that is practical and can be applicable within the rapid and security-focused context of DevSecOps. These metrics can provide measurable parameters to assess and manage the sustainability in the software processes, from coding and testing to deployment and maintenance.

The introduction of quantitative green metrics in DevSecOps is vital for the following reasons:

1. *Environmental Accountability:* Software developers, security experts and operations teams can ascertain their environmental impact. It will lead to more informed and responsible decision-making.
2. *Balancing Speed, Security, and Sustainability:* Appropriate integration without impacting the existing DevSecOps performance parameters ensures that pursuing sustainability will not compromise the speed and security.
3. *Benchmarking and Continuous Improvement:* The use of metrics allows organizations to benchmark their software practices against industry standards and regulatory requirements. It will help promote a culture of continuous improvement while adopting green DevSecOps.

2.7 Summary of Findings

The literature review has provided several key findings:

- *Emerging Field of Environmental Sustainability in DevSecOps:* We recognize the growing need to integrate environmental sustainability in software processes, in general, and DevSecOps workflows, in particular.
- *Green and Lean Software Development:* The idea of reducing waste and enhancing sustainability in software processes aligns well with DevSecOps objectives and emphasizes the need for environmental consideration in development frameworks.
- *Frameworks for Green Software Design:* Multiple studies show a focused approach towards minimizing carbon emissions and energy consumption, especially in cloud services, which can be integral to DevSecOps.

- *Systematic Mapping of Green Metrics in Software Engineering:* There is a growing emphasis on quantifiable environmental impact measures in software development.
- *DevSecOps Frameworks with Sustainability:* There is a great need to create frameworks with defined green metrics for DevSecOps and with the abilities for assessing the implementations and outcomes.

These findings emphasize a growing interest and feasibility in embedding sustainability into software development processes, particularly within DevSecOps. On the other hand, these also point to significant gaps in developing specific green metrics for DevSecOps.

2.8 Implications for Our Research

This literature review shapes the objectives and sets the direction of the proposed research. The insights acquired underscore the importance of developing green metrics for the DevSecOps practices. These metrics will address a critical gap in current sustainability practices by offering a quantifiable, practical toolset that can be integrated into the rapid, security-focused DevSecOps cycle.

The research will leverage the findings on existing sustainability efforts and frameworks in software development to create a set of metrics that are comprehensive in assessing environmental impacts and pragmatic for implementation within DevSecOps processes. The integration will allow software developers and IT operations teams to quantitatively monitor and improve the environmental performance for software applicatins without compromising on the speed and security.

Also, this research will contribute to the field by offering a novel approach to balancing technological advancement with environmental responsibility. The research will

provide a pathway for sustainable development that can adapt to future technological changes and environmental challenges.

CHAPTER III:

METHODOLOGY

3.1 Research Design

We use a mixed-methods research design in our study that combines qualitative and quantitative approaches. As specified in Chapter 2, we address the gaps in integrating green computing practices into DevSecOps frameworks. This dual approach enables us to comprehensively understand the topic by leveraging the strengths of both methods: *qualitative insights* for exploring nuanced challenges and solutions and *quantitative data* for measuring prevalence, impacts, and trends. Designing and Conducting Mixed Methods Research by Creswell and Clark (2017) offers a comprehensive guide to mixed methods research. It integrates the latest developments in the field with practical, step-by-step instructions.

A mixed-methods approach is particularly suited to our study due to the multifaceted nature of the research problem. Chapter 2 highlighted significant gaps, including the lack of standardized green metrics for DevSecOps and the limited adoption of sustainability frameworks in CI/CD pipelines. These gaps point to both methods as below:

Exploratory Insights: Qualitative methods, like semi-structured interviews, allow identification of challenges and opportunities for integrating sustainability into DevSecOps.

Confirmatory Analysis: Quantitative surveys allow measuring the prevalence and impact of these practices. It validates the themes identified qualitatively and providing cross-sectional insights across industries.

The literature review (Chapter 2) emphasized key frameworks and findings, such as the need for green metrics tailored to DevSecOps workflows (e.g., Lifecycle Assessment and Energy Consumption Metrics) and the gaps in integrating sustainability into fast-paced CI/CD environments. Our research design incorporates these findings by:

Qualitative Interviews: We have designed the interview questions to explore organizational awareness, barriers, and enablers of sustainability practices, drawing on frameworks like EnergiBridge and the Sustainability Awareness Framework (SusAF).

Quantitative Surveys: We have structured the survey questions to assess the adoption and impact of green metrics, guided by identified themes such as cultural resistance, technical challenges, and tool availability.

Our research design directly addresses the objectives outlined in Chapter 1:

- Investigating the current state of green computing practices in software development.
- Identifying key sustainability metrics that are relevant to DevSecOps.
- Developing a framework for integrating green metrics into DevSecOps.
- Evaluating the effectiveness of the proposed framework.
- Promoting industry adoption through actionable insights.

Now, let us discuss the flow of our study in detail. As illustrated in Figure 3.1, we conducted the research in three distinct phases.

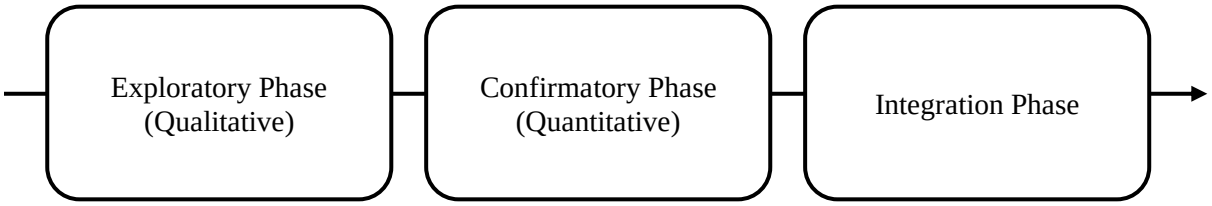

Figure 3.1a: Different phases of research

We have described the objective, method used, and outcome of each phase in Table 3.1a. The Exploratory Phase (Qualitative) aims to understand perceptions, challenges, and potential solutions for integrating green metrics into DevSecOps through semi-structured interviews with stakeholders like CTOs and sustainability officers. We have identified recurring themes in this phase that will be used later in the survey design. The Confirmatory Phase (Quantitative) measures sustainability practices' prevalence, challenges, and impacts using structured surveys distributed across diverse industries. This provides statistically reliable data to validate qualitative findings and establish benchmarks. Finally, the Integration Phase synthesizes results from both methods through thematic and statistical analysis to develop a validated framework for green metrics integration, addressing both theoretical gaps and practical applications.

Phase	**Objective**	**Method**	**Outcome**
Exploratory Phase (Qualitative)	Understand perceptions, challenges, and potential solutions for integrating green metrics into DevSecOps.	Semi-structured interviews with key stakeholders, such as CTOs, DevSecOps managers, and sustainability officers.	Identification of recurring themes and contextual insights to inform survey development.
Confirmatory Phase (Quantitative)	Measure the prevalence, challenges, and impacts of sustainability practices within DevSecOps environments.	Structured surveys distributed to software professionals across diverse industries and organizational sizes.	Statistically reliable data for validating qualitative findings and benchmarking practices.
Integration Phase	Synthesize qualitative and quantitative results to develop the proposed green metrics framework.	Thematic analysis of interview data and statistical analysis of survey responses were integrated to provide actionable	A validated framework for integrating green metrics into DevSecOps - addressing both theoretical and practical dimensions.

		recommendations.	

Table 3.1a: The flow of our research

Our mixed-methods approach offers several key benefits, as illustrated in Figure 3.1b. Complementarity guarantees that qualitative insights enhance the interpretation of quantitative data, so to provide a holistic understanding of the research problem. We achieve validation by cross-referencing data from both methods and it strengthens the study's validity and reliability. On another note, applicability creates a bridge between academic theory and industry practices; and it enables the proposed framework to be actionable and grounded in real-world challenges. Finally, Alignment with Gaps pertains to the dual-method approach to address the gaps that we have identified in the reviewed literature including the need for comprehensive sustainability metrics and practical implementation strategies.

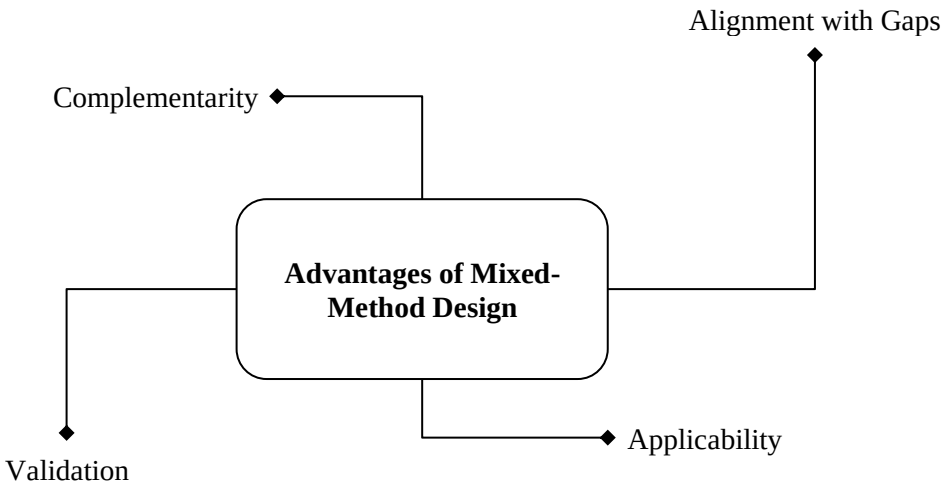

Figure 3.1b: Advantages of mixed-method design.

3.2. Data Collection Methods

We have designed our data collection process to gather insights into integrating green computing practices into DevSecOps. The process uses qualitative and quantitative methods to address the research objectives and gaps identified in Chapter 2.

This dual-method approach ensures the data collection process is comprehensive and aligned with the research objectives. Freundlieb and Teuteberg (2012) emphasize the importance of incorporating stakeholder-specific criteria in the design of data collection tools. Now, let us review each of the data collection methods.

3.2.1 Qualitative Data Collection (Semi-Structured Interviews)

Our key objective is to explore perceptions, challenges, and opportunities for integrating green metrics into DevSecOps frameworks. We have engaged key stakeholders from diverse roles in software development:

- Chief Technology Officers (CTOs),
- DevSecOps Managers,
- Sustainability Officers, and
- Software Development Team Leads.

As a sampling strategy, we have picked only participants with direct experience in DevSecOps and sustainability initiatives. It enables us to include participants with relevant expertise, enhancing the quality of the data collected.

We have organized the semi-structured interviews around the following themes (refer Appendix A for the interview guide):

- Awareness of environmental impacts in software development.
- Current sustainability practices and their alignment with DevSecOps.
- Technical and cultural challenges in integrating green metrics.

- Perceived benefits and barriers to adopting sustainability frameworks.

Table 3.2a describes the data collection process with regard to method, duration, recording and validation.

Method	Interviews conducted virtually via video conferencing tools to accommodate geographical diversity.
Duration	Each interview for approximately 45-60 minutes.
Recording	interviews recorded with participant consent - for transcription and analysis.
Validation	Pilot interview conducted to refine the questions and ensure their clarity and relevance.

Table 3.2a: The data collection process

3.2.2 Quantitative Data Collection (Structured Surveys)

We use structured surveys in the second phase to measure prevalence, challenges, and perceived impacts of ecological sustainability practices in DevSecOps across industries. We have designed the structured survey using insights from the qualitative phase to capture quantifiable data on key aspects. It focuses on Adoption (awareness and integration levels of green metrics), Challenges (technical, organizational, and cultural barriers), and Impacts (benefits of sustainability practices). The survey uses multiple-choice questions for categorical data, Likert-scale items to measure perceptions and attitudes, and open-ended questions for additional qualitative insights. Figure 3.2a illustrates our approach as described above.

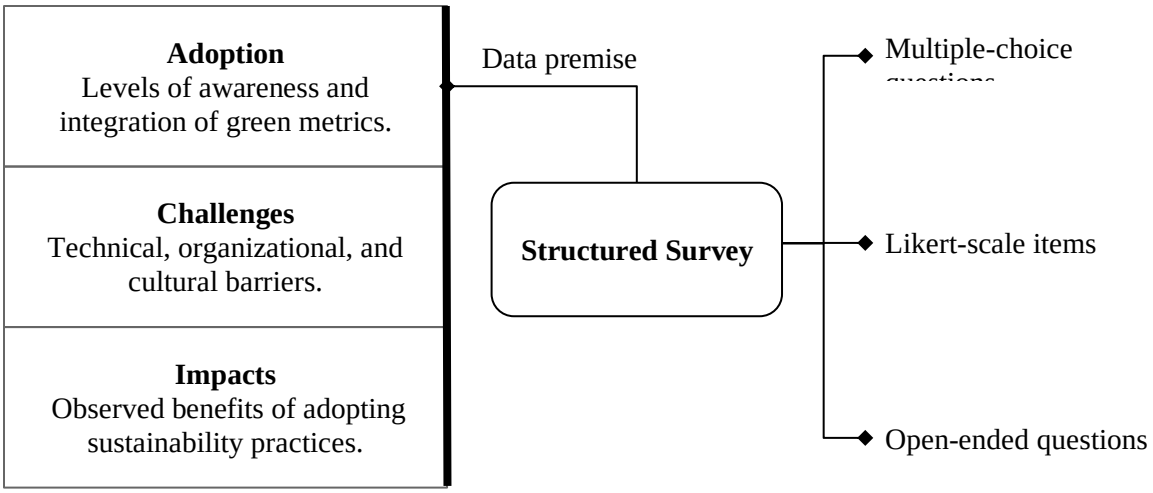

Figure 3.2a: The structured survey - the constituents and the quantifiable data to be captured.

The survey targeted software professionals from small, medium, and large enterprises in the technology, healthcare, and manufacturing sectors. To ensure a diverse respondent base, we used a convenience sampling method. To achieve statistical reliability, we distributed surveys through email and professional networks, with a target sample size of 100-150 responses. Table 3.2b describes the sampling and distribution.

Target Audience	Software professionals from small, medium, and large enterprises in technology, healthcare, and manufacturing.
Sampling Method	Convenience sampling was employed to reach a diverse respondent base.
Distribution Channels	We distributed surveys via email and professional networks.
Target Sample Size	100-150 responses targeted for statistical reliability.

Table 3.2b: Sampling and distribution for the structured surveys

The survey underwent a pilot test with 10 participants to ensure clarity and reliability. We incorporated feedback from the pilot phase into the final survey design.

3.2.3 Integration of Qualitative and Quantitative Methods

Integrating qualitative and quantitative methods provides a complementary approach to data collection. The **Qualitative Phase** provides input for the structured survey design by identifying key themes and challenges. Similarly, the **Quantitative Phase** validates these findings with broader statistical insights; i.e., qualitative themes like *organizational resistance* are quantified to measure their prevalence across industries, and quantitative data on *adoption rates* provided context for exploring underlying challenges qualitatively.

3.2.4 Tools and Technologies Used

We have used various tools and technologies in our study for rigor, efficiency, and consistency. These tools enable seamless collection, organization, analysis, and visualization of both qualitative and quantitative data.

Qualitative Data

1. Batoi Insight: For organizing and coding interview transcripts, enabling efficient thematic coding and pattern identification.
2. Taguette: Supplemented the analysis with an open-source platform for detailed qualitative data coding.
3. Google Workspace Apps: Utilized for collaboration, transcription review, and annotation of qualitative insights.

Quantitative Data

1. Microsoft Excel: For initial data cleaning, organization, and also, basic computation purposes.

2. Tableau: For advanced data visualization, helping precise and impactful representation of quantitative findings.
3. Python Custom Code: For specific statistical computations and tailored data visualization requirements.
4. Batoi Insight and Google Forms: For the design, distribution, and collection of survey responses

We have integrated results by synthesizing the findings using **Grounded Theory** and **Descriptive Statistics.** We used the former to derive patterns and insights from qualitative data. We used the latter to summarize quantitative findings while maintaining coherence between the two datasets.

3.3 Data Analysis Techniques

Our data analysis process blends qualitative and quantitative methods to examine the research objectives. We leverage the strengths of both methodologies in this mixed-methods approach that enables a nuanced understanding of integrating green computing practices into DevSecOps.

3.3.1 Qualitative Data Analysis

We have used grounded theory to analyze the interview data systematically. This approach allows for developing theories and insights based on patterns emerging from the data. We have followed the steps below for the analysis:

- Open Coding: We have done the initial line-by-line coding to identify key concepts related to sustainability in DevSecOps, such as awareness of environmental impact and challenges in metric integration.

- Axial Coding: We have grouped the related codes into broader categories, establishing connections and recurring themes, such as organizational resistance and opportunities for green metrics.
- Selective Coding: We have identified a core category - Strategies for Integrating Green Metrics in DevSecOps - to serve as the basis for our theoretical framework.

We have used the tool, Taguette, for tagging and Google Sheets for organizing and coding the qualitative data efficiently. We have adopted manual validation to ensure the accuracy and alignment of emerging patterns with the research objectives. The qualitative analysis provided insights into the following:

- Specific challenges encountered by organizations,
- Opportunities to incorporate sustainability practices into DevSecOps pipelines, and
- Contextual themes to guide the quantitative analysis phase.

3.3.2 Quantitative Data Analysis

We have used descriptive and inferential statistical methods to analyze survey data that provides both summary-level insights and identification of relationships between variables. We have followed the steps below for the analysis:

- Descriptive Statistics: We have calculated the measures of central tendency (like mean and median) and variability (standard deviation) for key survey items. We also created data visualizations (e.g., bar charts, pie charts, etc.) to present our findings.
- Cross-Tabulation Analysis: We have explored the relationships between demographic factors (e.g., organization size) and sustainability practices. For

example, we have cross-tabulated awareness of environmental impact with the integration of green metrics.

- Inferential Statistics: We have used Chi-Square tests to evaluate associations between variables, such as organizational size and the extent of green metric integration. We have also used regression analysis to assess predictors of successful green metric implementation, such as leadership support and tool availability.

We have used Python to write custom statistical computations and hypothesis-testing scripts. We have employed Tableau for data visualization to enhance clarity and presentation. We have also used Google Sheets to clean and organize the initial data. The quantitative analysis has provided actionable insights into:

- The prevalence of green computing practices in DevSecOps workflows.
- Statistical relationships between organizational characteristics and sustainability adoption.
- Factors influencing successful implementation of green metrics.

3.3.3 Integration of Qualitative and Quantitative Results

We have synthesized the results from the qualitative and quantitative analyses and have provided a holistic understanding of the research problem. The integration helped us align exploratory insights with statistical validation. We followed the following process:

- Theme Validation: We have validated the themes identified in the qualitative phase (e.g., organizational resistance) and quantified them using survey data.
- Comparative Analysis: We have compared qualitative findings on challenges with quantitative measures of their prevalence across industries and organizational sizes.

- Framework Development: We have combined the insights from both methods to develop the proposed green metrics framework, ensuring it addresses practical challenges while being grounded in empirical data.

Our integrated analysis produced the following outcomes:

- An understanding of the barriers and enablers of green computing practices in DevSecOps.
- Actionable recommendations for developing and implementing sustainability metrics in DevSecOps workflows.

We would like to note that an integrative analysis approach is essential to achive a thorough understanding, validate the framework, and generate actionable insights. We also claim that combining the contextual depth of qualitative analysis with the generalizability of quantitative data guarantees well-rounded findings. It also strengthens the reliability and applicability of the proposed framework through cross-referencing results from both methods. Moreover, the dual-method strategy confirms that the final recommendations are evidence-driven, practical, and relevant to academic and industry contexts. Figure 3.3a illustrates why this integrative analysis approach has been used.

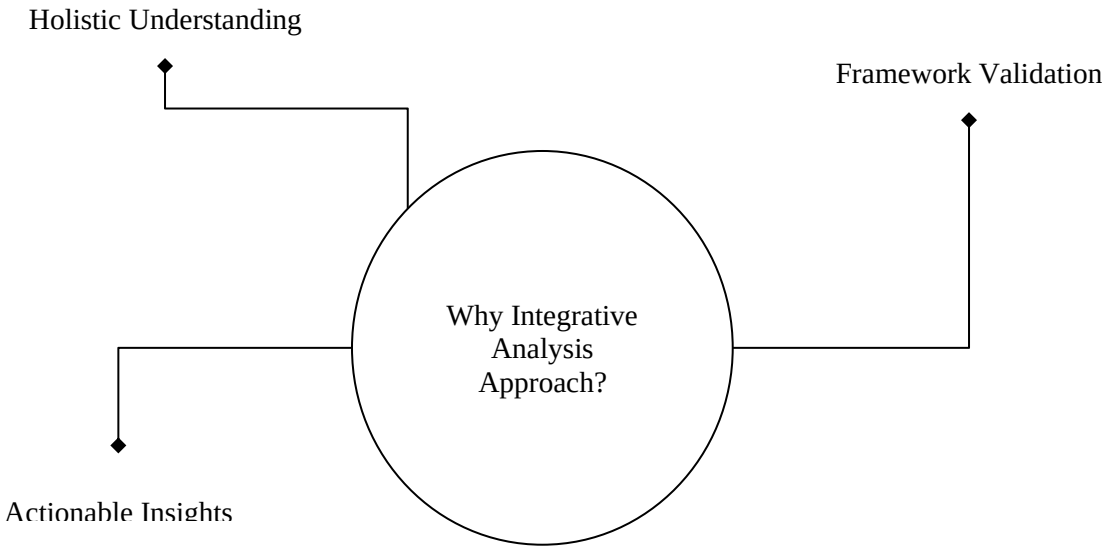

Figure 3.3a: Illustration of why the integrative analysis approach has been employed.

