Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Software Engineering
Master's Programme in Software Product Management and Business

Ahmed Rifat

Green Code - Towards Measuring the Efficiency of Software System Execution

Examiners: Professor Jari Porras
Shola Oyedeji, PhD

Supervisors: Professor Jari Porras
Shola Oyedeji, PhD

ABSTRACT

Lappeenranta-Lahti University of Technology LUT

School of Engineering Science

Software Engineering

Master's Programme in Software Product Management and Business

Ahmed Rifat

Green Code - Towards Measuring the Efficiency of Software System Execution

Master's Thesis

48 pages, 6 figures, 11 tables, 1 appendix

Examiners: Professor Jari Porras, Shola Oyedeji, PhD

Keywords: Software Energy Efficiency, Systematic Literature Review (SLR)

Motivation: Currently, Software energy efficiency is an emerging concern for the software industry and academy. This study investigates "how to measure the energy efficiency or greenness of software systems?". Before measurement, it is important for software professionals to know which elements in each phase of the software development life cycle (SDLC) are affecting the software's energy efficiency.

Aim: The purpose of this study was to find out the elements in each phase of the software development life cycle (SDLC) that are affecting the energy efficiency of software and provide guidelines for those affecting elements.

Method: Systematic literature review (SLR) and snowball approach were applied in this study to investigate the state of art on software energy efficiency. The elements and guidelines were identified from the literature review. Also, those elements were categorized in a mind map through software development life cycle (SDLC) phases.

Results: The study identified crucial elements that are affecting the software's energy efficiency and provided guidelines for enabling the greenness of software.

Implications: The study provides crucial elements and guidelines for improving the energy efficiency of software. This study leads to the development of a framework for helping software professionals to take decisions.

ACKNOWLEDGEMENTS

This study helps me to increase my knowledge about energy-efficient software development. Before starting the study, I was less familiar with the topic and its context. As a software engineer usually, I focused earlier on software performance metrics. After conducting this study, I am now able to think more clearly about the development of energy-efficient software.

I am willing to say thanks to Prof. Jari Porras and Shola Oyedeji, Phd for guiding me through my whole thesis work. They allowed me to conduct this study in depth. So that I can bring crucial results out of my study. Before conducting the study, I was less familiar with how to conduct systematic literature review (SLR). They guided me all the time when I needed a help. It was quite difficult for me to understand the topic in a holistic view. They provided a clear direction for getting the insightful information about this topic. I am lucky that I found Prof. Jari Porras as my supervisor.

However, I am also willing to say thanks to all of my colleagues for supporting me to conduct my study. In thesis time, I learned a lot of research insights from people around myself.

TABLE OF CONTENTS

(Arabic style page numbering starts on this page)

LIST OF SYMBOLS, ABBREVIATIONS AND TERMS

Code	A set of instructions for execution purpose
Green Software	Energy Efficient Software
SDLC	Software Development Life Cycle
SLR	Systematic Literature Review

1 INTRODUCTION

Software systems are design and developed as technical artefacts for solving real-world problems to facilitate different processes and activities in companies and organisations. Software systems consumes energy while performing different tasks which contributes to CO2 emissions. Consequently, software systems have enormous effects on users, society and the environment. Well designed software system that is energy efficient will help promote environmental sustainability.

1.1 Context and Research Problem

Due to digitalization software plays a more crucial role in organizational contexts and processes. But software is an intangible technical artifact. It is an outcome of a set of instructions (code) written by programmers or developers using different programming languages. Programmers develop software based on the customers' requirements. Software is an intangible technical artifact that provides functionalities to fulfill these requirements. Software fulfills the intended purposes by providing functionalities. An efficient software must run smoothly and follow the given conditions by the runtime environment. However, Software cannot run alone without the hardware. Software needs hardware to execute the functionalities. Therefore, the software uses hardware resources with energy. Also, it is important to consider the energy consumption of the software for minimizing the environmental effects.