Our chosen data analysis techniques enable a rigorous and comprehensive examination of the research objectives, thus ensuring that the proposed green metrics framework is theoretically robust and practically viable.

3.4 Ethical Considerations

Our research adheres to rigorous ethical standards to ensure the integrity of the study and protect participants' rights and confidentiality. We have applied ethical considerations throughout the data collection, analysis, and reporting phases and have adhered to institutional ethical guidelines as illustrated in Figure 3.4a and described in detail subsequently.

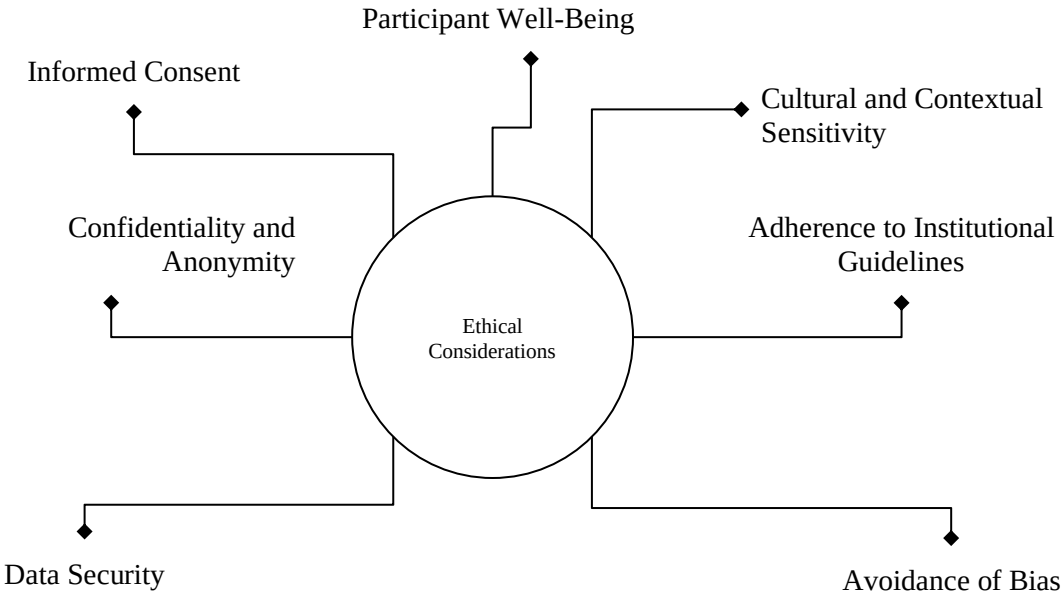

Figure 3.4a: Various ethical considerations in our research.

3.4.1 Informed Consent

We have provided participants with clear information about the study's objectives, methodology, and their role in the research. A structured consent process has ensured that all participants voluntarily agreed to participate. We have provided the following information:

- The purpose of the study and its potential benefits.
- The voluntary nature of participation, with the right to withdraw at any time without penalty.
- Assurance of confidentiality and anonymity.
- The use of collected data solely for research purposes.

We have adopted a robust consent mechanism for implementing ethical compliance and participant agreement across all data collection methods. For interviews, we have shared written consent forms electronically with participants and have to get those signed

before the commencement of the sessions. In the case of surveys, we have embedded an explicit consent option within the survey itself, requiring participants to agree to the terms before providing their responses. It confirmed that all participants are fully informed and voluntarily agree to contribute to the study.

3.4.2 Confidentiality and Anonymity

We have followed strict confidentiality and anonymity protocols to protect the privacy of our participants. We also anonymized interview transcripts by removing identifiable details such as names, organization names, and specific project information. For surveys, we have collected responses without linking any personal identifiers unless respondents explicitly provide them for follow-up. For European participants, we have securely stored digital recordings and transcripts in a secure cloud environment like Google Cloud that complies with data privacy regulations, like GDPR.

3.4.3 Data Security

We have followed strict data security protocols to prevent unauthorized access during our study. We have stored all digital files in a secured cloud environment like Google Cloud, to protect against data loss. As mentioned above, we have destroyed all physical notes and printed materials so that no sensitive information remains scattered and unsecured.

3.4.4 Avoidance of Bias

We have designed the study to minimize bias and ensure the objectivity of findings. In the interview process, we have followed a semi-structured interview guide to reduce interviewer bias while allowing for participant-driven discussions, conducting interviews

neutrally, and avoiding leading questions. For the survey design, we have structured our questions carefully to prevent bias, providing balanced response options. We have done a pilot testing to identify and address potential biases in the survey instrument. During data analysis, we have used transparent coding and statistical methods to arrive at an unbiased interpretation of the data.

3.4.5 Adherence to Institutional Guidelines

We have performed research in compliance with the ethical guidelines of the Swiss School of Business and Management (SSBM) Geneva. We have obtained ethical approval from the participants before data collection, confirming adherence to the institutional standards.

3.4.6 Cultural and Contextual Sensitivity

Due to the diverse backgrounds of participants, we have prioritized cultural and contextual considerations. We have used neutral language in interviews and surveys to avoid any cultural bias. We also acknowledge variations in organizational contexts so that our recommendations remain relevant across different industries and geographies.

3.4.7 Participant Well-Being

We have considered the well-being of our participants during the study. We have made every effort to minimize any inconvenience during participation. Participation has been entirely voluntary, with no pressure or coercion. We also offer follow-up support to participants for a chance to withdraw their contributions.

3.5 Limitations of the Study

We recognize the limitations of this study to contextualize the findings and identify areas for future research. In fact, these limitations pertain to a sample size of data, methodological constraints, and the evolving nature of the research domain. We present the list below:

Sample Size and Representation: For depth and relevance in the qualitative phase, we chose semi-structured interviews with a purposive sample of stakeholders. However, the limited number of interviews may not fully capture the diversity of practices and challenges across all industries and regions. We also aimed for 100-150 survey respondents for the quantitative phase to ensure statistical reliability. Despite this, the convenience sampling method may introduce bias, thereby may limit the generalizability of the results to a broader population.

Self-Reporting Bias: We have relied on self-reported data for both interviews and surveys, which may be subject to inaccuracies or biases. Participants might overstate their organization's commitment to sustainability or underestimate challenges due to social desirability or personal perspectives.

Evolving Nature of DevSecOps and Green Computing: We recognize that DevSecOps and green computing are rapidly changing fields, with new tools, methodologies, and frameworks emerging regularly. As a result, some findings may become outdated as technologies advance. Integrating sustainability metrics into DevSecOps is still in its early stages, with limited availability of case studies or practical implementations to analyze.

Limited Scope of Metrics: We focus primarily on key sustainability metrics such as energy consumption, carbon footprint, and lifecycle assessment. While these metrics

provide a solid foundation, we may not fully address other environmental or social dimensions of sustainability, such as electronic waste and social responsibility.

Contextual and Regional Variations: We have considered diverse industries and geographies; however, cultural and organizational differences may impact the applicability of our findings. It may also happen that practices in one region or sector may not align with those in another, limiting the universal applicability of the proposed framework.

Time Constraints: We recognize that the study's timeframe has constrained our ability to conduct longitudinal analyses or extensive pilot testing of the proposed framework. It limits the ability to assess the long-term impact of integrating sustainability metrics into DevSecOps workflows.

Technology Adoption Challenges: We acknowledge that organizations may face technological, financial, or operational barriers to adopting the proposed framework. These challenges could limit the practical implementation of findings in resource-constrained settings.

Before proceeding to the next chapter with results of our study, we would like to note that these limitations do not diminish this study's significance in addressing a critical gap at the intersection of sustainability and DevSecOps, though they may affect the breadth and generalizability of the findings. In fact, they emphasize the need for further research and iterative refinement of the proposed framework, paving the way for more robust and comprehensive studies in the future.

CHAPTER IV:

RESULTS

4.1 Introduction

We build on the foundations laid in our previous chapters, and present the findings of the study in this chapter; our focus is on qualitative and quantitative insights from research participants, benchmarking against industry practices, and real-world case studies. Our goal is to objectively report findings without interpretation, leaving discussions and analysis for the next chapter.

Key areas covered in this chapter include:

- Findings from Research: A synthesis of interview and survey results highlighting adoption levels, challenges, and opportunities for environmental sustainability integration in DevSecOps workflows.
- Benchmarking and Evaluation: Comparison of our research results with industry standards and best practices.
- Case Studies: Examination of real-world implementations of sustainability frameworks in DevSecOps environments, capturing measurable impacts and lessons learned.

These findings provide an empirical foundation for Chapter 5, where the Green Metrics Framework and Risk-Maturity Assessment Framework will be introduced and analyzed. While this chapter presents raw data and case study observations, the discussion on their significance and implications follows in the next chapter.

4.2 Findings from Research

Our study focuses on data collection through interviews and surveys. 62 professionals have participated in interviews. To maintain high analytical quality and avoid

saturation of similar information, we have selected 45 of those interview transcripts for detailed coding and analysis. The survey has received 150 responses, thus provides a robust quantitative basis for analysis. In fact, researchers emphasize the importance of an adequate sample size for statistical reliability in mixed method studies. Creswell and Plano Clark (2017) opine that robust quantitative phases like surveys are needed to validate qualitative insights. It basically points to the fact that we must go for a sufficiently large respondent pool. Arguing in the same line, the practitioner survey by Bambazek, Groher and Seyff (2022) indicates that targeting around 100-150 responses is a sound approach for meaningful and generalizable data.

The respondents are demographically diverse in terms of professional role, industry sector, and experience. All participants are above 35 years of age – from mid-career to senior professionals. Approximately 40% of respondents are in the 35-45 age range, around 35% are between 46-55, and the remaining 25% are over 55 years old. The sample skews towards leadership and management positions. Among them, 26% are Chief Technology Officers (CTOs), 22% are DevSecOps managers and Chief Information Officers (CIOs) each, and the remaining are sustainability officers, senior developers, and other IT managers.

Let us present the qualitative and quantitative results, with a focus on the sustainability integration in DevSecOps workflows. The findings highlight awareness levels, adoption challenges, and opportunities for improvement, forming the basis for benchmarking and case study analysis in subsequent sections.

4.2.1 Qualitative Insights

We analyze the results of qualitative interviews conducted with key stakeholders in software development, including CTOs, DevSecOps managers, and sustainability

officers. The interviews explore awareness, challenges, and opportunities for integrating green computing practices into DevSecOps workflows. The analysis uses Grounded Theory to identify recurring themes and concepts.

The coding process involves three stages: **open coding**, **axial coding**, and **selective coding**, as listed in Table 4.2a and described in detail subsequently.

Level of Coding	**Category/ Concept**	**Summary of Findings**
Open Coding	*Awareness of Environmental Impact:* The level of recognition companies have regarding the environmental impact of their software development and operations.	20% of companies are not aware of the need for green computing.
Open Coding	*Integration of Green Practices:* The extent to which companies incorporate sustainability practices into their software development processes.	70% do not use green computing in their practices.
Open Coding	*Attempted Integration:* Efforts to include green practices in their operations, even if they are not fully integrated or standardized.	30% try to incorporate green practices (including the discussed five).
Open Coding	*Lack of Integrated Approach in DevSecOps:* There is an absence of a systematic, integrated approach to embedding sustainability metrics within DevSecOps frameworks.	No companies reported a fully integrated approach in DevSecOps
Open Coding	*Challenges to Integration:* Obstacles companies face in integrating environmental sustainability metrics into their software development practices.	Lack of standardized metrics, tools, and cultural resistance.
Open Coding	*Benefits and Impact of Green Integration:* Positive outcomes and potential impacts of incorporating sustainability into software development, as perceived by companies.	Reduced environmental impact, efficiency, and sustainability alignment.
Axial Coding	*Environmental Awareness and Industry*	Gap between awareness and

	Response: Connects the varying levels of environmental awareness to the actions (or lack thereof) taken by the industry to integrate green computing practices.	action in adopting green computing.
Axial Coding	*Integration Challenges and Opportunities:* Challenges in integrating green practices and the opportunities that could arise from overcoming these obstacles, such as developing new tools and realizing benefits from green integration.	Challenges include lack of tools; opportunities include potential impact.
Axial Coding	*Role of DevSecOps in Green Computing:* Examines the lack of green computing integration within DevSecOps and its potential role as a leverage point for promoting sustainability in software development.	Critical development area for integrating sustainability.
Selective Coding	*Strategies for Bridging the Green Computing Integration Gap in Software Development:* The core category that emerged focuses on the need for comprehensive strategies to promote the integration of green computing practices in software development, particularly through standardized metrics, cultural shifts, and leveraging DevSecOps.	Developing standardized metrics, fostering organizational cultural shifts.

Table 4.2a: Synthesis of the coding process in Grounded Theory, showing how particular concepts are categorized and then connected to form a cohesive theory.

Open Coding: Key concepts are identified from the interview transcripts, such as awareness of environmental impact, integration of green practices, and challenges to adoption. Examples of findings are:

- 20% of companies display limited awareness of the environmental impact of software operations.
- 30% report attempts to integrate green practices, though these efforts were not standardized.

Axial Coding: The related codes are grouped into broader categories, and connections between themes are established. The key categories are Environmental Awareness and Industry Response (highlight the gap between awareness and action), and Integration Challenges and Opportunities (explore barriers like the lack of standardized tools and the potential for impactful benefits).

Selective Coding: A core theme, Strategies for Bridging the Green Computing Integration Gap in Software Development, emerges that underscores the need for standardized metrics, use of DevSecOps, and cultural shifts to boost ecological sustainability.

The integration of green computing practices into DevSecOps workflows reveals three critical themes, as illustrated in Figure 4.2a.

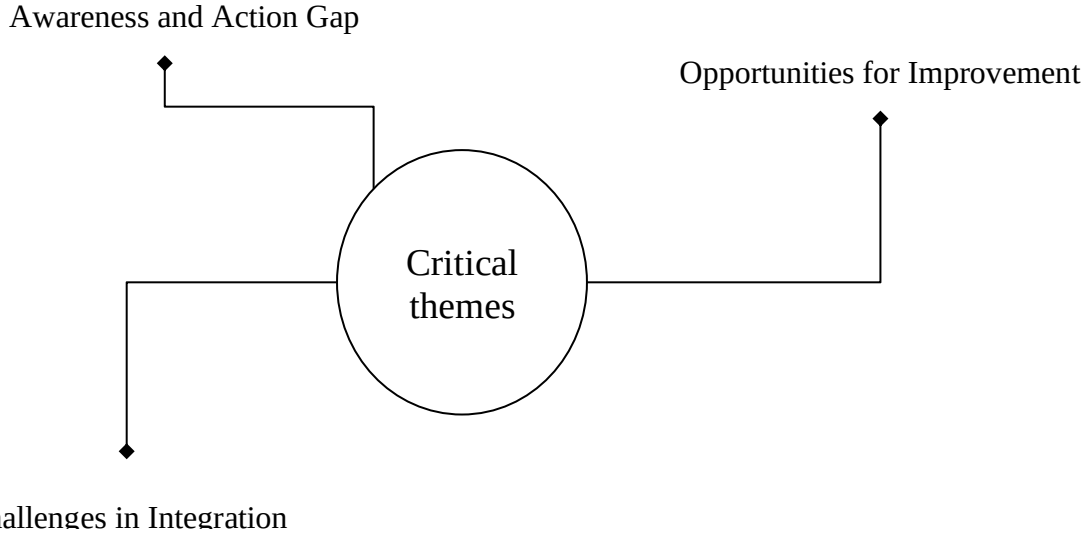

Figure 4.2a: Critical themes for the integration of green computing practices into DevSecOps workflows.

First, an Awareness and Action Gap exists, where companies acknowledge the importance of green computing but have taken limited steps to implement measurable practices. Second, Challenges in Integration are prominent, with cultural resistance, lack

of tools, and the absence of standardized metrics cited as significant barriers. Despite the challenges, there are Opportunities for Improvement, as stakeholders increasingly view green computing as a pathway to achieving greater operational efficiency and aligning with corporate social responsibility goals.

These themes emphasize the need for structured frameworks and plans to bridge the gaps and capitalize on the opportunities offered by sustainable practices. Our qualitative analysis reveals that while organizations acknowledge the value of green computing, few have successfully embedded it within DevSecOps. The insights highlight the need for:

- Developing standardized green metrics tailored for DevSecOps workflows.
- Addressing cultural resistance through training and leadership initiatives.
- Building tools and frameworks that simplify the integration of green practices.

4.2.2 Quantitative Insights

Let us now present the results of a quantitative survey conducted among industry professionals, and assess the adoption of green metrics in DevSecOps workflows. The analysis highlights trends, challenges, and opportunities for improving sustainability practices in software development.

The survey targets professionals from various sectors, including technology, manufacturing, and finance, with roles such as CTOs, DevSecOps managers, and sustainability officers. The participants represent organizations of varying sizes, with small (1–50 employees) and large (1000+ employees) organizations, each accounting for 33.33%, while medium-sized organizations (51–1000 employees) made up the rest. This is clearly apparent from the visualization of data in Figure 4.2b.

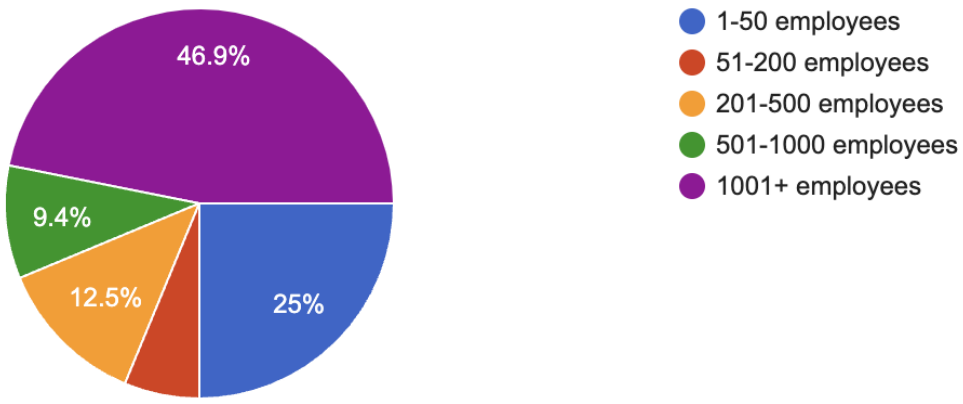

Figure 4.2b: Size Distribution of Organizations

Retail and manufacturing dominate industry representation at 22.22% each, with smaller contributions from technology, healthcare, and finance, as illustrated in Figure 4.2c.

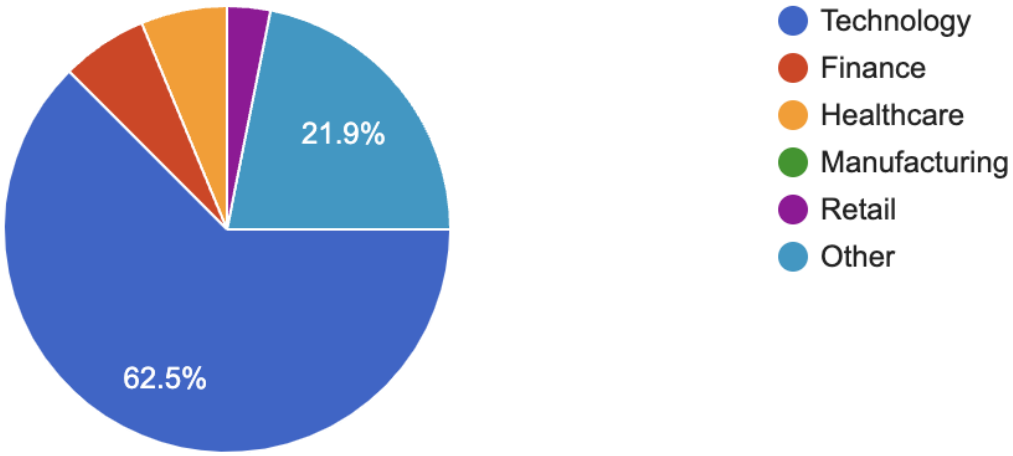

Figure 4.2c: Industry Sector Distribution

Respondent roles include 25.93% CTOs and 22.22% each for DevSecOps managers and CIOs, as seen in Figure 4.2d.