The energy efficiency of the code or software is less concerning to the programmers according to Pang et al. [21]. Pang et al. [21] stated that most of the time programmers only focus on hardware energy efficiency and in the development of software merely the run-time performance is optimized. Software developers or programmers focus mostly on functional requirements. But the energy efficiency is a non-functional requirement as stated by Abdulsalam et al. [1]. Most software developers and programmers don't know how to measure and increase the energy efficiency of software or code (Abdulsalam et al. [1]). Couto et al. [6] noticed that currently, the need of developing energy-efficient software is increasing rapidly. Software developers and programmers are becoming more aware of software energy consumption but they still lack the knowledge and tools to deliver energy-efficient software (Couto et al. [6]).

Therefore, the software's energy efficiency concerns are emerging recently. Before developing green or energy-efficient software, software developers and programmers must know what are the elements that need to be considered for delivering energy-efficient software. Therefore, this study was focused on answering the question "Which elements in each phase of the software development life cycle (SDLC) are affecting the software's energy efficiency and how to provide guidelines for enabling the greenness of software ?".

1.2 Research Aim

The research problems discussed above emerged from the need of identifying elements that affect software energy efficiency. Identifying the elements that affect green software is not sufficient for software developers or programmers to think about the energy efficiency of software. They can be acquainted with those elements, but it was rarely possible for them to make decisions or choices on those elements while aiming to deliver energy-efficient software.

Therefore, software developers and programmers needed to know how they can take decisions on those elements. Hence, the guidelines for those elements are needed to enable the energy efficiency concerns of software to be taken into account.

Concurrently, this study was aimed at providing guidelines for those elements to enable the greenness or energy efficiency of software. In this way, this study support software developers and programmers to consider energy efficiency concerns in their software systems.

1.3 Research Questions

The recent research articles [1,6] highlighted that currently software developers and programmers are becoming more aware of the energy efficiency of software.

Currently, energy-efficient software development concerns are increasing rapidly. Moreover, Couto et al. [6] reported that software professionals and programmers lack the knowledge and tools for delivering energy-efficient software. Hence, software developers and programmers need to enhance their knowledge about how they can develop green software. Thus, there is a necessity to make resources and tools available for them to gain knowledge about developing green software.

The opening research aim and research question for this thesis were "How can the greenness or energy efficiency of software be measured?". Before we can do measurements, it is crucial to identify the elements that are affecting the energy efficiency of software. That's why this study aimed to address the elements that affect the energy efficiency of software. The result of this study shows how software developers and programmers can think about energy-efficient or green software. Also, it will increase the awareness of software developers and programmers about green software. The guidelines will lead them to identify energy-inefficient decisions or choices for their software. These insights brought a new direction to my research. Therefore, in this study I needed to answer research questions:

RQ1: What are the elements affecting the energy efficiency (greenness) of software?

RQ2: What are the guidelines of those elements for developing energy-efficient software?

The clearly defined purpose will allow me to identify crucial elements that have influenced software energy efficiency and prepare guidelines for those elements. The results of this study will help software developers and programmers to identify elements affecting energy efficiency of software system and support developing green software system.

1.4 Research Methodology

This study investigated state-of-the-art research articles to find out answers to the research questions. The articles that were covered in this study provided insightful findings. The systematic literature review (SLR) was used in this study as the research method. Also, snowballing was used in the systematic literature review to find and enhance the relevant topic information. The results of this study were analyzed through Software Development Lifecycle (SDLC).

The study design and research process are detailed in section 3.

1.5 Delimitations

This study highlighted major elements that are affecting the energy efficiency of software and provide guidelines for tackling those elements. There are still some limitations in this study. Some aspects of the Software Development Lifecycle (SDLC) were not covered elaborately. Because there is a lack of previous research on those specific topics. The snowball approach helped to gather relevant articles easily.