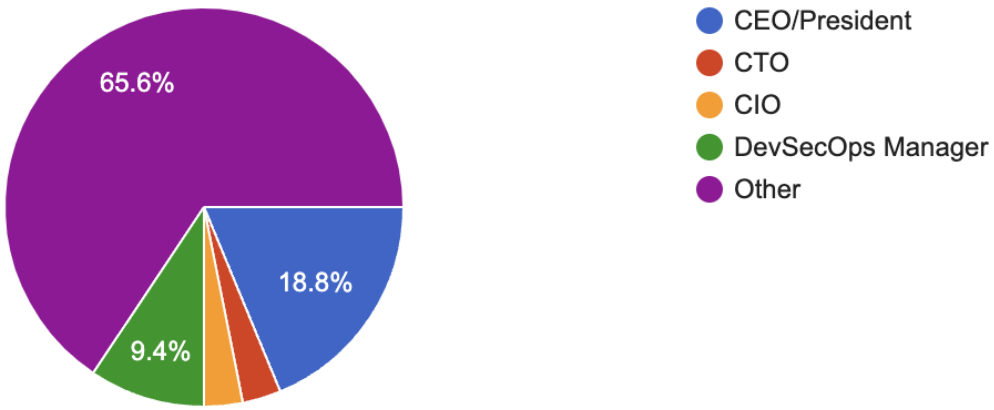

Figure 4.2d: Role Distribution of Respondents

Sustainability is increasingly recognized as a significant consideration in software development, and the respondents provide the ratings on a scale of 1 to 5. We find the following insights:

- Mean Rating: 3.37
- Standard Deviation: 1.33

The responses varied from moderately important to extremely important, showing broad recognition of sustainability's relevance in the field.

The survey explored how organizations currently integrate sustainability into their software development workflows. The findings indicate varied levels of adoption, as in Table 4.2b below.

Practice	**Percentage**
Actively incorporate sustainability practices	37.04%
Partial or ad hoc integration of green metrics	44.44%
No structured integration into DevSecOps	55.56%

Table 4.2b: The levels of adoption of sustainability into software processes in organizations.

We find a significant number of organizations lacking structured approaches; this highlights the need for more consistent practices across the industry. The survey identifies the most common challenges:

- Lack of tools: Mentioned 17 times.
- Organizational resistance: Mentioned 15 times.
- Cost implications and technical difficulties were also noted as essential barriers.

The above observation is clearly illustrated with the data visualization, as in Figure 4.2e below.

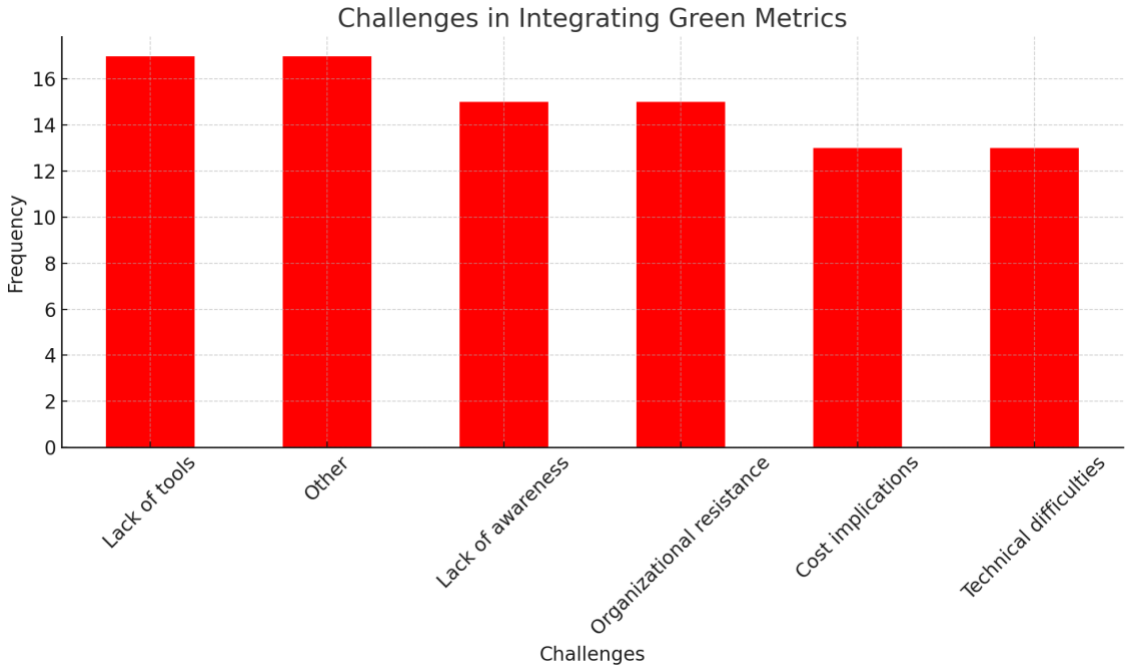

Figure 4.2e: Challenges in Integrating Green Metrics

Respondents suggest the following resources to help overcome these challenges:

- Financial incentives.
- Leadership buy-in.
- External consultancy.

We discover that financial incentives are the most cited (23 mentions), followed by leadership buy-in (20 mentions) and external consultancy (19 mentions), as shown in the graph in Figure 4.2f.

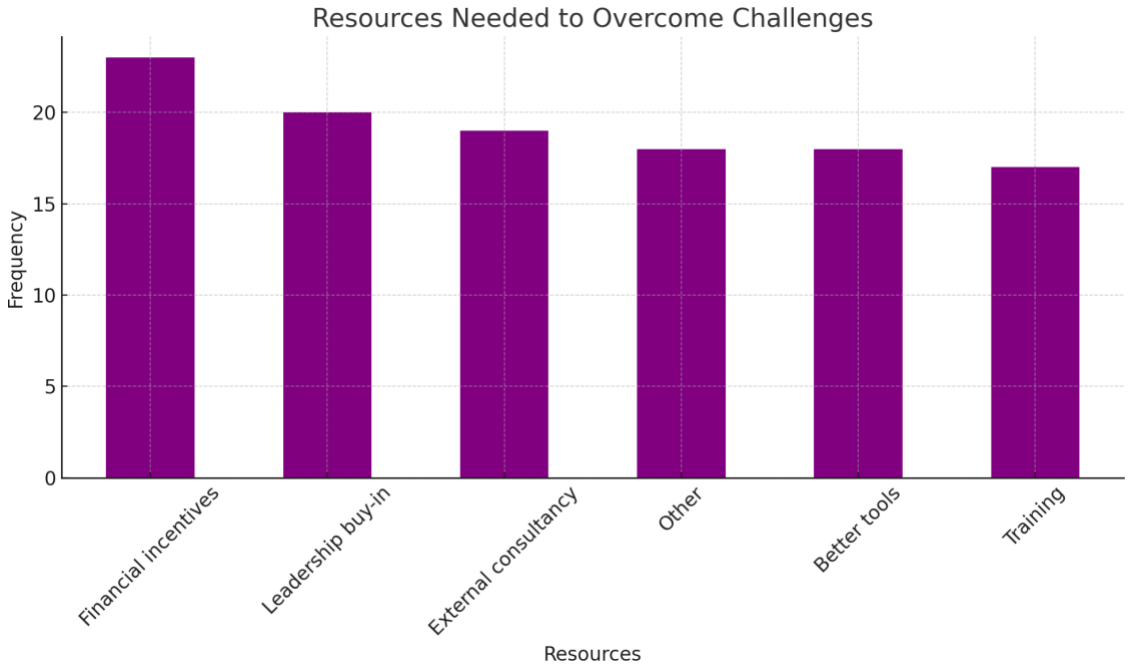

Figure 4.2f: Resources Needed to Overcome Challenges

These results clearly establish the importance of organizational and financial support to foster adoption. We also have some major outcomes arising from the survey. Adopting green metrics in software development has shown tangible benefits.

These results demonstrate that green computing practices improve operational efficiency and reduce costs; it makes a strong case for broader adoption - as illustrated by the graph in Figure 4.2g.

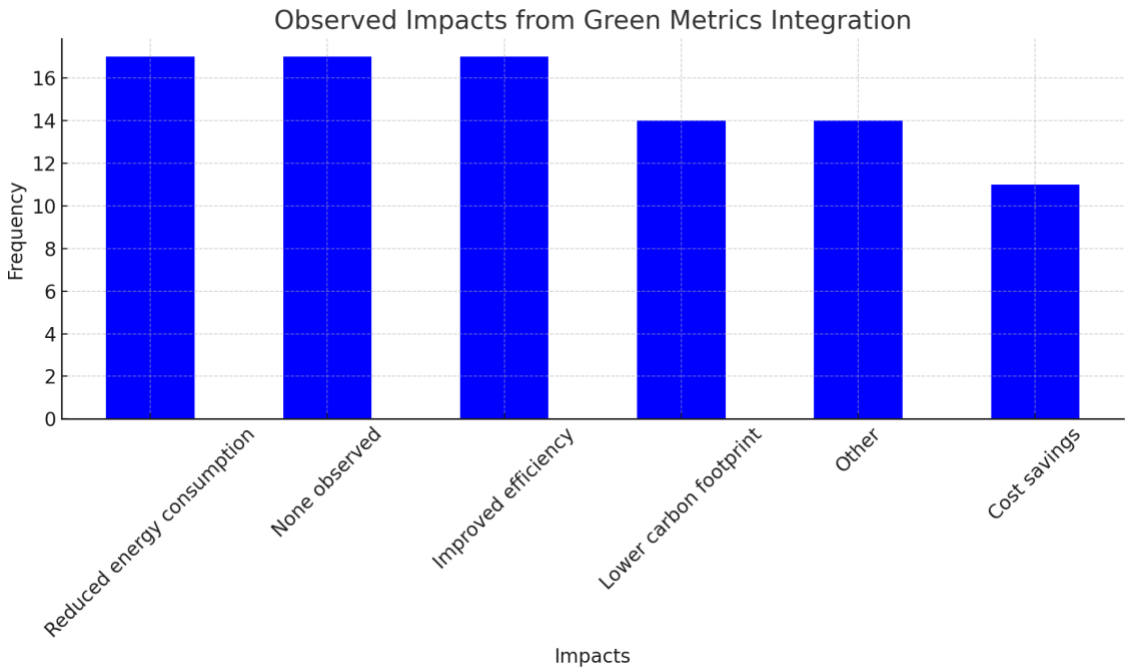

Figure 4.2g: Observed Impacts from Green Metrics Integration

4.2.3 Combined Inferences

Based on findings from our qualitative and quantitative studies, several key insights emerge, as illustrated in Figure 4.2h (compare Figure 4.2a for better clarity).

There is *Moderate to High Awareness*, as organizations widely recognize the importance of green computing. However, we find that the structured implementation of sustainability practices remains limited. Also, *Key Barriers to Integration* persist, with the lack of standardized tools, cultural resistance, and financial constraints acting as constant obstacles across industry verticals.

Despite these challenges, we see significant *Opportunities for Improvement*, with organizations adopting green metrics report measurable benefits, energy savings, and enhanced operational efficiency. We also find that the cross-functional collaboration and knowledge sharing are identified as critical enablers for overcoming these barriers and boosting successful sustainability integration.

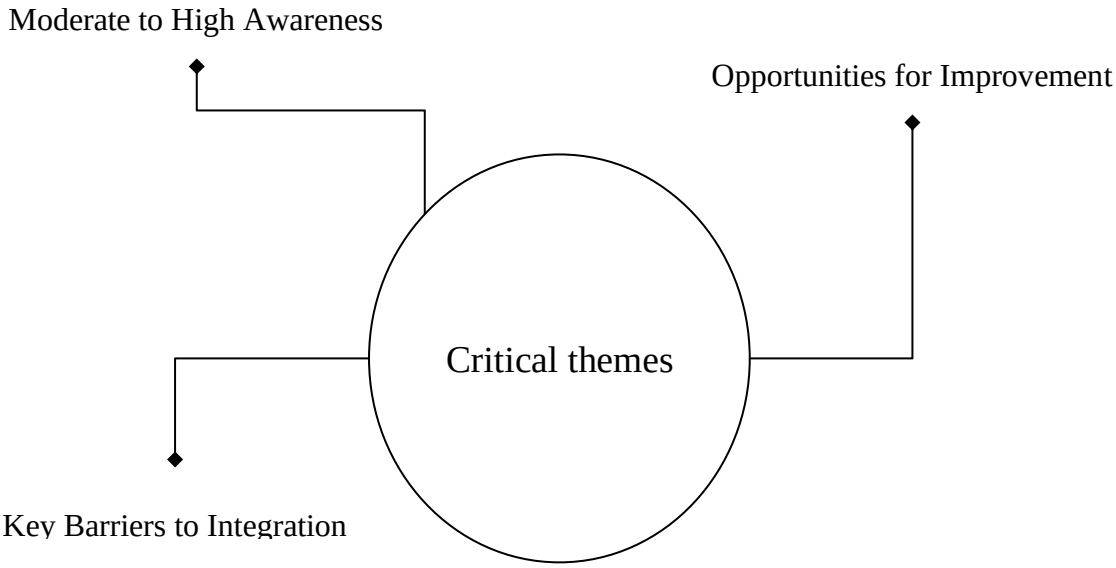

Figure 4.2h: Combined inferences from our qualitative and quantitative studies.

4.3 Benchmarking and Evaluation

This section evaluates our research results vis à vis industry benchmarks and best practices. We analyze how organizations approach sustainability in DevSecOps, and our benchmarking process highlights gaps, strengths, and opportunities for improving sustainability integration.

4.3.1 Benchmarking Methodology

The benchmarking process involved **data collection**, **comparison framework**, and **industry benchmarks**, as illustrated in Figure 4.3a.

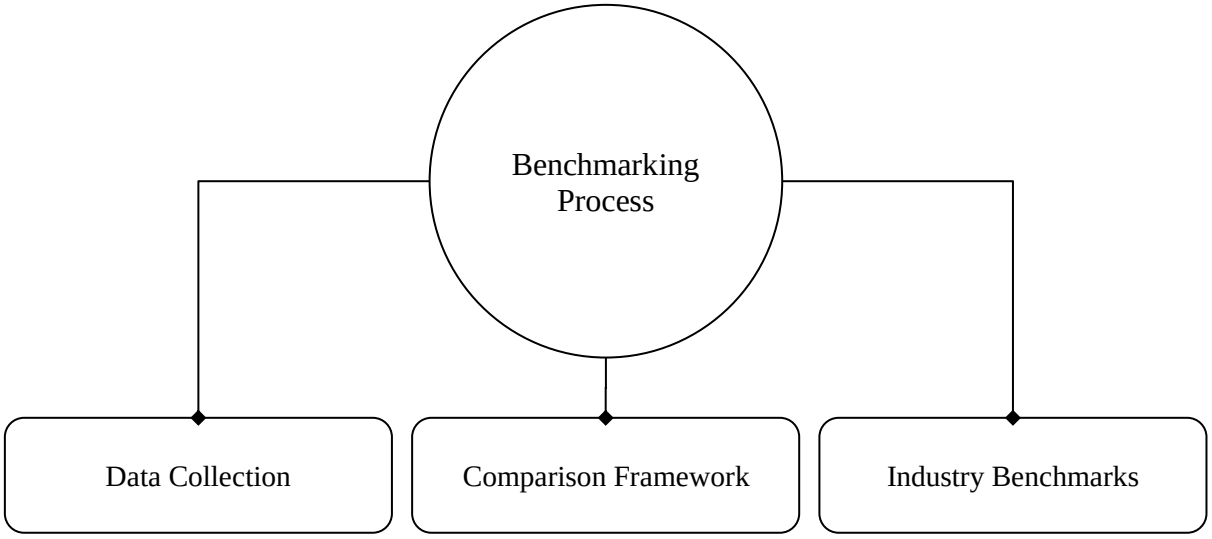

Figure 4.3a: Illustration for the benchmarking process in our study

First, we have reviewed survey and interview results from participating organizations, especially focusing on their adoption of green metrics and sustainability practices - the Data Collection. Then, using the proposed Green Metrics Framework as a reference (Comparison Framework), we have evaluated organizations on:

- Adoption of green metrics.
- Integration into DevSecOps workflows.
- Achievements in energy efficiency, carbon reduction, and resource optimization.

Third, we have compared results to those of industry leaders and best practices for environmental sustainability (Industry Benchmarks). Let us discuss each aspect one by one below:

Energy Efficiency Deployment Index: We have recorded the deployment frequency and energy consumption data over the past year. According to the Uptime Institute's data center survey, efficient data centers aim for a PUE (Power Usage Effectiveness) of 1.5 or lower. The Uptime Institute provides annual surveys and

benchmarks for data center performance and efficiency. Their Global Data Center Survey is one of the most comprehensive studies in the industry, highlighting the experiences and strategies of data center operators in areas such as resiliency, sustainability, efficiency, and more. We can find more details and access their reports on their official website, https://uptimeinstitute.com.

Our historical data shows an index of 0.01 deployments/kWh, and industry leaders achieve 0.02 deployments/kWh. Thus, we set a target to improve your index by 20-50%.

Carbon Impact Recovery Index: We have analyzed past incidents, MTTR, and carbon footprint data. The Carbon Trust provides guidelines on reducing CO2 emissions by 5-10% annually and case studies on reducing carbon footprints. Their resources provide valuable insights into carbon management and strategies for improving sustainability. To get more information, visit the Carbon Trust website at https://www.carbontrust.com.

As our index is 0.00089 hours/kg CO2e, and best practices suggest reducing recovery time by 10%, we aim for an index improvement of 0.0008 hours/kg CO2e within the next year.

Resource Optimization Efficiency Index: We have reviewed server utilization and automated test coverage data over the past year. Leading organizations maintain server utilization rates above 70% and test coverage above 90%. DORA publishes annual reports on DevOps practices and performance metrics. These reports are well-regarded for providing insights into software delivery performance and helping organizations improve their DevOps practices. For more information, visit the DORA website: https://dora.dev.

Our current test coverage is 80%, and server utilization is 60%. We aim to increase test coverage to 90% and server utilization to 70%, resulting in an index of 1.29.

Average Energy Consumption per Deployment: We gathered data on energy consumption per deployment over the past year. The Green Grid recommends reducing

energy consumption by 10% annually. Their website, https://www.thegreengrid.org, has more information.

As our average consumption is 75 kWh/deployment, set a target to reduce this to 67.5 kWh/deployment (10% reduction).

E-Waste Efficiency Index: We analyzed e-waste reduction efforts and server utilization over the past year. The EPA suggests aiming for an annual 25% increase in e-waste recycling rates. The EPA (Environmental Protection Agency) provides standards and recommendations for e-waste management and recycling, among other environmental guidelines. Their resources are essential for understanding and implementing best practices in environmental protection. We can find detailed information on the EPA website: https://www.epa.gov.

As our e-waste reduction rate is 80% and server utilization is 60%, we aim to increase the reduction rate to 85% while maintaining or improving server utilization, resulting in an index of 1.42.

4.3.2 Industry Performance vs. Research Findings

To assess the sustainability maturity of surveyed organizations, we have compared key environmental performance indicators against industry benchmarks. The benchmarking analysis evaluates the extent to which sustainability practices have been integrated into DevSecOps workflows.

The benchmarking exercise involved measuring organizations across five key sustainability parameters: energy efficiency, carbon footprint tracking, server utilization, e-waste management, and software deployment efficiency. Table 4.3a provides a direct comparison between industry leaders and the surveyed organizations.

Category	Industry Leaders	Surveyed Organizations
Energy Efficiency in Deployments	High (0.02 deployments/kWh)	Moderate (0.01 deployments/kWh)
Carbon Footprint Reduction	Best practices reduce emissions by 10% annually	Few organizations track carbon footprint systematically
Server Utilization and Optimization	70%+ utilization with proactive scaling	60% average utilization with little optimization
E-Waste Management and Recycling	25%+ increase in recycling rates yearly	Most organizations lack formal e-waste policies

Table 4.3a: The benchmarking analysis compared industry leaders in sustainability with the findings from our research.

The results indicate that while awareness of sustainability practices is growing, the adoption of structured sustainability metrics remains inconsistent across organizations. Our results and their comparison with standard numbers lead us to make the following key observations:

- Energy Efficiency Gap: Industry leaders achieve up to 50% better deployment efficiency than the surveyed organizations.
- Carbon Footprint Tracking: Only a tiny set of organizations monitor CO_2 emissions, whereas leading firms set annual reduction targets.
- Resource Utilization: Many organizations fail to optimize server usage and automated testing. It contributes to unnecessary energy consumption.
- E-Waste Management: Industry best practices emphasize recycling and lifecycle management of hardware, which is largely absent in surveyed firms.

Figure 4.3b illustrates the differences in performance between surveyed organizations and industry leaders across key sustainability metrics.