2 RELATED WORK

Nowadays, software energy efficiency related studies are increasing rapidly. In earlier, most research studies focused on hardware energy efficiency as stated by Drangmeister et al. [7]. Also, Drangmeister et al. [7] mentioned that major research articles highlighted the energy efficiency of hardware and how to produce energy-efficient hardware. The reason behind considering the energy efficiency of hardware is that ICT systems use a lot of hardware resources. The hardware-level energy measurement is an easier task than the software-level energy measurement and software has a great impact on overall energy consumption according to Drangmeister et al. [7]. Therefore, considering the energy efficiency of the software is crucial nowadays.

However, several studies focused on the energy efficiency of software. This section was depicted with state-of-the-art related articles. Couto et al. [6] stated that software developers and programmers are now becoming more aware of the energy efficiency of software.

Capra et al. [4] conducted a study of 63 open-source applications to observe how the software’s functional type affects its energy efficiency. They reported that Enterprise resource planning (ERP), text and gaming applications are less energy efficient than other software or applications. They also mentioned that CPU-intensive tasks are more energy inefficient than ram-intensive tasks performed by software. They identified that development environments have a great effect on software energy consumption. Grosskop et al. [35] conducted a study discussing application-level energy measurement and optimization. They discussed more on how the application-level energy consumption can be optimized. Besides, they reported some crucial models and metrics for evaluating application-level energy characteristics. They highlighted algorithm-level software energy optimization. Capra et al. [4] did not cover the application-level energy optimization recommendations. However, Grosskop et al. [35] provided a holistic view that is concerning application-level energy measurement and optimization.

Besides, Erdélyi et al. [8] identified some special factors that need to consider when developing green software. They conducted their study through the software development life cycle (SDLC) but did not cover all phases of SDLC. They highlighted some key points (elements) and activities that need to consider when developing green software. Capra et al. [4] discussed just the effect of CPU and RAM intensive usages on software energy efficiency, but Erdélyi et al. [8] focused on key points and activities for delivering green software. However, Salam et al. [37] conducted a study through the systematic literature review (SLR) to identify factors that affect the development of green software. They reported 20 factors that have a great influence on developing green or energy-efficient software. They claimed that these 20 factors need to be considered by software professionals for delivering green software to customers.

Manotas et al. [18] conducted a study with the help of 464 software practitioners using interviews and surveys. They captured the software practitioners' thoughts about energy efficiency when developing software through the software development life cycle (SDLC) phases. They also discussed the implications of the findings. Even though, Manotas et al. [18] provided just an overview of what to do for developing green software, nothing about the elements that influence software energy efficiency.

Kocak et al. [16] conducted a study to identify all the tradeoffs between software sustainability and software quality requirements. They proposed an ANP framework for identifying the tradeoffs. They highlighted some green metrics for estimating the energy efficiency of software. These metrics (CPU usage etc.) have a great influence on software energy efficiency. However, some other metrics or elements (CPU Idle state, threads etc.) affect software energy efficiency. Marantos et al. [36] conducted a study for developing a tool for evaluating the energy consumption of a software application. They also proposed a framework that allows for evaluating software energy consumption. This framework and tool can support a wide variety of CPUs. They also claimed that their proposed framework and tool will help software developers to evaluate the energy consumption of a software application and make decisions based on the evaluation results.

Georgiou at el. [12] depicted the techniques and tools that are convenient for developing energy-efficient software. They conducted their study by reviewing relevant articles in all the software development life cycle (SDLC) phases. They captured some crucial aspects and elements that we need to consider for developing energy-efficient software. Even though they highlighted some crucial aspects and elements but they provided less information about the guidelines of those elements.

In essence, no study has captured all the major elements that need to consider for developing energy-efficient software. Besides, there is less information about how to deal with those elements to enable the greenness of software. This thesis study focuses on finding all major elements that affect software energy efficiency. Also, this thesis study provides some crucial guidelines to deal with those elements to achieve energy-efficient software.