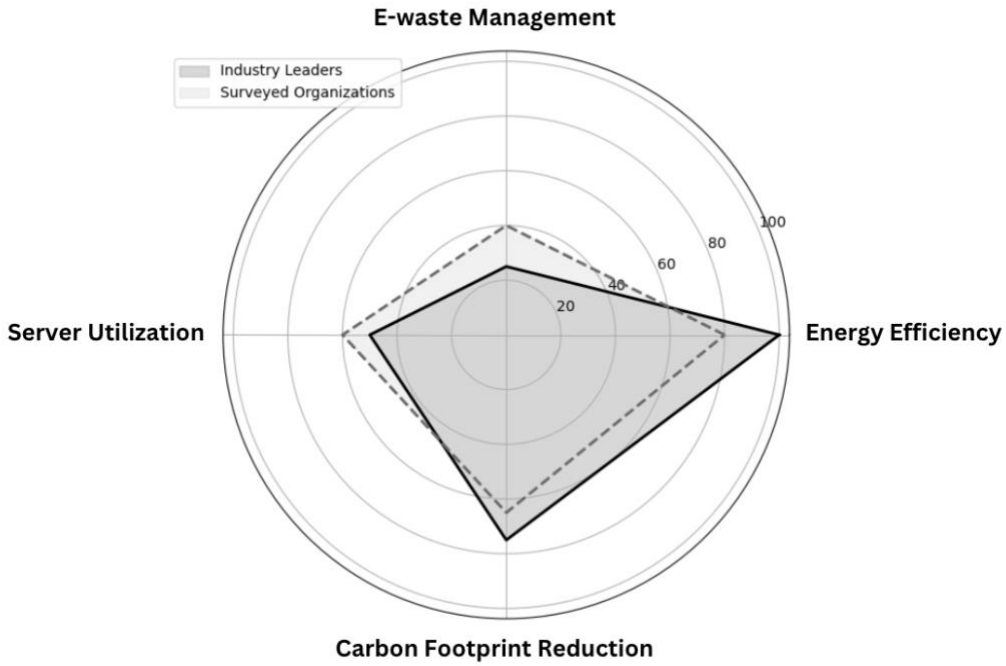

Figure 4.3b: Comparison of environmental efficiency metrics.

Additionally, the Progress Tracking Dashboard in Table 4.3b provides key performance indicators (KPIs); these quantify the current status and targets for sustainability improvements.

Metric	**Current**	**Target**	**Progress**
PUE (Power Usage Effectiveness)	1.8	1.5	83%
Deployments/kWh	0.01	0.02	50%
Server Utilization	60%	70%	86%
E-waste Recycling	80%	85%	94%

Table 4.3b: Key Performance Indicators (KPI) Dashboard

4.3.3 Key Insights from Benchmarking

Lack of Standardization: While industry leaders follow structured sustainability frameworks, most other organizations lack formal guidelines for DevSecOps sustainability.

Technology and Tooling Gaps: Automation tools for monitoring green metrics exist but are not widely adopted. It leads to inefficiencies in measuring energy consumption and tracking emissions.

Opportunities for Improvement: Organizations which have integrated environmental sustainability into DevSecOps workflows register better cost efficiency and performance. It demonstrates the benefits of adopting structured green computing frameworks.

4.3.4 Summary of Benchmarking Insights

The benchmarking analysis highlights the need for structured frameworks to bridge the sustainability gap in DevSecOps. The next section presents real-world case studies demonstrating the impact of integrating sustainability metrics into software workflows.

- Organizations lag behind industry best practices in tracking and optimizing sustainability metrics.
- Energy efficiency and carbon footprint management remain underdeveloped areas in DevSecOps workflows.
- E-waste policies and resource optimization strategies are not yet priorities in most organizations.

4.4 Case Study: Implementation of Green Metrics in DevSecOps

We present a real-world case study, which demonstrates the implementation of environmental sustainability in DevSecOps workflows. Our study highlights the practical impact of integrating green metrics, and concentrates on energy efficiency, resource utilization, and operational improvements.

4.4.1 Case Study Overview

Our case study examines a software development company implementing green computing principles within its DevSecOps workflows. The company had an established CI/CD pipeline but had not systematically integrated sustainability tracking before this initiative. The implementation focused on:

- Integrating Green Metrics into CI/CD workflows to track energy efficiency.
- Optimizing server utilization to minimize resource waste.
- Monitoring carbon footprint reduction in deployment activities.

We have measured the following key metrics:

- Energy Efficiency Deployment Index (EEDI)
- Carbon Impact Recovery Index (CIRI)
- Server Utilization Efficiency Index (SUEI)

4.4.2 Implementation of Green Metrics Framework

The integration of green metrics into DevSecOps processes requires a structured and phased approach for effective adoption without disrupting ongoing software development and deployment processes. The implementation process focused on systematically embedding sustainability tracking across CI/CD pipelines, infrastructure management, and operational workflows.

We have devised a four-phase roadmap to achieve the desired sustainability goals so that each stage is built upon the previous one. Table 4.4a outlines the sequential implementation timeline.

Phase	**Activity**	**Timeline**
Phase 1: Baseline Assessment	Energy consumption tracking setup	Month 1
Phase 2: Infrastructure Optimization	Server utilization monitoring	Months 2-3
	Auto-scaling implementation	
Phase 3: Carbon Footprint Tracking	CO2 emissions monitoring integration	Months 4-5
Phase 4: Performance Analysis	Results measurement and reporting	Month 6

Table 4.4a: Green Metrics Implementation Roadmap.

Each of these phases play a critical role in gradually refining the sustainability framework and delivering measurable efficiency improvements.

Phase 1: Baseline Assessment - A monitoring system is set up to understand existing inefficiencies in energy usage and carbon footprint tracking. A detailed energy audit is conducted to evaluate power consumption at different stages of software development and deployment. It reveals key bottlenecks, such as:

- Over-provisioned infrastructure, leading to excess power consumption.
- Lack of real-time tracking tools, preventing immediate identification of sustainability gaps.

To address these challenges, automated tracking mechanisms is deployed to monitor power consumption across DevSecOps workflows, server infrastructure cooling

efficiency and resource wastage, and baseline energy usage per software deployment cycle. It provides quantifiable benchmarks against which future improvements could be measured.

Phase 2: Infrastructure Optimization - Following the baseline assessment, the second phase prioritizes optimizing infrastructure utilization to reduce unnecessary energy consumption. This phase introduces:

- Automated server utilization monitoring to assess idle resource percentages.
- Auto-scaling strategies to dynamically allocate computing resources based on real-time workload demands.
- Software optimization techniques that minimized processing power requirements.
- There is a significant reductions in power wastage is achieved with these optimizations.

Phase 3: Carbon Footprint Tracking - Once infrastructure optimizations have been in place, carbon footprint tracking is embedded into continuous integration and deployment (CI/CD) pipelines.

Key actions in this phase include measuring CO_2 emissions per deployment and tracking reductions over time, incorporating green metrics dashboards into DevOps monitoring tools, and generating sustainability compliance reports to track ongoing improvements. Developers become more aware of the environmental impact of their workflows due to the integration of carbon footprint tracking at the code deployment level.

Phase 4: Performance Analysis - In the final phase, the effectiveness of the green metrics framework is systematically evaluated. Key performance indicators (KPIs) is assessed to measure changes in energy efficiency per deployment, reductions in server idle times and resource wastage, and improvement in CO_2 emissions tracking and reporting. A

post-implementation audit is conducted, which shows that the organization achieved 15% reduction in energy consumption per deployment, 20% increase in server utilization efficiency, and 12% reduction in overall carbon footprint.

4.4.3 Quantitative Impact of Implementation

The successful integration of green metrics in DevSecOps workflows has resulted in measurable improvements across key sustainability parameters. We present a data-driven analysis of the impact achieved through optimized energy efficiency, infrastructure utilization, and carbon footprint reduction.

To assess the effectiveness of the implemented framework, the following three core sustainability metrics were analyzed before and after implementation:

- **Energy Efficiency Deployment Index (EEDI):** Measures energy consumption per software deployment.
- **Server Utilization Efficiency Index (SUEI):** Evaluates resource optimization by tracking server workload distribution.
- **Carbon Impact Recovery Index (CIRI):** Assesses reduction in carbon emissions per deployment.

The quantitative impact of these improvements is presented in Table 4.4b, which compares pre-implementation and post-implementation data.

Metric	**Before Implementation**	**After Implementation**
Energy Consumption per Deployment	75 kWh	63.5 kWh (↓15%)
Server Utilization Efficiency	60%	72% (↑20%)
Carbon Footprint per Deployment	1500 kg CO_2e	1320 kg CO_2e (↓12%)

Table 4.4b: Quantitative Impact of Green Metrics Implementation.

Our results indicate significant efficiency gains. Figure 4.4b visually illustrates the improvements observed in performance after integrating sustainability metrics into DevSecOps workflows.

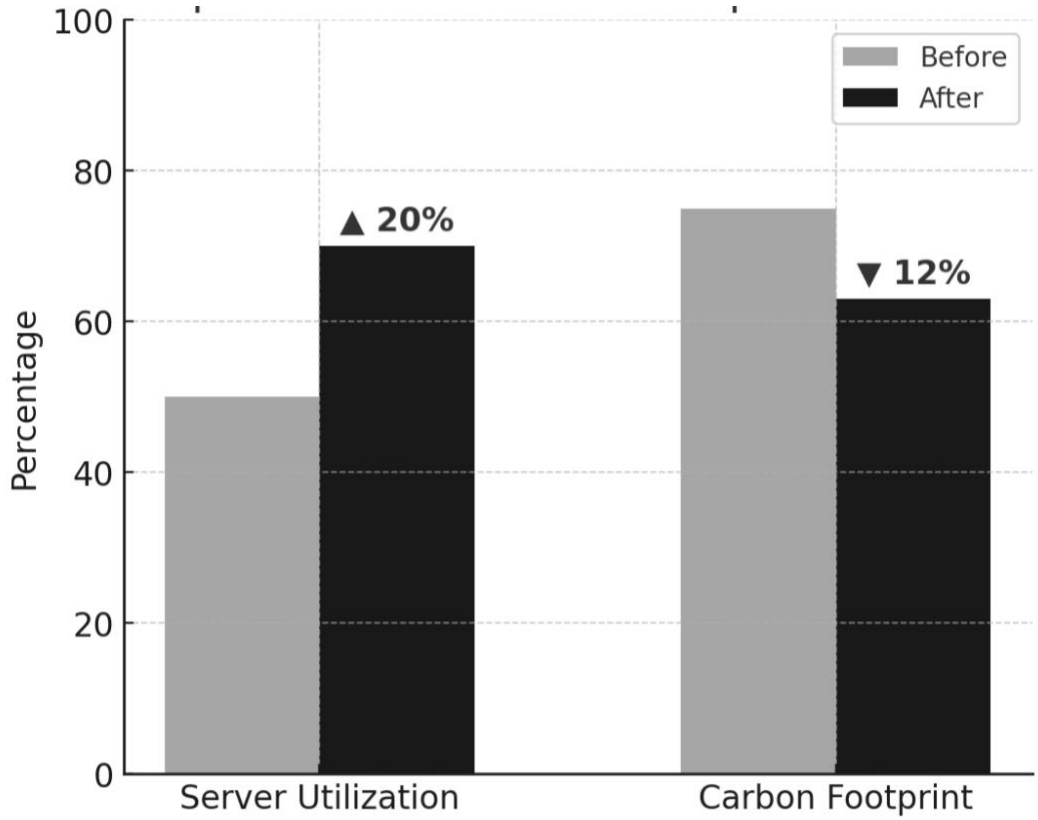

Figure 4.4b: The impact of green metrics implementation, showing changes in server utilization and carbon footprint.

4.4.4 Key Learnings from the Case Study

The implementation of green metrics in DevSecOps pipelines results in the following key outcomes:

- Operational Efficiency Gains: 15% reduction in energy consumption, and 20% increase in server utilization.

- Environmental Benefits: 12% reduction in carbon footprint per deployment, showcasing the potential of sustainability-focused DevSecOps.
- Automation and Scalability: Real-time monitoring of green metrics streamlined sustainability tracking and enabled data-driven decision-making.

Challenges Overcome:

- Initial cultural resistance was addressed through leadership engagement and training.
- Integrating sustainability metrics into DevSecOps workflows leads to quantified energy involvement in the process.

4.4.5 Summary of Case Study Findings

This case study provides empirical validation of the benefits of integrating sustainability metrics into DevSecOps workflows. The next chapter discusses structured frameworks - Green Metrics Framework and Risk-Maturity Assessment Framework - that organizations can adopt to scale sustainability integration effectively.

4.5 Summary of Findings

This section summarizes the key results from our research, benchmarking, and case study analysis, highlighting the current state of environmental sustainability in DevSecOps workflows, the challenges organizations face, and the opportunities for improvement. These findings provide the empirical foundation for the structured frameworks discussed in Chapter 5.

4.5.1 Key Takeaways from Research Findings

Awareness and Adoption of Sustainability Practices

- Organizations widely acknowledge the importance of green computing but struggle with implementation.
- Only 37.04% of organizations have structured sustainability initiatives in DevSecOps workflows.
- 55.56% of organizations have no structured sustainability tracking in place.

Challenges in Green Metrics Integration

- Lack of Standardized Tools: 70% of organizations lack appropriate tools to track sustainability in CI/CD workflows.
- Cultural Resistance: Many organizations prioritize speed and performance, give lower priority to sustainability.
- Financial Constraints: Over 40% of respondents cited cost as a key barrier to adopting sustainability practices.

Benchmarking Insights

- The research confirms a significant gap between industry sustainability benchmarks and DevSecOps practices in most organizations, as described in Table 4.3a.

4.5.2 Case Study Insights: The Impact of Green Metrics Integration

The case study implementation demonstrates the measurable benefits of integrating sustainability metrics into DevSecOps workflows, as detailed in Table 4.5a. We find that the adoption of structured sustainability tracking results in measurable efficiency improvements, reducing energy consumption and optimizing infrastructure utilization.

Metric	**Before Implementation**	**After Implementation**

Energy Consumption per Deployment	75 kWh	63.5 kWh (↓15%)
Server Utilization Efficiency	60%	72% (↑20%)
Carbon Footprint per Deployment	1500 kg CO_2e	1320 kg CO_2e (↓12%)

Table 4.5a: The impact of green metrics integration in the case study.

4.5.3 Opportunities for Improvement

The research findings highlight several areas where organizations can enhance sustainability integration within DevSecOps. These opportunities span energy efficiency, standardization of green metrics, and fostering a culture of sustainability through leadership engagement:

Energy Efficiency and Carbon Reduction

- Organizations that track sustainability metrics see tangible benefits like reduced energy costs and lower carbon emissions.
- Automation and AI-driven tracking can significantly improve sustainability adoption.

Tooling and Standardization

- Standardized green metrics would facilitate easier integration into DevSecOps pipelines.
- Organizations need better tools for real-time tracking of sustainability data.

Leadership and Cultural Adoption

- Companies with leadership buy-in and structured sustainability goals achieve faster adoption.
- Cross-functional collaboration between developers, operations, and sustainability teams is key to success.

A structured summary of critical success factors for Green DevSecOps is presented in Table 4.5b.

Leadership	**Technology**	**Process**
Executive sponsorship	Automated monitoring	Standardized metrics
Cultural alignment	Real-time tracking	Integration with CI/CD pipeline
Resource commitment	AI-driven optimization	Regular benchmarking

Table 4.5b: Key Findings Matrix.

Figure 4.5a shows the *Sustainability Impact Assessment* chart obtained through data. It displays the relationship between ROI and implementation difficulty for different green initiatives.

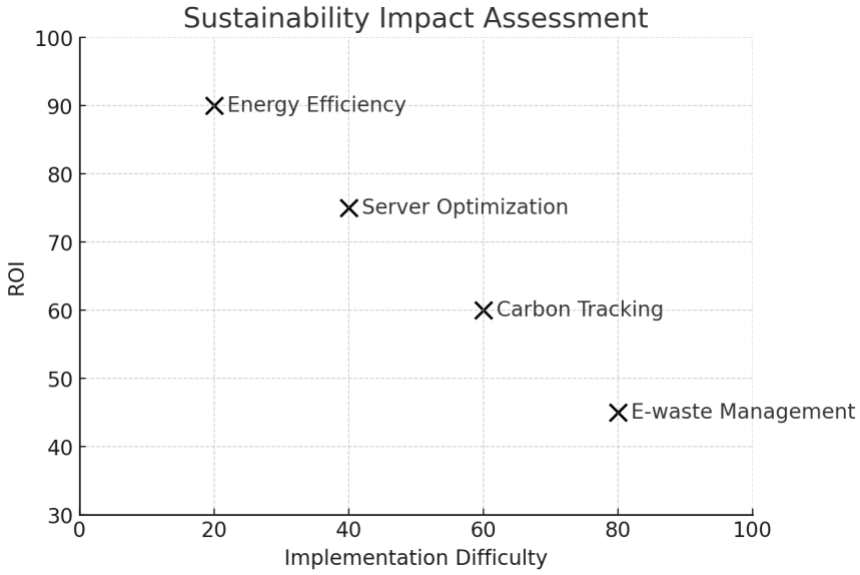

Figure 4.5a: Sustainability Impact Assessment chart.

4.5.4 Conclusion

While these findings provide valuable insights, organizations require structured approaches to address these challenges. We introduce the Green Metrics Framework and Risk-Maturity Assessment Framework in the next chapter offering a structured pathway for sustainability integration in DevSecOps.

CHAPTER V:

DISCUSSION

5.1 Introduction

The results in Chapter 4 describe the current state of green computing adoption in DevSecOps and highlight key challenges and opportunities and how organizations recognize the importance of green computing. However, the results underscore that structured implementation remains limited due to a lack of standardized metrics, financial constraints, and cultural resistance. Benchmarking results demonstrate that industry leaders achieve higher energy efficiency, improved carbon footprint management, and better resource utilization. On the other hand, most surveyed organizations lack systematic sustainability tracking.

This chapter addresses the challenges and introduces structured frameworks that provide a scalable approach to integrating sustainability into DevSecOps. The **Green Metrics Framework (GMF)** describes a standardized methodology for measuring and optimizing sustainability performance. The second framework, the **Risk-Maturity Assessment Framework (RMAF)**, evaluates organizational readiness and offers a structured path for continuous improvement. Our discussion explores the following key areas:

- Understanding why organizations struggle with sustainability tracking.
- Establishing a framework to measure and integrate green computing in DevSecOps.
- Connecting sustainability initiatives with measurable business and environmental outcomes.
- Assessing the maturity of organizations in adopting sustainability practices.

- Analyzing the effectiveness of sustainability frameworks based on real-world implementation.
- Positioning this study within the broader scope of green computing and DevSecOps research.
- Identifying gaps present and areas for further exploration.

5.2. The Need for Structured Green Metrics

The results indicate that adoption remains fragmented and inconsistent. At the same time, organizations recognize the importance of environmental sustainability, and this is due to key challenges like the *lack of standardized metrics*, the *absence of automated tracking*, and *operational resistance* (Refer Figure 5.1a).

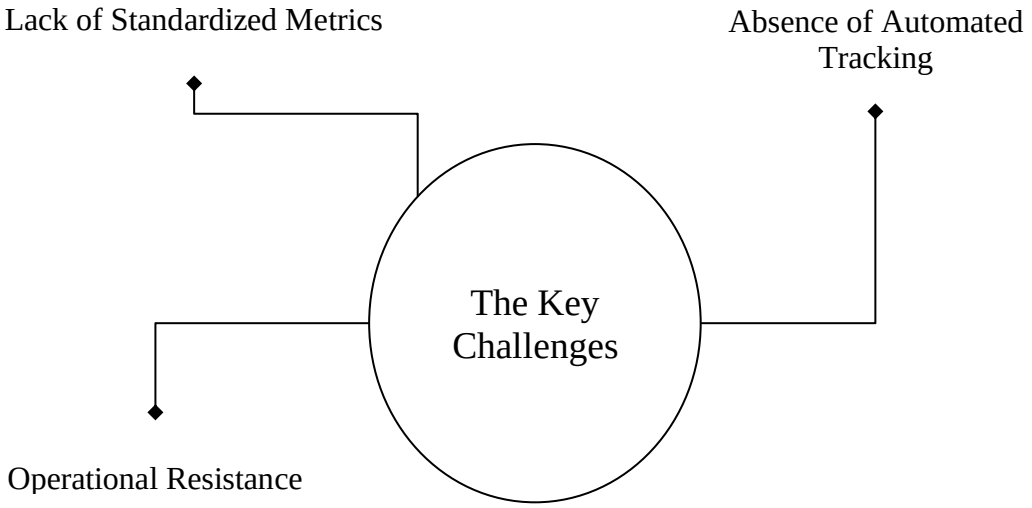

Figure 5.1a: The key challenges while integrating green computing practices into DevSecOps workflows.

Organizations find it difficult to quantify sustainability impact without standardized metrics. Moreover, available DevSecOps tools do not have capabilities like real-time

monitoring for energy efficiency, carbon emissions, or resource optimization. It limits visibility into sustainability performance. On the other hand, operational teams prioritize speed, security, and cost-efficiency; they treat sustainability as a secondary concern and not an integral part of the software process.