3 RESEARCH PROCESS

Before starting this study, it was essential to design the research for achieving the research goal. This study used the software development life cycle (SDLC) to categorize the findings. This study was followed by the research process depicted in **Figure 1: Research Roadmap**. In the beginning, this study aimed to find the answer to "How can the greenness or energy efficiency of software be measured?". Before answering this question it is crucial to focus on the elements that are affecting the energy efficiency of software. After that, the research questions of this study were finalized. This study used a systematic literature review (SLR) with the snowballing approach to finding the answer to the research questions (**RQ1, RQ2**). This study depicted the crucial elements and guidelines that are relevant to the elements. This study highlighted that these crucial elements have great influences on the energy efficiency of software system. This study depicted a mind map (**Figure 6**: **Mind Map**) for illustrating the elements that are affecting the energy efficiency of software. Also, the guidelines (**Table 11: Guidelines)** for these elements will help software professionals to make decisions on these crucial elements for delivering green software.

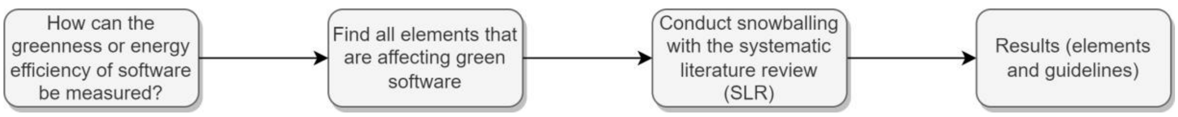

Figure 1: Research Roadmap

The systematic literature review (SLR) with a snowball approach was used in this study as the research method. The Snowball approach helped this study to find similar topics and state-of-the-art articles to reduce the noise of article selection. The research methodology of this study was depicted in **Figure 2: Research Methodology**.

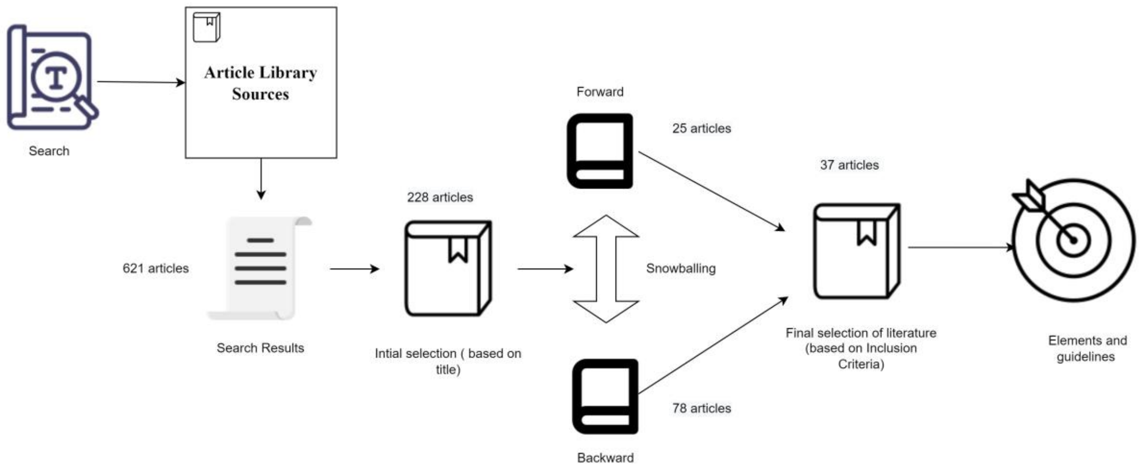

Figure 2: Research Methodology

In below, I described each step of the **Research Methodology (Figure 2)** elaboratively.

3.1 Search Strings and Article Library Sources

To get informative articles this study looked through each of the SDLC phases with different aspects to capture elements that influence software energy consumption.

Main search string format: (Keyword) AND ("energy" OR "efficiency" OR "Usage" OR "factors") AND AfterYear=2012 AND BeforeYear=2022.