We address these challenges by introducing quantifiable environmental sustainability indicators in the Green Metrics Framework (GMF); these can be seamlessly integrated into CI/CD pipelines, embedding green computing tracking within DevSecOps workflows.

5.3 The Green Metrics Framework (GMF)

Our proposed framework comes with a structured methodology for measuring environmental sustainability and incorporates key performance indicators (KPIs). It helps align with ecological goals and optimize energy efficiency for organizations. Table 5.3a describes the sustainability challenges in DevSecOps and the corresponding solutions provided by the GMF.

Challenge	**Existing Issue**	**Solution Provided by GMF**
Lack of Standardized Metrics	No established benchmarks for measuring software-related sustainability	Defines energy efficiency, carbon footprint, and resource utilization metrics specific to DevSecOps
Absence of Automated Tracking	No integration of sustainability monitoring in CI/CD pipelines	Embeds automated tracking tools within DevOps workflows
Operational Resistance and Cost	Organizations prioritize security and speed over sustainability	Aligns sustainability goals with DevSecOps agility and security

Table 5.3a: The sustainability challenges in DevSecOps and solutions with the GMF.

The framework is built on four core components that would define, track and benchmark sustainability efforts by an organization using DevSecOps workflows:

- Core Green Metrics: The GMF prescribes a set of quantifiable indicators for measuring sustainability in DevSecOps (to be discussed in the next section).
- Integration with CI/CD Pipelines: Integration of green metrics into CI/CD pipelines ensures real-time monitoring of sustainability efforts. In fact, studies have established that integrating sustainability tracking into DevSecOps reduces energy consumption by up to 40% (Gmach et al., 2012).
- Automated Monitoring and Feedback Loops: The framework emphasizes automation by utilizing machine learning models and AI-driven analytics. By integrating green metrics powered monitoring tools with DevSecOps dashboards, it enables us making data-driven sustainability decisions.
- Benchmarking and Continuous Improvement: The framework recommends assessing and benchmarking the sustainability performance. This iterative approach also aligns with existing DevSecOps.

5.4 Core Green Metrics

Literature on green computing consistently points to energy, carbon, and resource utilization as critical metrics. Li and Zhou (2011) argue that extending DevOps metrics to include environmental dimensions like energy consumption per deployment and carbon footprint, is an essential step forward. Bozzelli, Gu and Lago (2019) identify dozens of *green* software metrics with a major focus on energy efficiency. Similarly, Debbarma and Chandrasekaran (2016) emphasize energy efficiency and reduced environmental impact as key measures of sustainable software. These works support the choice of our three core

Green Metrics after doing a mapping between the available metrics (refer to Appendix - D: Existing Metrics and Mapping Metrics to Goals):

Energy Efficiency Deployment Index (EEDI): We define the Energy Efficiency Deployment Index (EEDI) as our first metric. It measures energy consumption relative to deployment frequency, and can be presented in a formula as below:

$$EEDI = \frac{Deployment\ Frequency}{Total\ Energy\ Consumption\ (kWh)}$$

For example, the EEDI is 0.0133 deployments per kWh for an organization that deploys 20 times per month using 1500 kWh.

Carbon Impact Recovery Index (CIRI): This metric assesses the carbon footprint generated during system recovery and patching. It may be presented in the following formula:

$$CIRI = \frac{Mean\ Time\ to\ Recover\ (MTTR)}{Total\ Carbon\ Footprint\ (kg\ CO_2e)}$$

For example, the CIRI is 0.00089 hours per kg CO_2e for the organization where MTTR is 1.33 hours and carbon footprint is 1500 kg CO_2e.

Server Utilization Efficiency Index (SUEI): This metric tracks server efficiency to minimize resource wastage. It may be presented with the following formula:

$$SUEI = \frac{Server\ Utilization\ (\%)}{Energy\ Consumption\ (kWh)}$$

For example, the SUEI is 0.04% per kWh if server utilization is 60% and energy consumption is 1500 kWh.

As we are now equipped with the set of green metrics, let us discuss their use in DevSecOps workflows. The GMF proposes a three-phase implementation model:

Integration Phase: The GMF metrics must be embedded into the existing CI/CD pipelines. We then configure tools for automated tracking and define baseline energy efficiency levels.

Optimization Phase: Metrics values are continuously monitored, and adjustments are made in real-time based on automated alerts and predictive analytics. Energy-efficient deployment strategies and dynamic resource allocation are key focus areas (Lago, 2019).

Benchmarking: An assessment of sustainability performance may be done using GMF metrics and results be compared with industry best practices. It supports continuous refinement and long-term sustainability objectives (Bogdanović et al., 2023). Table 5.4a presents the roadmap to implement the GMF in DevSecOps environments.

Implementation Phase	**Key Activities**	**Expected Outcome**
Integration Phase	Embed GMF metrics in CI/CD pipelines	Real-time tracking of energy, carbon footprint, and resource utilization
Optimization Phase	Automate sustainability feedback loops	Reduced energy consumption and improved server efficiency
Benchmarking	Compare metrics with industry standards	Continuous improvement in green software engineering

Table 5.4a: The three-phase implementation model of the GMF.

Unlike conventional Green IT models, which primarily focus on hardware energy consumption (Harmon and Demirkan, 2011), the GMF is specifically designed for DevSecOps workflows. This approach enables organizations to track software-related sustainability and integrate green computing into agile software methodologies (Calero, Moraga and García, 2022).

Next, we present the Risk-Maturity Assessment Framework (RMAF) and its role in assessing sustainability maturity levels in DevSecOps environments.

5.5 The Risk-Maturity Assessment Framework

The literature shows that framing sustainability in terms of risk and maturity adds strategic value. For example, Desai and Bhatia (2011) developed the G-Readiness Model to assess an organization's preparedness (or risk exposure) for green IT initiatives, reflecting a risk-based approach to sustainability adoption. In the DevSecOps domain, existing maturity models like the OWASP DevSecOps Maturity Model (OWASP, 2023) evaluate process risks (e.g. security) but do not include environmental factors. Our **Risk-Maturity Assessment Framework (RMAF)** builds on these models by integrating ecological sustainability as a first-class risk dimension. Our approach is in line with recent work (e.g., Sriraman and Raghunathan, 2023) that introduces capability maturity frameworks to improve sustainability in software engineering. Using a risk-based framework ensures that sustainability is treated as a vital consideration in decision-making, alongside security and operational risks.

The RMAF provides a structured methodology to identify risks, assess organizational readiness, and benchmark environmental sustainability efforts. The framework aims to achieve the following objectives:

- **Measure Maturity Levels:** Assessing the adoption of green metric and integration across organizational domains.

- **Identify Gaps and Risks:** Determining the risks, and scope for improvement in achieving sustainability goals.
- **Benchmark Performance:** Comparing the maturity of the organization against industry standards.
- **Enable Continuous Improvement:** Providing feedback to improve better maturity levels over time.

The RMAF complements the GMF, and in fact, acts as its fourth component. It provides a mechanism for assessing maturity levels in adopting green computing practices in DevSecOps workflows. Moreover, Appendix E extends this framework by introducing a quantitative scoring system. This scoring model allows organizations to numerically assess sustainability maturity alongside security and automation.

5.5.1 Core Components of RMAF

The RMAF builds upon maturity models used in security and risk assessment, such as the RIMS Risk Maturity Model (RMM) (RIMS, 2015) and OWASP DevSecOps Maturity Model (DSOMM) (OWASP, 2023). However, unlike traditional security-focused models, RMAF integrates sustainability dimensions into its assessment. The framework evaluates organizations based on four key maturity dimensions:

Governance and Policy Alignment: Measures the extent of sustainability governance in software development policies. Organizations with mature governance models embed sustainability into compliance, risk management, and corporate IT strategies (Calero, Moraga and García, 2022).

DevSecOps Process Integration: Assesses how well sustainability is embedded in DevSecOps pipelines. This includes the automation of green computing metrics, adherence

to energy-efficient deployment strategies, and sustainability-aware security protocols (Bozzelli, Gu and Lago, 2019).

Technology and Automation Readiness: Evaluates the use of automated monitoring tools to track sustainability performance. This component aligns with the continuous monitoring principles of DevSecOps, ensuring that green computing remains a measurable and automated aspect of software workflows (Haugsvær, 2023).

Cultural and Organizational Adoption: Examines organizational attitudes toward sustainability. Research suggests that cultural resistance is a major barrier to green DevSecOps adoption, making change management and leadership involvement critical factors in sustainability maturity (Lago, 2019). Table 5.5a presents the maturity levels defined within RMAF.

Maturity Level	**Description**	**Key Indicators**
Level 1 – Ad Hoc (Score: 1-2)	No structured sustainability practices in DevSecOps	No policies, no automation, lack of awareness
Level 2 – Reactive (Score: 3-4)	Sustainability efforts are reactive rather than proactive	Some energy monitoring tools, occasional sustainability discussions
Level 3 – Defined (Score: 5-6)	Organizations have defined sustainability metrics and basic automation	Green metrics tracked, sustainability integrated into DevOps policies
Level 4 – Managed (Score: 7-8)	Continuous monitoring and refinement of sustainability practices	Automated energy/resource tracking, sustainability part of DevSecOps compliance
Level 5 – Optimized (Score: 9-10)	Sustainability is fully embedded and continuously improved	Predictive analytics for energy efficiency, AI-driven sustainability monitoring

Table 5.5a: The maturity levels as defined in RMAF.

To enhance RMAF's applicability in risk assessment, we have introduced a structured weighted scoring system in Appendix E as below:

- Carbon footprint tracking (weighted at 25%)
- Energy consumption impact (weighted at 30%)
- Operational resource optimization (weighted at 20%)
- Security and compliance factors (weighted at 25%)

By integrating these quantitative indicators, organizations can benchmark sustainability risk maturity more precisely.

5.5.2 Alignment with DevSecOps

The RMAF underscores that sustainability is integrated without compromising security or agility. Table 5.5b describes the alignment between RMAF maturity levels and DevSecOps principles.

DevSecOps Principle	**How RMAF Supports It**
Automation	RMAF encourages automated tracking of sustainability metrics to ensure real-time environmental monitoring
Continuous Monitoring	Maturity levels emphasize sustainability integration into security monitoring tools
Risk Management with Quantitative Scoring	RMAF enhances risk assessment by integrating traditional risk management with a quantitative risk-scoring methodology (Appendix E). This enables organizations to apply numerical risk indicators for sustainability alongside security and automation.
Collaboration and Culture	RMAF assesses cultural adoption, ensuring that teams prioritize green computing in DevSecOps

Table 5.5b: The RMAF maturity levels and DevSecOps principles.

5.5.3 Application and Benefits of RMAF

We present a structured assessment and improvement process within the scope of the RMAF below:

- Self-Assessment: An organization conducts internal assessment using the five maturity levels. This enables them to identify gaps and define next steps in sustainability adoption (Dahab et al., 2022).
- Strategic Roadmap Development: Based on the assessment results, the organization should create a roadmap for incremental improvements, aligning sustainability goals with DevSecOps strategies (Bogdanović et al., 2023).
- Integration into Risk Management Frameworks: RMAF findings can be embedded into corporate risk assessments, ensuring that sustainability risks (such as energy inefficiency, high infrastructure emissions) are treated on par with cybersecurity risks (RIMS, 2015).

The main benefits of implementing RMAF include:

- Improved tracking of green computing in DevSecOps environments.
- Better alignment between security, risk, and sustainability objectives.
- Higher DevSecOps maturity, ensuring organizations can automate and scale sustainability efforts.

5.5.4 Comparative Analysis: RMAF vs. Existing Models

While other maturity models (such as OWASP DSOMM, RIMS RMM, and Green IT maturity frameworks) focus on security, enterprise risk, or hardware sustainability, RMAF uniquely combines these perspectives within DevSecOps workflows. Table 5.5c compares RMAF with other maturity models.

Framework	Primary Focus	How RMAF Differs
OWASP DevSecOps Maturity Model (DSOMM)	Security maturity in DevSecOps	RMAF adds sustainability to risk management
RIMS Risk Maturity Model (RMM)	Enterprise risk assessment	RMAF focuses on sustainability within DevSecOps
Green IT Maturity Models	Hardware sustainability	RMAF applies green computing to software development workflows

Table 5.5c: A comparison of RMAF with other maturity models.

In fact, RMAF connects security, risk, and sustainability, making it one of the first maturity models to integrate green computing into DevSecOps.

5.5.5 Challenges in Implementing RMAF

Despite its advantages, RMAF faces several adoption challenges:

- **Cultural Resistance:** Many organizations prioritize speed and security over sustainability, making it difficult to promote eco-friendly DevSecOps policies (Lago, 2019).
- **Lack of Awareness:** It may happen that organizations do not recognize sustainability as a risk factor. It will lead to low adoption rates (Calero, Moraga and García, 2022).
- **Cost of Implementation:** Small and medium-sized enterprises (SMEs) usually find it difficult to invest in sustainability tools or automated tracking systems (Dahab et al., 2022).

Let us now proceed for a comparative analysis of this study's proposed frameworks against existing research, reinforcing their academic and practical contributions.

5.6 Comparison with Prior Work

We now compare the GMF and RMAF with existing green computing frameworks, risk maturity models, and DevSecOps security methodologies. This comparative analysis highlights the novel contributions of our research while acknowledging the foundational concepts drawn from prior work.

5.6.1 Green Metrics Framework vs. Existing Green Computing Models

Existing models have traditionally focused on hardware efficiency, energy optimization in data centers, and environmental impact assessments (Calero, Moraga and Piattini, 2021). Few studies have explicitly and systematically addressed green computing within DevSecOps workflows. Table 5.6a compares GMF with existing sustainability models.

Framework	**Scope**	**Limitations**	**How GMF Differs**
Green IT Balanced Scorecard (Wati and Koo, 2011)	Measures IT sustainability using a scorecard-based approach	Focuses mainly on enterprise-level IT, not software development	GMF integrates sustainability directly into CI/CD pipelines
Sustainable Computing Framework (Philipson, 2011)	Evaluates sustainability in enterprise computing and infrastructure	Lacks DevSecOps-specific indicators	GMF applies green metrics to software delivery and security workflows
G-Readiness Model (Desai and Bhatia, 2011)	Assesses organizational readiness for green IT	Does not address continuous monitoring of sustainability in software engineering	GMF introduces real-time energy and resource tracking in DevSecOps
Software Sustainability Metrics (Bozzelli, Gu and Lago, 2019)	Defines sustainability indicators for software design and development	Limited application to CI/CD workflows	GMF integrates real-time DevSecOps automation and monitoring

Table 5.6a: A comparison of GMF with existing sustainability models.

The Green Metrics Framework (GMF) is different with its focus on DevSecOps, ensuring that ecological sustainability tracking and optimization are automated within CI/CD pipelines rather than being treated as a separate organizational responsibility. This integration provides continuous feedback loops that enable real-time improvements in software sustainability.

5.6.2 Risk-Maturity Assessment Framework (RMAF) vs. Existing Maturity Models

The Risk-Maturity Assessment Framework (RMAF) builds upon established maturity models in security and risk management but uniquely integrates sustainability as a core risk dimension in DevSecOps workflows. Table 5.6b presents a comparative analysis.

Framework	**Focus**	**Limitations**	**How RMAF Differs**
OWASP DevSecOps Maturity Model (DSOMM) (OWASP, 2023)	Evaluates security maturity in DevSecOps	Does not include sustainability metrics	RMAF adds environmental risk as a DevSecOps consideration
RIMS Risk Maturity Model (RMM) (RIMS, 2015)	Enterprise risk assessment framework	Not designed for software development workflows	RMAF applies risk maturity principles to DevSecOps environments
Green IT Maturity Models (Gmach et al., 2012)	Hardware energy efficiency and IT sustainability	Focuses on infrastructure, not software pipelines	RMAF applies risk-based sustainability evaluation to CI/CD

Table 5.6b: A comparison analysis of RMAF with other maturity models.

Unlike existing maturity models that focus on security (OWASP DSOMM) or enterprise-wide risk (RIMS RMM), RMAF introduces a new sustainability dimension

within risk assessment. This integration allows DevSecOps teams to quantify sustainability risks, track progress, and prioritize continuous improvements in eco-efficient software engineering.

5.6.3 Novel Contributions of GMF and RMAF

The combined application of GMF and RMAF introduces several unique contributions to the fields of green computing, risk assessment, and DevSecOps:

- Integration of Sustainability into DevSecOps Pipelines: Unlike prior models, which focus on data centers or general IT sustainability, GMF embeds real-time sustainability tracking within CI/CD workflows. This allows DevSecOps teams to automate green computing practices and reduce software-related environmental impacts.
- Risk-Based Sustainability Assessment for DevSecOps: RMAF is the first framework to introduce ecological sustainability as a risk factor in the maturity evaluations of DevSecOps. It puts green computing in the same level as security governance.
- Automation and Continuous Monitoring: Both frameworks emphasize automated tracking, AI-driven analytics, and feedback loops to optimize sustainability. Prior models rely on static assessments, but GMF and RMAF incorporate automation and continuous monitoring as the core features.
- This comparative analysis confirms that the GMF and RMAF introduce novel advancements. Our frameworks provide an actionable roadmap for embedding green computing into software engineering.

Next, we explore the challenges and opportunities in adopting our two frameworks.

5.7 Challenges and Opportunities

While the GMF and RMAF offer structured approaches to the integration of sustainability into DevSecOps, their implementation presents several challenges and new opportunities. This section discusses key barriers to adoption and potential strategic opportunities for improving sustainability in DevSecOps.

5.7.1 Challenges in Implementing Sustainability in DevSecOps

Despite increasing awareness of sustainability, organizations face multiple barriers when attempting to integrate green computing into software delivery workflows. The following challenges have been identified based on the literature and research findings.

- Cultural Resistance and Organizational Priorities: Many organizations perceive sustainability as a secondary concern, prioritizing security, agility, and cost-efficiency over environmental considerations (Lago, 2019). In fast-paced DevSecOps environments, teams are incentivized to focus on speed and security rather than eco-efficient software engineering. Without executive buy-in and leadership support, sustainability remains a low-priority goal (Harmon and Demirkan, 2011).
- Lack of Standardized Sustainability Metrics: While the GMF introduces DevSecOps-specific sustainability indicators, the broader IT industry lacks standardized benchmarks for measuring software sustainability performance. Existing sustainability models focus primarily on data centers, hardware efficiency, and enterprise IT (Calero, Moraga and García, 2022). Without universal green software metrics, organizations struggle to compare and validate sustainability improvements.

- Integration Complexity in CI/CD Pipelines: DevSecOps pipelines are highly automated and optimized for continuous delivery, making it challenging to integrate real-time sustainability tracking. Existing CI/CD tools lack native features for monitoring carbon footprint, energy efficiency, or resource utilization (Dahab et al., 2022). Implementing sustainability tracking requires custom configurations and additional infrastructure, which can increase operational overhead.
- High Costs and Resource Constraints: SMEs (small and medium-sized enterprises) face financial and technical constraints in adopting sustainability frameworks. AI-powered monitoring tools, cloud-based sustainability dashboards, and real-time energy tracking solutions often require additional investments in infrastructure and workforce training (Bogdanović et al., 2023).
- Lack of Awareness in Risk Management: Many organizations do not recognize energy inefficiency or high resource consumption as risk factors. Traditional risk management models prioritize security, regulatory compliance, and operational risks, with little emphasis on environmental sustainability (RIMS, 2015).

5.7.2 Opportunities for Advancing Sustainability in DevSecOps

Despite the challenges mentioned above, the adoption of green computing in DevSecOps presents opportunities for organizations to align with global sustainability goals and drive long-term innovation.

- Aligning Sustainability with Business and Regulatory Compliance: As regulations on environmental sustainability are considered increasingly important, organizations have an opportunity to leverage this for compliance

and competitive advantage. Government policies, such as the EU Green Deal and the Corporate Sustainability Reporting Directive (CSRD), ask businesses to track and report carbon emissions and environmental impact (European Commission, 2023). By adopting GMF and RMAF, organizations can align with sustainability policies while improving software efficiency.

- Development of Industry Standards for Green DevSecOps: The lack of standardized green DevSecOps metrics presents an opportunity for industry and regulatory bodies to develop universal benchmarks. Collaborations will lead to establishing green computing standards similar to existing cybersecurity compliance frameworks (Calero, Moraga and García, 2022).
- AI and Automation for Sustainability Tracking: AI-driven monitoring and predictive analytics offer new possibilities for real-time sustainability tracking in DevSecOps. Organizations can develop intelligent monitoring systems that automatically assess eco-efficiency during software processes (Haugsvær, 2023).
- Cost Reduction Through Sustainable Software Engineering: Sustainability initiatives in DevSecOps offers an economic opportunity. The energy-efficient software development practices can reduce operational costs by optimizing server utilization, workload allocation, and cloud resource consumption (Gmach et al., 2012).
- Promoting Sustainability Awareness: One of the most effective ways to drive green computing adoption is through training and capacity-building initiatives. Organizations can integrate sustainability education into DevSecOps training programs, and equip teams with tools and best practices for implementing eco-friendly software development (Bozzelli, Gu and Lago, 2019).

To summarize, our two frameworks offer challenges such as cultural resistance, metric standardization issues, and cost constraints, and also opportunities such as leveraging AI-driven monitoring, regulatory compliance incentives, and sustainability education to overcome these barriers. In the next chapter, we shall make concluding remarks and recommendations, outline practical strategies for adopting GMF and propose future research directions in green software engineering.

CHAPTER VI:

CONCLUSION AND RECOMMENDATIONS

6.1 Summary of Findings

Our research focusses on the integration of green computing and DevSecOps without operational efficiency and security. We have developed two structured frameworks by combining qualitative and quantitative research methods:

- The Green Metrics Framework (GMF): We define a set of metrics to measure sustainability in CI/CD pipelines and then to benchmark and continually improve.
- The Risk-Maturity Assessment Framework (RMAF): We formulate a methodology for evaluating an organization's sustainability maturity within DevSecOps environments.
- The research reveals the key challenges and opportunities in embedding sustainability into DevSecOps workflows.

6.1.1 Key Findings

Organizations recognize the importance of sustainability but lack the necessary tools and standardized metrics to implement it effectively (Calero, Moraga and García, 2022). Cultural resistance, perceived trade-offs between sustainability and performance, and financial constraints remain barriers to successful adoption (Lago, 2019).

The GMF provides a structured approach for measuring sustainability in DevSecOps workflows. It embeds real-time tracking tools into CI/CD pipelines and introduces DevSecOps-specific environmental KPIs. The RMAF introduces sustainability as a critical risk dimension and enables organizations to assess their maturity in green computing integration.

While implementing sustainability in DevSecOps, we identify several challenges. The absence of industry-wide benchmarks makes it difficult for organizations to compare and validate sustainability improvements. Existing DevSecOps tools lack native features for real-time sustainability monitoring and require custom configurations. SMEs face financial and technical barriers for adopting sustainability frameworks as it requires additional investments in infrastructure and training.

At the same time, we see opportunities to advance Green DevSecOps. Global regulations, such as the EU Green Deal, present an opportunity to integrate sustainability into compliance-driven software development (European Commission, 2023). AI and automation can enhance real-time tracking of energy consumption, resource utilization, and eco-efficiency. Optimizing server utilization, workload allocation, and cloud resource consumption can reduce operational costs while enhancing sustainability.

6.1.2 Contributions to Research and Practice

We would like to point out that our research has contributed to green computing, DevSecOps, and software engineering by:

- Introducing quantifiable sustainability metrics for software development.
- Aligning sustainability with DevSecOps security and risk frameworks.
- Providing a structured methodology (RMAF) for sustainability maturity assessment.
- Demonstrating the economic and environmental benefits of integrating green computing into DevSecOps workflows.

6.2 Practical Recommendations

Here, we provide practical recommendations for implementing GMF and RMAF. Organizations must implement structured strategies to facilitate the adoption of sustainability in DevSecOps. Based on the findings of this study, the following practical recommendations are proposed for using the GMF and RMAF in software development workflows.

6.2.1 Recommendations for Organizations

Organizations should prioritize sustainability as a core component of their software delivery strategies. They should embed sustainability KPIs, such as the Energy Efficiency Deployment Index (EEDI) and Carbon Impact Recovery Index (CIRI), into DevSecOps monitoring tools. Real-time sustainability tracking may also be implemented using AI-driven analytics and automated dashboards.

Organizations should evaluate their sustainability maturity levels using RMAF and align their sustainability strategies with DevSecOps and other cybersecurity risk frameworks. Sustainability should be integrated into corporate governance policies, ensuring compliance with global environmental sustainability standards.

DevSecOps teams should be trained on best practices in eco-efficient software engineering, including sustainable coding techniques and green computing policies. Awareness campaigns should promote the economic and operational benefits of sustainability; in fact, it will help reduce cultural resistance to green DevSecOps adoption.

Organizations should adopt cloud-based sustainability solutions to optimize resource allocation and usage. AI should be used to automate sustainability decisions, and to ensure energy-efficient deployments without manual intervention. On the side of people management, organizations should also implement green performance indicators in

employee evaluation metrics. It will encourage DevSecOps teams to prioritize sustainability.

6.2.2 Recommendations for Policymakers and Regulators

Regulatory bodies should develop standardized sustainability metrics tailored for software engineering, similar to existing security and compliance frameworks. Industry-wide benchmarks should be created to define sustainability maturity levels accompanied by clear guidelines for green DevSecOps adoption.

Organizations should be required to report carbon emissions, energy efficiency, and resource utilization in software engineering workflows. Policies should align green DevSecOps reporting with global environmental initiatives, such as the EU's Corporate Sustainability Reporting Directive (CSRD).

Governments should introduce tax incentives, grants, and subsidies for organizations implementing sustainability monitoring in DevSecOps. Financial incentives should support AI-driven sustainability solutions to encourage businesses to invest in green technology without economic burden. Policymakers should facilitate collaborations between technology companies, researchers, and regulatory agencies to advance green DevSecOps implementation and enhancements. Sustainability-focused research initiatives should receive government funding, encouraging the development of next-generation eco-friendly DevSecOps tools.

6.2.3 Recommendations for Practitioners and DevSecOps Teams

Software developers, security professionals, and DevSecOps engineers must adopt sustainable software engineering practices to drive eco-friendly digital transformation. Teams should integrate real-time sustainability monitoring tools into their CI/CD

workflows. Existing DevOps security tools, such as the *OWASP DevSecOps Maturity Model (DSOMM)*, should be extended to include sustainability risk assessments.

Developers should use power-efficient algorithms and optimize code execution to minimize energy consumption. Sustainable software engineering should be incorporated into agile development methodologies. They should implement AI-powered predictive analytics to optimize server usage and deployment energy consumption. Automated testing should include sustainability impact assessments.

DevSecOps professionals should advocate a sustainability-first mindset in software process. It involves advocating the inclusion of sustainability in project planning. Sustainability considerations should also be integrated into security risk assessments.

Organizations must use the GMF and RMAF in their software development, security governance, and corporate policies to achieve sustainable DevSecOps. Policymakers should establish standardized regulations for environmental sustainability. Practitioners must adopt eco-friendly development techniques to ensure long-term sustainability.

6.3 Future Research Directions

Advancing the Green Metrics Framework (GMF) and Risk-Maturity Assessment Framework (RMAF) requires further exploration regarding scalability, standardization, automation, and industry adoption. The following key areas highlight promising directions for future research in sustainable software engineering.

6.3.1 Expanding the Scope of Green Metrics in DevSecOps

The GMF proposed in this research focuses on key software sustainability indicators, such as energy efficiency, carbon footprint, and resource utilization. However, there is a need for further refinement to expand these metrics, and we shall discuss these below:

Lifecycle-based Sustainability Metrics: Future research should explore sustainability indicators across the entire software development lifecycle (SDLC), from requirements engineering to maintenance and decommissioning.

Quantifying Social and Economic Sustainability: Current green computing models focus on environmental impact. Additional research is needed to define socioeconomic sustainability factors, such as ethical AI, workforce sustainability, and the long-term impact of sustainable coding practices.

Standardization of Sustainability KPIs: Future studies should focus on establishing industry-wide DevSecOps sustainability benchmarks, similar to existing cybersecurity and software quality standards.

6.3.2 AI and Automation for Sustainability Monitoring

One of the key challenges identified in this study is the lack of automated sustainability tracking within DevSecOps pipelines. Research on AI-driven sustainability monitoring can focus on exploring how machine learning models can predict energy consumption patterns, enabling proactive optimization of software deployment strategies. Research can examine how AI-powered load balancing and resource allocation models can dynamically adjust server workloads to minimize energy consumption. Further research efforts must include how we incorporate sustainability tracking in AIOps (Artificial Intelligence for IT Operations).

6.3.3 Economic Impact of Green DevSecOps Adoption

We have focussed into the environmental aspects of sustainability in our current study. As we have seen in the Chapter 2 – Literature Review, there are other areas of importance including economic. Further research is needed to quantify the economic benefits of green DevSecOps adoption. Future research should conduct empirical studies on the financial return of sustainability initiatives, including energy cost reductions, cloud efficiency savings, and regulatory compliance benefits. New research can explore how organizations can monetize sustainability initiatives using carbon credits and sustainability-driven business incentives. Studies should analyze how cloud providers and software vendors can optimize infrastructure to align with corporate ESG (Environmental, Social, and Governance) goals.

6.3.4 Policy and Regulatory Frameworks for Green DevSecOps

As sustainability finds acceptance in corporate and governmental policies, further research should explore the regulatory landscape for green software engineering. Future research should evaluate how different countries regulate sustainability in software development and identify best practices for harmonizing global. We should focus on creating a standardized framework for integrating sustainability into corporate IT compliance programs, similar to ISO 14001 environmental management. Empirical studies should examine how government incentives, tax breaks, and regulatory mandates influence corporate adoption of green DevSecOps principles.

6.3.5 DevSecOps Culture in Sustainability Adoption

A major barrier identified in this research is cultural resistance to sustainability integration in software engineering teams. Future studies can investigate how developer

mindsets, team collaboration, and organizational culture impact sustainability initiatives. We should assess how gamification strategies and sustainability reward systems influence developer engagement in green software practices. Qualitative research should examine organizations that have successfully embedded sustainability into DevSecOps.

6.4 Closing Remarks

Our research highlights the need for Green DevSecOps while bridging the gap between operational performance, security, and environmental responsibility. We have demonstrated that sustainability does not have to conflict with security and agility; rather, it can be embedded within DevSecOps. The Green Metrics Framework and the Risk-Maturity Assessment Framework offer structured, actionable models for organizations to:

- Measure and optimize sustainability in CI/CD pipelines.
- Integrate environmental risks into security and compliance frameworks.
- Align software engineering with corporate ESG goals.

While our frameworks provide a foundation, widespread adoption will require cultural shifts, regulatory support, and advancements in AI-driven automation. Overcoming standardization gaps, cost constraints, and organizational inertia will boost long-term software development sustainability. Organizations, researchers, and policymakers must collaborate to:

- Develop standardized sustainability benchmarks for DevSecOps.
- Leverage AI, automation, and cloud computing for real-time green computing monitoring.
- Align industry sustainability goals with evolving regulatory frameworks.

By adopting *Green DevSecOps*, organizations would be in a position to ensure that software engineering contributes to technological advancement and a sustainable digital

future. The findings of this study serve as a call to action for businesses, governments, and researchers to work towards a harmonized approach to sustainable software development.

Our study is not the final realization but a step toward broader adoption of sustainability in software engineering. Future advancements will shape a DevSecOps landscape that is secure, efficient, and environmentally responsible, ensuring that technological progress aligns with planetary well-being.

APPENDIX A:

INITIAL INTERVIEW GUIDE

Here's the initial set of questions meant for Chief Technology Officers (CTOs) or Chief Information Officers (CIOs) and is aimed at understanding their perspectives, experiences, and practices related to sustainability in software development:

1. General Information
    a. Can you briefly describe your role and responsibilities within your organization?
    b. How would you describe your organization's commitment to environmental sustainability?
2. Awareness and Importance
    a. How aware are you and your organization of the environmental impact of software development and operations?
    b. In your opinion, how important is it for software development practices to be environmentally sustainable?
3. Current Practices
    a. Does your organization currently incorporate any sustainability practices or metrics in software development? If yes, can you provide some examples?
    b. How is sustainability measured and reported within your software development lifecycle?
4. Integration of Green Metrics into DevSecOps
    a. Are you familiar with the concept of integrating green metrics into DevSecOps practices? If yes, what are your thoughts on it?

b. What challenges do you foresee in integrating environmental sustainability metrics into DevSecOps practices?
c. Has your organization taken any steps toward integrating green metrics into DevSecOps? If so, can you share some of the initiatives or metrics used?

5. Tools and Technologies
    a. What tools or technologies do you currently use that support sustainability in software development?
    b. Are there any tools or technologies you wish were available to better support sustainability efforts in DevSecOps?
6. Impact and Benefits
    a. What impact could integrating green metrics into DevSecOps have on software development practices?
    b. Can you discuss any perceived benefits or drawbacks of incorporating sustainability into DevSecOps?
7. Organizational Challenges and Solutions
    a. What organizational barriers exist to adopting green metrics in DevSecOps, and how might they be overcome?
    b. What support or resources would be most beneficial to your organization in integrating sustainability into DevSecOps?
8. Future Outlook
    a. How do you see the future of sustainability in software development evolving?
    b. What steps should the tech industry take to promote environmental sustainability in software development?
9. Advice and Recommendations

a. What advice would you give to other organizations looking to integrate green metrics into their DevSecOps practices?
b. Are there any best practices or lessons you would like to share from your experience or observation?

We have designed the questions incite thoughtful conversations and provide deep insights into current practices, challenges, and the potential for integrating sustainability into software development processes. The responses will inform our research and contribute to broader knowledge sharing and development of best practices in sustainable software development.

APPENDIX B:

INITIAL SURVEY QUESTIONNAIRE TO UNDERSTAND

INTEGRATING OF GREEN METRICS INTO DEVSECOPS

We have designed a structured questionnaire for collecting data about integrating green metrics with DevSecOps from software companies and individuals with roles like CTOs or CIOs.

A. Respondent Information

1. Role in the Organization:
    i. CEO/President
    ii. CTO
    iii. CIO
    iv. DevSecOps Manager
    v. Other (Please specify): _____
2. Size of Organization:
    i. 1-50 employees
    ii. 51-200 employees
    iii. 201-500 employees
    iv. 501-1000 employees

v. 1001+ employees

3. Industry Sector:
    i. Technology
    ii. Finance
    iii. Healthcare
    iv. Manufacturing
    v. Retail
    vi. Other (Please specify): _____

B. Awareness and Importance

4. How aware are you of the environmental impact of software development?
    i. Very aware
    ii. Somewhat aware
    iii. Not very aware
    iv. Not aware at all
5. On a scale of 1 to 5, how important is it for software development practices to be environmentally sustainable? (1 = Not important at all, 5 = Extremely important)

C. Current Practices

6. Does your organization currently incorporate sustainability practices in software development?
    i. Yes
    ii. No
7. If yes, what percentage of your projects incorporate these practices?
    i. 0-25%
    ii. 26-50%
    iii. 51-75%

iv. 76-100%

D. Integration of Green Metrics into DevSecOps

8. Are you familiar with the concept of integrating green metrics into DevSecOps practices?
    i. Yes
    ii. No
9. Has your organization integrated green metrics into DevSecOps?
    i. Yes
    ii. No
10. If yes, to what extent have green metrics been integrated into your DevSecOps practices?
    i. Fully integrated
    ii. Partially integrated
    iii. Only in the initial stages

E. Challenges and Solutions

11. What challenges have you encountered in integrating green metrics into DevSecOps? (Select all that apply)
    i. Lack of awareness or understanding
    ii. Technical difficulties
    iii. Cost implications
    iv. Lack of suitable tools or technologies
    v. Organizational resistance
    vi. Other (Please specify): _____
12. What resources would be most beneficial in overcoming these challenges? (Select all that apply)

i. Training and education
ii. Better tools and technologies
iii. Financial incentives or support
iv. Leadership buy-in
v. External consultancy or expertise
vi. Other (Please specify): _____

F. Impact and Benefits

13. What impacts have you observed from integrating green metrics into DevSecOps? (Select all that apply)
    i. Reduced energy consumption
    ii. Lower carbon footprint
    iii. Improved efficiency
    iv. Cost savings
    v. None observed
    vi. Other (Please specify): _____
14. On a scale of 1 to 5, how would you rate the overall benefit of integrating green metrics into DevSecOps practices in your organization? (1 = No benefit, 5 = Significant benefit)

This questionnaire facilitates statistical analysis of such integration's adoption, challenges, and impacts.

APPENDIX C:

SUSTAINABILITY AND DEVSECOPS MATURITY SURVEY

We have organized the following survey by category (or domain), aligning each question to the corresponding scoring category defined in Appendix E. Each question will be answered on a 1–5 Likert scale (1 = lowest maturity/not at all, 5 = highest maturity/fully implemented), allowing responses to feed directly into the scoring model.

1. Security
    a. CI/CD Security Integration: Are automated security checks integrated into your continuous integration/continuous delivery pipeline? Consider practices like static code analysis, dependency vulnerability scanning, or dynamic testing on each build/deployment. (1 = No automated security in CI/CD pipeline, 5 = Security scans are fully integrated at every stage of CI/CD, and failures can block a build/release if issues are found.)
    b. Security Metrics Tracking: Does your team collect and monitor DevSecOps security metrics regularly? Examples: number of vulnerabilities found per release, mean time to remediate critical security issues, compliance score from security scans. (1 = No security metrics are tracked or reported, 5 = Comprehensive set of security metrics is tracked, reported to stakeholders, and used to drive continuous security improvements.)
    c. Security Standards Compliance: To what degree does your development process adhere to established security standards or policies (e.g. OWASP Top 10, organization-specific security guidelines) and undergo regular compliance checks? (1 = No formal security standards followed in development, 5 = Strict adherence to security standards with regular audits or reviews to ensure compliance at each release.)
2. Automation
    a. Build and Integration Automation (CI): How fully automated is your build and integration process? (1 = Builds and integrations are run manually with ad-hoc processes, 5 = Fully automated Continuous Integration – every code commit triggers an automated build and integration test suite with no manual intervention.)

b. Testing Automation: To what extent are tests automated and integrated into the pipeline (including unit, integration, and other testing types)? (1 = Little to no test automation – reliance on manual testing, 5 = Extensive automated testing is in place covering most code, and tests run automatically on each build with high coverage and reliability.)

c. Deployment Automation (CD): Are deployments to staging and production environments automated? (1 = Deployments are performed manually with custom scripts or human steps, 5 = Fully automated Continuous Deployment – code changes are deployed to staging/production through an automated pipeline, with no manual steps required and governed by pipeline gates/policies.)

d. Infrastructure as Code and Environment Provisioning: Does the team utilize Infrastructure as Code (IaC) and automated environment provisioning? (1 = Environments and configurations are set up manually for each deployment, 5 = All infrastructure is defined and managed as code with one-click or automated provisioning of environments, ensuring consistency across development, test, and production.)

3. Risk Management

a. Risk Identification Process: Does your team follow a formal process or framework to identify and assess risks (security, operational, compliance, etc.) during the software lifecycle? (1 = No formal risk identification – risks are addressed reactively, if at all, 5 = Formal risk management framework in place – the team proactively identifies and assesses risks in planning stages and continues to update risk assessments throughout the project.)

b. Risk Monitoring and Metrics: To what extent are risk metrics tracked and reviewed regularly? For example, maintaining a risk register with risk severity scores or tracking the number of high-risk items outstanding. (1 = Risks are not documented or quantified, 5 = Clear risk metrics are defined, tracked continuously, and reviewed in routine meetings to inform decision-making.)

c. Mitigation and Response Planning: Does the team have documented risk mitigation plans and incident response procedures, and are these tested or updated on a regular basis? (1 = No documented mitigation or incident response plans exist, 5 = Comprehensive risk mitigation strategies are documented for major identified risks, and incident response plans are in place and regularly tested/refined based on drills or past incidents.)

4. Sustainability

   a. Carbon Footprint Tracking: To what extent does your team measure and track the carbon footprint of its software development, deployment, and operational processes? (1 = No tracking or measurement of carbon footprint, 5 = Comprehensive tracking of carbon emissions with regular monitoring and targets for reduction.)

   b. Energy Efficiency Practices: How thoroughly are energy efficiency considerations integrated into your development and deployment workflows (e.g. using efficient algorithms, optimizing code for lower CPU/memory use, utilizing energy-efficient hardware or cloud services)? (1 = Not at all – energy efficiency is not considered, 5 = Energy efficiency is a key criterion in design, coding, and operations, applied consistently across projects.)

c. Resource Optimization: To what extent does your team optimize computing resources to minimize waste (e.g. using auto-scaling to avoid idle servers, rightsizing cloud instances, shutting down unused resources)? (1 = No resource optimization – significant idle or wasted resources, 5 = Highly optimized resource usage with minimal waste and continuous adjustments to improve efficiency.)

APPENDIX D:

EXISTING METRICS AND MAPPING METRICS TO GOALS

The Existing Matrix

Based on the literature review, we find that the following metrics are used in the domain of DevSecOps and green computing, respectively:

Metric	Description	Calculation	Reference
DevSecOps Metrics			
Deployment Frequency	Measures how often new code is deployed to production.	The number of deployments to production per day/week/month. For example, if there were 20 deployments in a month, the deployment frequency is 20 deployments/month.	Forsgren, Humble and Kim (2018)
Change Lead Time	Time taken from code commit to deployment in production.	Track the timestamp for each code commit and its corresponding deployment. Calculate the difference in time for each pair and take the average. Example: If commit-to-deploy times were 1 hour, 2 hours, and 3 hours, the average lead time is (1+2+3)/3 = 2 hours.	Forsgren, Humble and Kim (2018)
Mean Time to Recover (MTTR)	Measures the average time taken to recover from a failure in production.	Record the downtime for each incident. Sum the total downtime and divide by the number of incidents. Example: If there were 3 incidents with downtimes of 1 hour, 2 hours, and 1 hour, the MTTR is (1+2+1)/3 = 1.33 hours.	Forsgren, Humble and Kim (2018)

Vulnerability Detection Rate	Number of vulnerabilities detected during a specific period.	Count the number of vulnerabilities detected over a period. Divide by the number of code releases or commits in that period. Example: If 10 vulnerabilities were detected in 5 releases, the detection rate is 10/5 = 2 vulnerabilities per release.	Pluralsight Flow Transformation Team (2023)
Vulnerability Patch Time	Measures the time taken to fix detected vulnerabilities.	Record the time taken to fix each detected vulnerability. Calculate the average patch time. Example: If vulnerabilities were patched in 5 hours, 10 hours, and 15 hours, the average patch time is (5+10+15)/3 = 10 hours.	Pluralsight Flow Transformation Team (2023)
Automated Test Coverage	Percentage of the codebase covered by automated tests.	Measure the number of lines of code tested by automated tests. Divide by the total lines of code in the codebase. Example: If 800 lines of code are tested out of 1000 lines, the coverage is 800/1000 = 80%.	Roche (2013)
Green Metrics for DevSecOps			
Energy Consumption	Measures the amount of energy consumed by IT infrastructure.	Monitoring tools are used to measure the total energy consumption of IT infrastructure. Sum the energy usage (in kWh) for servers, storage, and networking equipment over time. Example: If servers consume 1000 kWh and networking equipment 500 kWh, total consumption is 1500 kWh.	Debbarma and Chandrasekaran (2016)
Carbon Footprint	Total greenhouse gas emissions caused by IT	Calculate the total greenhouse gas emissions (CO2 equivalents) from energy	Debbarma and Chandrasekaran (2016)

	operations.	consumption and other sources. Sum the emissions from all IT operations. Example: If data centers emit 1000 kg CO2e and other operations emit 500 kg CO2e, total emissions are 1500 kg CO2e.	
Server Utilization	Measures how efficiently servers are utilized.	Monitor CPU, memory, and storage usage over a period. Calculate the average utilization percentage. Example: If CPU usage is 60%, memory usage is 70%, and storage usage is 50%, the average is (60+70+50)/3 = 60%.	Barroso and Hölzle (2007)
E-Waste Reduction	Tracks the reduction of electronic waste through recycling and proper disposal.	Track the amount of e-waste recycled or properly disposed of. Divide by the total e-waste generated. Example: If 800 kg of e-waste is recycled out of 1000 kg, the reduction rate is 800/1000 = 80%.	Forti et al. (2020)
Renewable Energy Usage	Percentage of IT infrastructure powered by renewable energy sources.	Measure the amount of energy from renewable sources. Divide by the total energy consumption. Example: If 400 kWh of the 1000 kWh total is from renewables, the usage is 400/1000 = 40%.	International Energy Agency (2023)

Mapping Metrics to Goals

Let us now demonstrate how the metrics within the Green Metrics Framework align with specific sustainability goals. We propose to design the framework to measure and manage three key sustainability goals:

- **Energy Efficiency:** Optimize energy consumption across development and operational processes.
- **Carbon Reduction:** Minimize the carbon footprint of deployments and activities in system recovery.
- **Resource Optimization:** Efficient use of infrastructure and reduced electronic waste.

Let us discuss each of these three aspects and demonstrate possible calculations that can be incorporated.

Energy Efficiency

We aim to measure the energy efficiency of frequent deployments. The following are the relevant ones to consider:

- *Deployment Frequency:* Number of deployments to production per unit time (e.g., daily, weekly).
- *Change Lead Time:* Time from code commit to deployment in production.
- *Energy Consumption:* Total energy consumed by IT infrastructure.

Now, we may map to create the following metrics:

1. Energy Efficiency Deployment Index:

$$Energy\ Efficiency\ Deployment\ Index\ = \frac{Deployment\ Frequency}{Energy\ Consumption}$$

For example, if there are 20 deployments per month and the total energy consumption is 1500 kWh, the index is:

$$Index = \frac{20\ deployments/month}{1500\ kWh} = 0.0133\ deployments/kWh$$

2. Average Energy Consumption per Deployment:

$$Average\ Energy\ Consumption\ per\ Deployment = \frac{Energy\ Consumption}{Deployment\ Frequency}$$

For example, for 20 deployments and 1500 kWh:

$$Average = \frac{1500\ kWh}{20\ deployments} = 75\ kWh/deployment$$

3. Energy Efficiency Change Lead Time Index:

$$Energy\ Efficiency\ Change\ Lead\ Time\ Index = \frac{Change\ Lead\ Time}{Energy\ Consumption}$$

For example, if the average lead time is 2 hours and the total energy consumption is 1500 kWh:

$$Index = \frac{2\ hours}{1500\ kWh} = 0.00133\ hours/kWh$$

Carbon Reduction

The goal is to understand the environmental impact of recovery and patch processes. The following are the metrics that may be considered:

- Mean Time to Recover (MTTR): Average time taken to recover from a failure in production.

- Vulnerability Patch Time: Time taken to fix detected vulnerabilities.
- Carbon Footprint: Total greenhouse gas emissions caused by IT operations.

Now, let us map these:

1. Carbon Impact Recovery Index:

$$Carbon\ Impact\ Recovery\ Index\ = \frac{MTTR}{Carbon\ Footprint}$$

It should be noted that CO_2e stands for *carbon dioxide equivalent*, a standard unit to measure carbon footprints. It represents the impact of all greenhouse gases (GHGs) in terms of the amount of CO_2 that would have the same global warming potential (GWP). CO_2e allows us to express the total impact of various GHGs, including methane (CH_4), nitrous oxide (N_2O), and others, in a single metric. Each gas has a different GWP, meaning some gases are much more potent in trapping heat in the atmosphere than CO_2. For instance, methane has a GWP of about 28-36 times that of CO_2 over 100 years. To summarize, 1500 kg CO_2e means that the combined impact of all the greenhouse gases emitted is equivalent to the impact of emitting 1500 kg of CO_2.

In our example, if MTTR is 1.33 hours and the carbon footprint is 1500 kg CO2e:

$$Index\ = \frac{1.33\ hours}{1500\ kg\ CO2e}\ =\ 0.00089\ hours/kg\ CO2e$$

2. Carbon Impact Patch Index:

$$Carbon\ Impact\ Patch\ Index\ = \frac{Vulnerability\ Patch\ Time}{Carbon\ Footprint}$$

In our example, if the average patch time is 10 hours and the carbon footprint is 1500 kg CO2e:

$$Index\ = \frac{10\ hours}{1500\ kg\ CO2e}\ =\ 0.00667\ hours/kg\ CO2e$$

Resource Optimization

The goal here is to ensure the efficient use of resources and minimize environmental impact. The following are the metrics that may be considered:

- Server Utilization: Average utilization percentage of CPU, memory, and storage.
- Automated Test Coverage: Percentage of the codebase covered by automated tests.
- E-Waste Reduction: The amount of e-waste recycled or properly disposed of is divided by the total e-waste generated.

Now, let us map these:

1. Server Utilization Efficiency Index:

$$Server\ Utilization\ Efficiency\ Index\ = \frac{Server\ Utilization}{Energy\ Consumption}$$

In our example, if server utilization is 60% and energy consumption is 1500 kWh:

$$Index = \frac{60\%}{1500\ kWh} = 0.04\%/kWh$$

2. Test Coverage Resource Optimization Index:

$$Test\ Coverage\ Resource\ Optimization\ Index = \frac{Automated\ Test\ Coverage}{Server\ Utilization}$$

In our example, if automated test coverage is 80% and server utilization is 60%:

$$Index = \frac{80\%}{60\%} = 1.33$$

3. E-Waste Efficiency Index:

$$E - Waste\ Efficiency\ Index = \frac{E - Waste\ Reduction}{Server\ Utilization}$$

In our example, if e-waste reduction is 80% and server utilization is 60%:

$$Index = \frac{80\%}{60\%} = 1.33$$

In perspective, the integrated framework combines DevSecOps metrics with sustainability metrics to measure alignment with sustainability goals. We calculate the indices for energy efficiency, carbon reduction, and resource optimization. Thus, organizations can quantitatively assess performance and identify areas for improvement.

APPENDIX E:
SCORING MODEL FOR SUSTAINABILITY MATURITY AND RISK IN DEVSECOPS

We have proposed this scoring model to enhance the DevSecOps framework and explicitly integrate sustainability maturity with traditional DevSecOps risk indicators. It means evaluating how well development practices reduce environmental impact in parallel with managing security, automation, and risk. The scoring model outlined here introduces sustainability metrics (e.g., carbon footprint reduction, energy efficiency, resource optimization) into the maturity assessment, with weighted scoring that balances eco-conscious practices with security and automation goals. A Likert-scale (1–5) rating is used for each area to quantify maturity levels, and statistical measures (weighted averages and variance) help interpret overall performance. The result is a structured model with a composite score reflecting DevSecOps excellence and environmental responsibility. It defines how to quantify responses from Appendix C's questionnaire into actionable metrics.

Integrating Sustainability with DevSecOps Risk Indicators

Traditional DevSecOps scoring models consider code security, CI/CD automation, compliance, incident response, and other risk indicators. This revised model augments those areas with a sustainability dimension, treating it as a first-class risk and quality management concern. When assessing DevSecOps maturity, we shall now evaluate teams on sustainability practices alongside security and operational metrics.

- Holistic Categories: The scoring is divided into key categories, such as security, automation, risk management, and sustainability. Sustainability is introduced as a new category of evaluation rather than an afterthought. Each category has

specific criteria: Security might include vulnerability management and compliance, Automation might cover continuous integration/deployment and testing, Risk Management might consist of incident response and governance, and Sustainability covers the metrics above (carbon, energy, resource use).

- Parallel Risk Indicators: Sustainability risks (like excessive energy use or non-compliance with environmental standards) are considered parallel to other DevSecOps risks (such as security vulnerabilities or operational downtime). This integration acknowledges that poor sustainability practices can pose long-term risks to the organization (e.g., regulatory, reputational, or cost risks) just as security flaws or lack of automation can. For instance, a high carbon footprint or energy-inefficient system might risk non-compliance with emerging green regulations or incur higher operational costs. Thus, it's included in the risk scoring.
- Pipeline Integration: In practical terms, teams embed checks and measures for sustainability in their DevSecOps pipeline. It could mean adding automated sustainability audits (similar to security audits) that flag energy-inefficient code or architectures and incorporating tools to measure carbon emissions per build or deployment. GreenOps practices – such as CI/CD telemetry monitoring power usage and carbon output – ensure sustainability is continuously managed, just like security tests are. By integrating these into the pipeline, organizations treat sustainability metrics as equally visible and actionable as build quality or security scan results. (For example, a build pipeline might fail a release if it exceeds a certain energy consumption threshold, analogous to failing on too many security vulnerabilities.)

This integrated approach ensures the scoring model reflects a balanced DevSecOps maturity: a mature team not only excels in shipping secure, reliable software fast but does so in an energy-efficient, environmentally conscious way.

Weighted Scoring Model Balance

The model uses weighted scoring across the categories to incorporate sustainability without overshadowing other critical DevSecOps aspects. Each category (Security, Automation, Risk Management, Sustainability, etc.) is assigned a weight reflecting its importance to the overall DevSecOps maturity score. Sustainability is given a meaningful weight (e.g., 20–25% of the total score) to influence the overall score proportionately with long-established domains like security or automation. This weighted approach prevents any single area from dominating the score and encourages a balanced improvement across all areas. Table E.1 provides an example of weight distribution.

Domain	**Weight**
Security	30%
Automation	25%
Risk Management	20%
Sustainability	25%

Table E.1: Example Weight Distribution

In this scenario, sustainability is kept on par with the other major categories (each roughly a quarter of the focus). Organizations can adjust weights based on strategic priorities, but sustainability should carry a significant weight to ensure it is not neglected. The goal is to balance eco-conscious practices with security and reliability rather than trade one for the other.

The rationale behind it is that weighting underscores that sustainability is a core component of DevSecOps excellence, not just a nice-to-have. By quantifying it, leaders can make informed decisions: for instance, even if a team has strong security and automation, a low sustainability score (with its assigned weight) will pull down the overall score, highlighting a gap. Conversely, teams that are strong in sustainability but weak in security would see that reflected. The weighted model states that true maturity means performing well across all dimensions.

The next thing that we consider is **Composite Scoring**. With weights assigned, each category's raw score (on the Likert 1–5 scale) contributes to a composite **DevSecOps Sustainability Maturity Score**. This composite is essentially a weighted average (details on calculation in a later section). For example, if Security scored 4/5, Automation 3/5, Risk 4/5, and Sustainability 2/5, the overall score would be calculated from those values with their respective weights. It provides a single concise benchmarking metric while allowing drill-down into category-specific scores.

By introducing weighted scoring, the model balances sustainability considerations with security, automation, and risk management priorities. It encourages teams to improve sustainability without sacrificing other areas (and vice versa), promoting a well-rounded DevSecOps practice.

Likert Scale Maturity Levels (1–5) for Sustainability

Each aspect of the scoring model, including sustainability, is assessed on a Likert scale from 1 to 5, indicating maturity levels from low (immature practices) to high (optimized practices). The scale is defined as follows for sustainability (similar scales apply to other categories):

- Level 1 – Ad Hoc / Absent: Sustainability is not considered in the development process. No specific practices or metrics are in place for carbon reduction or energy efficiency. The team is likely unaware of the software's environmental impact. (Example: no resource usage monitoring beyond basic cost; servers run 24/7 regardless of need).
- Level 2 - Initial / Reactive: Some sustainability awareness exists, often driven by external prompts or individual efforts. Practices are reactive and not standardized. Isolated attempts to reduce waste might exist (e.g., occasional cloud cost optimizations that incidentally reduce energy use). No formal metrics are tracked yet, or tracking is infrequent.
- Level 3 - Defined / Managed: Sustainability practices are defined and repeatable. The team has set measurable goals for carbon footprint reduction and energy efficiency, and basic metrics are collected (e.g., tracking energy consumption of builds using more efficient instances). Tools or processes (like power usage monitoring in CI pipelines) are in place, though improvements may be manual or early. Sustainability is part of risk assessments and is considered in planning (for instance, choosing a data center region with lower-carbon electricity for deployments).
- Level 4 – Proactive / Integrated: The team proactively optimizes for sustainability. Clear metrics (carbon emissions, energy per transaction, etc.) are tracked for each release, and the team regularly analyzes and acts on this data. Automation supports sustainability by integrating automated shutdown of idle resources, continuous carbon monitoring, green design patterns, etc. Sustainability has dedicated ownership (by appointing a champion or team) and

is part of the organization's performance reviews or objectives. The practices are primarily standardized and consistently executed.

- Level 5 – Optimizing / Innovative: Sustainability is an ingrained, strategic priority, and the team continuously innovates to improve it. The software delivery process includes advanced optimizations like running tasks when and where renewable energy is available (known as carbon-aware scheduling) and extensive use of renewable-powered infrastructure. The organization meets internal sustainability targets and contributes to industry best practices (publishing metrics, open-sourcing green tools, etc.). At this level, sustainability efforts are fully balanced with other DevSecOps goals. Sustainability metrics show year-over-year improvement and efficient resource usage (e.g., minimal carbon per user or build), reflecting leadership in sustainable DevSecOps.

Each level on this Likert scale provides a quantitative score (1 through 5) that feeds into the scoring model. Using a Likert scale enables straightforward quantification of qualitative maturity traits. Teams can self-assess or be audited against descriptions for each level to determine their score. This approach brings clarity and consistency to measuring sustainability maturity, akin to how one might rate security or process maturity on a numeric scale.

Statistical Interpretation: Weighted Averages and Variance

The model produces an overall maturity score after scoring each category (including sustainability) on the 1–5 scale and applying weights. It is calculated as a weighted average of all category scores - Weighted Average (Composite Score). Multiply each category's score by its weight (as a percentage or coefficient), sum these up, and

divide by the sum of weights (usually 100% if weights are percentages). For example, if Security=4 (weight 0.30), Automation=3 (0.25), Risk=4 (0.20), and Sustainability=2 (0.25), the overall score = (40.30 + 30.25 + 40.20 + 20.25) = 3.25 out of 5 (or 65 out of 100). This weighted scoring reflects performance balanced across all areas – a low sustainability score will pull down the average, even if other areas are strong, and vice versa. Over time, improvements in any category will raise the composite score, allowing teams to track holistic progress.

Besides the average, the model looks at the variance (or spread) of scores across categories – mainly focusing on sustainability vs. other areas. A high variance indicates the team excels in some areas but lags in others. For instance, if sustainability scored much lower than all other categories, the variance would be high, highlighting an imbalance in DevSecOps maturity. On the other hand, a low variance (scores are more uniform across categories) suggests a well-rounded approach. Calculating the variance (or standard deviation) of category scores helps identify whether sustainability is an outlier – far ahead or far behind other indicators. This is important for interpretation: even if the overall average is moderate, a high variance might signal uneven maturity that warrants attention (e.g., strong security and automation, but poor sustainability practices could be a risk regarding long-term viability and compliance).

Organizations can interpret their results more effectively using these statistics. The weighted average gives a single DevSecOps Sustainability Maturity score to benchmark against targets or industry standards. The variance provides insight into whether the team should focus on leveling up a particular area. For example, an organization might find an overall score of 3.5/5 with a significant variance – upon review, they see sustainability as only 2/5 while all other areas are 4/5. It indicates a clear need to invest in sustainability (tooling, training, process improvements) to balance the DevSecOps program. Conversely,

if sustainability is high but perhaps automation is low, resources might shift to improving CI/CD without losing the sustainability gains.

By examining both the aggregated score and the score distribution, the model rates performance and guides risk management and continuous improvement efforts. A balanced, low-variance scorecard implies the organization is uniformly mature (an ideal scenario), whereas any significant gaps pinpoint where additional effort is needed to achieve overall DevSecOps excellence.

Structured Scoring Model Overview

Bringing it all together, below is a structured view of the DevSecOps scoring model with sustainability integrated. Each category is assessed on a 1–5 scale and weighted; the example weights and criteria can be adjusted to fit an organization's needs, as in Table E.2 below.

Category	**Key Focus**	**Risk Indicator**	**Weight**	**Maturity Score (1–5)**	**Weighted Score**
Security	Vulnerability management, secure coding, compliance adherence.	Number of Open Vulnerabilities, Time to Remediate	30%	4	1.20
Automation	CI/CD pipeline integration, automated testing, infrastructure as code.	Deployment Frequency, Lead Time for Changes	25%	3	0.75
Risk Mgmt	Threat modeling, incident response, governance processes.	Incident Frequency, Meantime to Recovery	20%	4	0.80
Sustainability	Carbon reduction initiatives, energy-	Carbon Emissions per	25%	2	0.50

	efficient architecture, resource optimization practices.	Release, Energy Use per Transaction, % of Resources Optimized			
Overall - Weighted average of the above.	**Composite DevSecOps and Sustainability Score**	**-**	**100%**	**-**	**3.25 (out of 5)**

Table E.2: Structured Scoring Model with Sustainability.

Each category is rated 1–5 (Likert-scale) on maturity, multiplied by weight to contribute to the overall score. In the example above, the low Sustainability score (2) significantly lowers the overall composite to 3.25 despite high Security and Risk scores, indicating a gap. The model encourages a balanced improvement: teams should aim to raise their sustainability maturity (e.g., from $2 \rightarrow 4$) while maintaining or improving other areas, which would substantially increase the overall score (in the example, if Sustainability became 4, the overall would jump to 4.0).

In summary, this scoring model embeds sustainability into DevSecOps evaluation in a formal, quantifiable way. By integrating sustainability metrics (carbon footprint, energy efficiency, resource usage) with existing risk indicators and weighted Likert-scale scoring, organizations get a comprehensive view of their DevSecOps maturity. This model assesses how well teams deliver secure and reliable software and how responsibly they do so with respect to the planet.

REFERENCES

Ahmaro, I.Y.Y., Bin Mohd Yusoff, M.Z. and Abualkishik, A.M. (2014) 'The current practices of green computing approaches in Malaysia', in *Proceedings of the 6th International Conference on Information Technology and Multimedia.* ieeexplore.ieee.org, pp. 341–345.

Ahmed, A.M.A.A. (2019) 'DevSecOps: Enabling Security by Design in Rapid Software Development', in *International Journal of Advanced Computer Science.*

Agarwal, S. et al. (2015) 'Green solutions: A pilot study on green technology and green computing', in *International Journal*, 5(10), pp. 680–686.

Albertao, F. et al. (2010) 'Measuring the sustainability performance of software projects', in *2010 IEEE 7th International Conference on E-Business Engineering. 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE), IEEE,* pp. 369–373.

Alharthi, A.D., Spichkova, M. and Hamilton, M. (2018) 'SuSoftPro: Sustainability Profiling for Software', in *2018 IEEE 26th International Requirements Engineering Conference (RE),* pp. 500–501.

Andrikopoulos, V. et al. (2022) 'Sustainability in Software Architecture: A Systematic Mapping Study', in *48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 426–433.

Anwar, H. and Pfahl, D. (2017) 'Towards Greener Software Engineering Using Software Analytics: A Systematic Mapping', in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE*, pp. 157–166.

Arseneault, E. et al. (2022) 'Experience-Based Guidelines for Effective Planning & Management of Software Integration & Test Activities in the Agile/DevSecOps

Environment', in *2022 IEEE 29th Annual Software Technology Conference (STC)*, pp. 155–155.

Azad, N. and Hyrynsalmi, S. (2023) 'DevOps critical success factors - A systematic literature review', in *Information and Software Technology*, 157, p. 107150.

Bambazek, P., Groher, I. and Seyff, N. (2022) 'Sustainability in Agile Software Development: A Survey Study among Practitioners', in *2022 International Conference on ICT for Sustainability (ICT4S)*, pp. 13–23.

Bang, S.K. et al. (2013) 'A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps', in *Proceedings of the 2nd annual conference on Research in information technology. New York, NY, USA: Association for Computing Machinery (RIIT '13)*, pp. 61–62.

Barroso, L.A. and Hölzle, U. (2007) 'The case for energy-proportional computing', in *IEEE Computer*, 40(12), pp. 33–37. DOI: 10.1109/MC.2007.443.

Bash, C. et al. (2023) 'Sustainability: Fundamentals-Based Approach to Paying It Forward', in *Computer*, 56(1), pp. 125–132.

Bermon Angarita, L., Fernández Del Carpio, A. and Osorio Londoño, A.A. (2022) 'A bibliometric analysis of DevOps metrics', in *DESIDOC Journal of Library & Information Technology*, 42(6), pp. 387-396. doi:10.14429/djlit.42.6.18365.

Bozzelli, P., Gu, Q., and Lago, P. (2019) 'A systematic literature review on green software metrics', *VU University Amsterdam*, Available at: https://core.ac.uk/display/43408497 (Accessed: 6 January 2025).

Bogdanović, Z. et al. (2023) "The Role of DevOps in Sustainable Enterprise Development", in *Sustainability: Cases and Studies in Using Operations Research and Management Science Methods*, Cham: Springer International Publishing, pp. 217–237.

Calero, C., Moraga, M.Á. and García, F. (2022) ‘Software, Sustainability, and UN Sustainable Development Goals’, in *IT Professional*, 24(1), pp. 41-48. doi: 10.1109/MITP.2021.3117344.

Calero, C., Moraga, M.Á. and Piattini, M. (2021) ‘Introduction to Software Sustainability’, in *Software Sustainability*. Cham: Springer International Publishing, pp. 1–15.

Catolino, G. (2020) ‘A blessing in disguise? Assessing the Relationship between Code Smells and Sustainability’, in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 779–780.

Chang, V., Wills, G. and De Roure, D. (2010) ‘A Review of Cloud Business Models and Sustainability’, in *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 43–50.

Chitchyan, R. et al. (2016) ‘Sustainability design in requirements engineering: state of practice’, in *Proceedings of the 38th International Conference on Software Engineering Companion*. New York: Association for Computing Machinery (ICSE ’16), pp. 533–542.

Chitchyan, R., Noppen, J. and Groher, I. (2015) ‘What Can Software Engineering Do for Sustainability: Case of Software Product Lines’, in *2015 IEEE/ACM 5th International Workshop on Product Line Approaches in Software Engineering*, pp. 11–14.

Condori-Fernandez, N. and Lago, P. (2018) ‘Characterizing the contribution of quality requirements to software sustainability’, in *The Journal of systems and software*, 137, pp. 289–305.

Creswell, J.W., and Clark, V.L.P. (2017) *Designing and Conducting Mixed Methods Research*. SAGE Publications.

Dahab, S. et al. (2022) ‘A learning-based approach for green software measurements’, in *Proceedings of ITEA3 Project Measure Conference*.

Debbarma, T. and Chandrasekaran, K. (2016) ‘Green Measurement Metrics Towards a Sustainable Software: A Systematic Literature Review’, in *IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2016)*, Jaipur, India, pp. 1–7.

Desai, A. and Bhatia, M. (2011) ‘The G-Readiness Model: Assessing organizational readiness for green IT initiatives’, in *Journal of Sustainable IT Practices*, 4(2), pp. 98–112.

Erdélyi, K. (2013) ‘Special factors of development of green software supporting eco sustainability’, in *2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY). IEEE*, pp. 337–340.

European Commission (2023) European Green Deal and Corporate Sustainability Reporting Directive (CSRD). *Brussels: European Commission.* Available at https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal_en (European Green Deal) and https://finance.ec.europa.eu/capital-markets-union-and-financial-markets/company-reporting-and-auditing/company-reporting/corporate-sustainability-reporting_en (CSRD) (Accessed: 9 February 2025).

Fakhar, F. et al. (2012) ‘Software level green computing for large scale systems’, in *Journal of Cloud Computing: Advances, Systems and Applications*, 1(1), p. 4.

Ferri, J., de Barros, R.M. and Brancher, J.D. (2011) ‘Proposal for a Framework Focus on Sustainability’, in *30th International Conference of the Chilean Computer Science Society*, pp. 127–134.

Forsgren, N., Humble, J. and Kim, G. (2018) *Accelerate: The Science of Lean Software and DevOps*. Portland: IT Revolution Press.

Forti, V. et al. (2020) *The Global E-waste Monitor 2020*. Bonn/Geneva: United Nations University/ITU.

Freitag, C. et al. (2021) 'The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations', in *Patterns* (New York, N.Y.), Vol. 2 No. 9, p. 100340.

Freundlieb, M. and Teuteberg, F. (2012) 'Evaluating the Quality of Web Based Sustainability Reports: A Multi-method Framework', in *2012 45th Hawaii International Conference on System Sciences*, pp. 1177–1186.

Garousi, V., Felderer, M. and Mäntylä, M.V. (2019) 'Guidelines for including grey literature and conducting multivocal literature reviews in software engineering', in *Information and Software Technology*, 106, pp. 101–121.

Garscha, P. (2021) 'From Sustainability in Requirements Engineering to a Sustainability-Aware Scrum Framework', in *2021 IEEE 29th International Requirements Engineering Conference (RE)*, pp. 462–467.

Gerostathopoulos, I., Raibulet, C. and Lago, P. (2022) 'Expressing the adaptation intent as a sustainability goal', in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*. New York: Association for Computing Machinery (ICSE-NIER '22), pp. 36–40.

Gmach, D. et al. (2012) 'Profiling Sustainability of Data Centers', in *Proceedings of the IEEE International Conference on Sustainable Computing and Communications*, pp.1-6. DOI: http://dx.doi.org/10.1109/SusCom.2012 (Accessed: 6 January 2025).

Groher, I. and Weinreich, R. (2017) ‘An Interview Study on Sustainability Concerns in Software Development Projects’, in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 350–358.

Gupta, A. (2022) ‘An Integrated Framework for DevSecOps Adoption’, in *International journal of computer trends and technology*, 70(6), pp. 19–23.

Guzman Camacho, N. (2024) ‘Unlocking the Potential of AI/ML in DevSecOps: Effective Strategies and Optimal Practices’, in *Journal of Artificial Intelligence General Science (JAIGS)*, 3(1), pp.1-12.

Harmon, R.R. and Auseklis, N. (2009) ‘Sustainable IT services: Assessing the impact of green computing practices’, in *PICMET ’09 - 2009 Portland International Conference on Management of Engineering & Technology*. ieeexplore.ieee.org, pp. 1707–1717.

Harmon, R.R. and Demirkan, H. (2011) ‘The Corporate Sustainability Dimensions of Service-Oriented Information Technology’, in *2011 Annual SRII Global Conference*, pp. 601–614.

Haron, H. et al. (2015) ‘Software Reusability in Green Computing’, in *Advanced science letters*, 21(10), pp. 3283–3287.

Haugsvær, S.B. (2023) ‘Sustainable BizDevOps: A novel methodology for reducing the carbon footprint of web products’, in *NTNU*. Available at: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3079159 (Accessed: 26 December 2023).

Ibrahim, S.R.A., Sallehudin, H. and Yahaya, J. (2023) ‘Exploring Software Development Waste and Lean Approach in Green Perspective’, in *2023 International Conference on Electrical Engineering and Informatics (ICEEI)* (pp. 1-6).

International Energy Agency (2023) *Electricity Market Report 2024 – Executive Summary*, IEA.

Jayalath, J.M.T.I. et al. (2019) 'Green Cloud Computing: A Review on Adoption of Green-Computing attributes and Vendor Specific Implementations', in *2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*. ieeexplore.ieee.org, pp. 158–164.

Jimenez, E., Calero, C., and Moraga, M.Á. (2022) CSRE4SOC (CSR evaluation for software companies). arXiv preprint arXiv:2209.13372.

Jindal, G. and Gupta, M. (2012) 'Green computing "future of computers"', in *International Journal of Emerging Research in Management & Technology*, 1(2), pp. 14–18.

Kienzle, J., Strooper, P., and Viller, S. (2016) 'Towards a sustainable software development lifecycle', in *Computer Science - Research and Development*, 31(1), pp. 5-15.

Kipp, A. and Jiang, T. (2011) 'Green Metrics for Energy-aware IT Systems', in *Proceedings of the 2011 International Conference on Complex, Intelligent, and Software Intensive Systems*, pp. 381-386.

Koskinen, A. (2019) 'DevSecOps: Building Security into the Core of DevOps', in *Journal of DevOps Practices*.

Lago, P. (2019) 'Architecture Design Decision Maps for Software Sustainability', in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, pp. 61–64.

Lami, G. and Buglione, L. (2012) 'Measuring Software Sustainability from a Process-Centric Perspective', in *Proceedings of the Joint Conference of the 22nd*

International Workshop on Software Measurement and the 7th International Conference on Software Process and Product Measurement. IEEE.

Landauer, C. (2006) 'Wrapping Architectures for Long-Term Sustainability', in *2006 Second International IEEE Workshop on Software Evolvability (SE'06)*, pp. 44–49.

Lange, S., Pohl, J. and Santarius, T. (2020) 'Digitalization and energy consumption. Does ICT reduce energy demand?', in *Ecological economics: the journal of the International Society for Ecological Economics*, 176, p. 106760.

Le, D.M. et al. (2016) 'Relating Architectural Decay and Sustainability of Software Systems', *in 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 178–181.

Li, Q. and Zhou, M. (2011) 'The survey and future evolution of green computing', in *Proceedings of the IEEE International Conference on Green Computing and Communications. IEEE.*

Mack, S.D. (2023) *The DevSecOps Playbook: Deliver Continuous Security at Speed*. New York: John Wiley & Sons.

Mallouli, W. et al. (2020) 'Metrics-driven DevSecOps', in *Proceedings of the 15th International Conference on Software Technologies*. 15th International Conference on Software Technologies, SCITEPRESS - Science and Technology Publications. Available at: https://doi.org/10.5220/0009889602280233.

Mann, H., and Maurer, F. (2015) 'A case study on the impact of scrum on overtime and customer satisfaction in agile software development projects', in *Journal of Systems and Software*, 117, pp. 170-183.

Mourão, B.C., Karita, L., and Machado, I.C. (2018) 'Green and Sustainable Software Engineering: A Systematic Mapping Study', in *Proceedings of the 7th Brazilian*

Symposium on Software Components, Architectures, and Reuse, pp. 121–130. doi:10.1145/3275245.3275258.

Muthu, M., Banuroopa, K. and Arunadevi, S. (2019) 'Green and Sustainability in Software Development Lifecycle Process', in *Sustainability Assessment at the 21st Century*, 27(63). Available at: https://books.google.com/books?hl=en&lr=&id=yJj8DwAAQBAJ&oi=fnd&pg=PA63&dq=software+development+green+computing&ots=4uFAQbWCZe&sig=YBEEuojkyLAuUSZS9NW5REX5DI0.

Ono, T., Iida, K. and Yamazaki, S. (2017) 'Achieving sustainable development goals (SDGs) through ICT services', in *Fujitsu Sci. Tech. J*, 53(6), pp. 17–22.

Orozco-Garcés, C.-E., Pardo-Calvache, C.-J., and Suescún-Monsalve, E. (2022) 'Metrics Model to Complement the Evaluation of DevOps in Software Companies', in *Revista Facultad de Ingeniería*, 31(62), e14766. doi:10.19053/01211129.v31.n62.2022.14766.

OWASP (2023) 'OWASP DevSecOps Maturity Model (DSOMM)', *Open Web Application Security Project (OWASP)*. Available at: https://owasp.org/www-project-devsecops-maturity-model/ (Accessed: 9 February 2025).

Pakalapati, N., Venkatasubbu, S. and Sistla, S.M.K. (2023) 'The Convergence of AI/ML and DevSecOps: Revolutionizing Software Development', in *Journal of Knowledge Learning and Science Technology*, 2(2), pp.190-212. DOI: https://doi.org/10.60087/jklst.vol2.n2.p212 (Accessed 6 January 2025).

Penzenstadler, B. et al. (2021) 'Iterative Sustainability Impact Assessment: When to propose?', in *2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS)*, pp. 5–6.

Petrović, N. (2023) ‘Machine Learning-Based Run-Time DevSecOps: ChatGPT Against Traditional Approach’, preprint, pp. 1–5.

Philipson, G. (2011) ‘A Framework for Green Computing’, *International Journal of Green Computing (IJGC)*, 2(1), pp. 12–26.

Pluralsight Flow Transformation Team (2023) ‘26 DevOps KPIs and Metrics: Guide to DORA Progress’, in *Pluralsight Blog*. Available at: https://www.pluralsight.com/resources/blog/business-and-leadership/devops-metrics (Accessed: 16 January 2025).

Prates, L. et al. (2019) ‘DevSecOps Metrics’, in *Information Systems: Research, Development, Applications, Education. Springer International Publishing*, pp. 77–90.

Rahul, B.S., Kharvi, P. and Manu, M.N. (2019) 'Implementation of DevSecOps using open-source tools', in *International Journal of Advance Research, Ideas and Innovations in Technology*, 5(3), pp. 1050-1051.

Rajapakse, R.N. et al. (2022) ‘Challenges and solutions when adopting DevSecOps: A systematic review’, in *Information and Software Technology, Elsevier*, Vol. 141, p. 106700.

Riekstin, A.C. et al. (2016) ‘Monitoring and Measurement System for Green Operation of Geographically Distributed ICT Services’, in *Proceedings of the International Conference on Energy-Efficient Computing and Networking*. ACM.

Rath, A.K. (2013) *Cloud Computing: Facing the Reality*. Kindle edition. Available at: https://www.amazon.in/Cloud-Computing-Reality-Ashwini-Rath-ebook/dp/B00EYNGVH0 (Accessed: 16 January 2025).

Rath, A. et al. (2012) ‘Decision points for adoption of cloud computing in SMEs’, in *2012 International Conference for Internet Technology and Secured*

Transactions. London, 10–12 December. IEEE. ISBN: [Provide the ISBN number if available]. Available at: https://ieeexplore.ieee.org/document/6470904 (Accessed: 16 January 2025).

Rath, A. (2024) *Concepts and Practices of DevSecOps: Crack the DevSecOps interviews*. New Delhi: BPB Publishers.

Raja, S.P. (2021) 'Green Computing and Carbon Footprint Management in the IT Sectors', in *IEEE Transactions on Computational Social Systems*, 8(5), pp. 1172–1177.

RIMS (2015) 'RIMS Risk Maturity Model (RMM)', *Risk and Insurance Management Society (RIMS)*. Available at: https://www.rims.org/Tools/risk-maturity-model (Accessed: 9 February 2025).

Roche, J. (2013) 'Adopting DevOps practices in quality assurance', *Communications of the ACM*, 56(11), pp. 38–43.

Ruokolainen, T. and Kutvonen, L. (2012) 'An Architecture Framework for Facilitating Sustainability in Open Service Ecosystems', in *2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops*, pp. 84–93.

Sagar, S. and Pradhan, N. (2021) 'A Review: Recent Trends in Green Computing', in *B. Balusamy, N. Chilamkurti, and S. Kadry (eds) Green Computing in Smart Cities: Simulation and Techniques*. Cham: Springer International Publishing, pp. 19–34.

Saha, B. (2018) 'Green computing: current research trends', in *International Journal of Computer Sciences and Engineering*, 6(3), pp. 467–469.

Sallou, J., Cruz, L. and Durieux, T. (2023) 'EnergiBridge: Empowering Software Sustainability through Cross-Platform Energy Measurement'. arXiv preprint arXiv:2312.13897.

Schopf, J.M. (2009) ‘Sustainability and the Office of CyberInfrastructure’, in *2009 Eighth IEEE International Symposium on Network Computing and Applications*, pp. 1–3.

Singh, S. (2015) ‘Green computing strategies & challenges’, in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. ieeexplore.ieee.org, pp. 758–760.

Spencer, T. and Singh, S. (2024) ‘What the data centre and AI boom could mean for the energy sector’, in *International Energy Agency*. Available at: https://www.iea.org/commentaries/what-the-data-centre-and-ai-boom-could-mean-for-the-energy-sector [Accessed 2 January 2025].

Sriraman, G. and Raghunathan, S. (2023) 'A Systems Thinking Approach to Improve Sustainability in Software Engineering - A Grounded Capability Maturity Framework', in *Sustainability*, 15(11), p. 8766. Available at: https://www.mdpi.com/2071-1050/15/11/8766 (Accessed: 6 January 2025).

Chen, T. and Suo, H. (2022) ‘Design and Practice of Security Architecture via DevSecOps Technology’, in *2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 310-313). IEEE.

Tee, M., Abdullah, R. and Abdullah, S. (2015) ‘A systematic literature review of green software development in collaborative knowledge management environment’, in *International Journal of Advanced Computer Technology*, 4(1), pp. 63-72.

Tjoa, A.M. and Tjoa, S. (2016) ‘The Role of ICT to Achieve the UN Sustainable Development Goals (SDG): 6th IFIP World Information Technology Forum, WITFOR 2016, San José, Costa Rica, September 12-14, 2016, Proceedings’, *in F.J. Mata and A. Pont (eds) ICT for Promoting Human Development and*

Protecting the Environment. Cham: Springer International Publishing (IFIP Advances in Information and Communication Technology), pp. 3–13.

Tortoriello, V. (2022) *Definition of a DevSecOps Operating Model for software development in a large Enterprise* (Doctoral dissertation, Politecnico di Torino). Available at: https://webthesis.biblio.polito.it/23649/ (Accessed: 18 July 2023).

Uddin, M. and Rahman, A.A. (2012) 'Energy efficiency and low carbon enabler green IT framework for data centers considering green metrics', in *Renewable and Sustainable Energy Reviews*, 16(6), pp. 4078–4094.

Venters, C.C. et al. (2018) 'Software sustainability: Research and practice from a software architecture viewpoint', in *The Journal of Systems and Software*, 138, pp. 174–188.

Vikram (2015) 'Green computing', in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 767–772.

Wang, J., Palanisamy, B. and Xu, J. (2020) 'Sustainability-aware Resource Provisioning in Data Centers', in *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, pp. 60–69.

Wati, Y. and Koo, C. (2011) 'Green IT Balanced Scorecard: A comprehensive model for sustainable IT management', *Sustainability*, 3(3), pp. 349–366. doi:10.3390/su3030349.

Welter, P. and Benitti, F.B.V. (2013) 'Green metrics to software development: A systematic literature review', in *Journal of Systems and Software*, 103, pp.259–281. Available at: https://www.semanticscholar.org/paper/Green-metrics-to-software-development-A-systematic-Welter-Benitti/ece18755f61c4ecad3ef09981e115b46f9bdad59 [Accessed 6 January 2025].

Winters, T. (2018) ‘Non-atomic refactoring and software sustainability’, in *Proceedings of the 2nd International Workshop on API Usage and Evolution*. New York, NY, USA: Association for Computing Machinery (WAPI ’18), pp. 2–5.

Wlodarczyk, T.W. and Rong, C. (2010) ‘On the Sustainability Impacts of Cloud-Enabled Cyber Physical Space’, in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pp. 597–602.

Wolfram, N., Lago, P. and Osborne, F. (2017) ‘Sustainability in software engineering’, in *2017 Sustainable Internet and ICT for Sustainability (SustainIT)*, pp. 1–7.

Zohaib, M. (2023) ‘Towards Sustainable DevOps: A Decision Making Framework’, in *Journal of DevOps and Sustainable Computing*, Available at: https://doi.org/10.48550/arXiv.2303.11121 [Accessed 6 January 2025].