Source
Wiley
Springer
Semantic Scholar
Sciencedirect
Sage Pub
IEEE
ACM

Table 1: Article Library Sources List

SDLC_PHASE	ASPECT
Requirements Engineering	Functional Requirements
Requirements Engineering	Software Quality Model
Requirements Engineering	Requirement Elicitation Strategies
Software Design	Software Architecture
Software Design	Design Patterns
Software Development	Development Environment
Software Development	Programming Language
Software Development	Parallel Computing
Software Development	Approximate Computing
Software Development	Cloud Computing
Software Development	Data Structure
Software Development	Database
Software Development	High Performance Computing
Testing and Verification	Testing with Energy Measurement
Deployment	Deployment Environment
Deployment	Load Balancing
Maintenance	Refactoring

Table 2: SDLC Phases and Aspects Keywords

In the beginning, this study was focused on finding articles that are relevant to

the aim of the study. After that, the major articles' insights indicate that this study needed to focus on different aspects in each software development life cycle (SDLC) phase to orchestrate elements that are affecting the energy efficiency of software. Therefore, to conduct this study I used (**Keyword = "SDLC_PHASE")** in the main search string format to get the aspects that need to consider on software energy efficiency from article library sources **(Table 1: Article Library Sources List)** for each software development life cycle (SDLC) phase. Once I got crucial aspects (**Table 2: SDLC Phases and Aspects Keyword**) then I again performed search activities in article library sources (**Table 1: Article Library Sources List)** with (**Keyword = "SDLC_PHASE" OR "ASPECT")** in the main search string format to get elements and guidelines. After this stage, I got several articles from different article library sources (**Table 1: Article Library Sources List)**. The below (**Figure 3: No of articles from primary search with article library sources**) depicts the total number of articles from different article library sources (**Table 1: Article Library Sources List)**.
I got a total of 621 articles as a primary search article set.

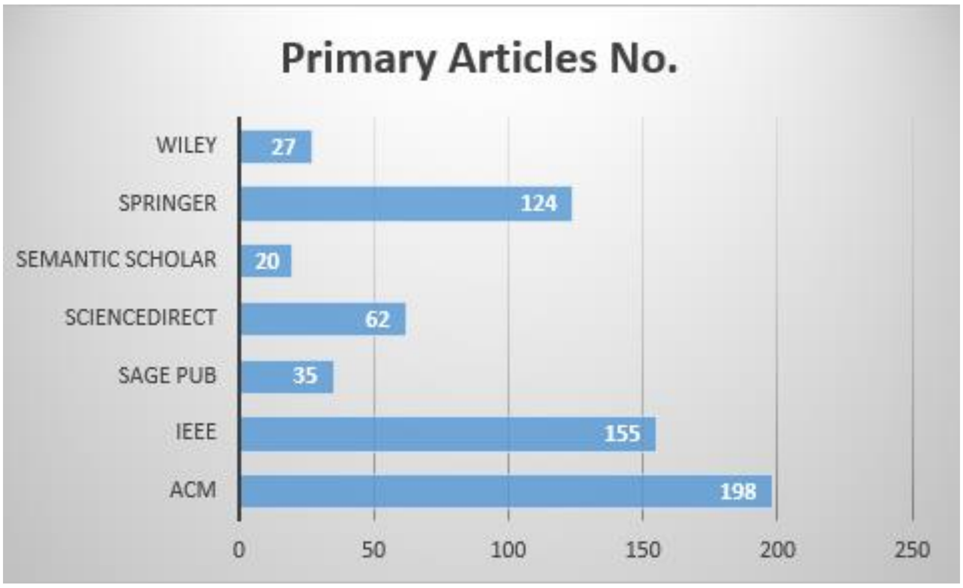

Figure 3: No of articles from primary search with article library sources

3.2 Initial Selection of literature

In this stage, the initial set of articles was selected based on inclusion criteria **(IC01-IC03)**. Previously, the energy efficiency of hardware was considered broadly. However, the energy efficiency of the software is also depending on the energy efficiency of the hardware according to Pang et al. [21]. At this stage, I had a good number of articles for answering my research questions. After this stage, I got a total of 228 articles as the initial articles set. Below (**Figure 4: No of articles in initial articles set with article library sources**) depicts the total number of articles from different article library sources (**Table 1: Article Library Sources List)**. After this stage, there was a more precious number of articles with information than in earlier primary search sets.

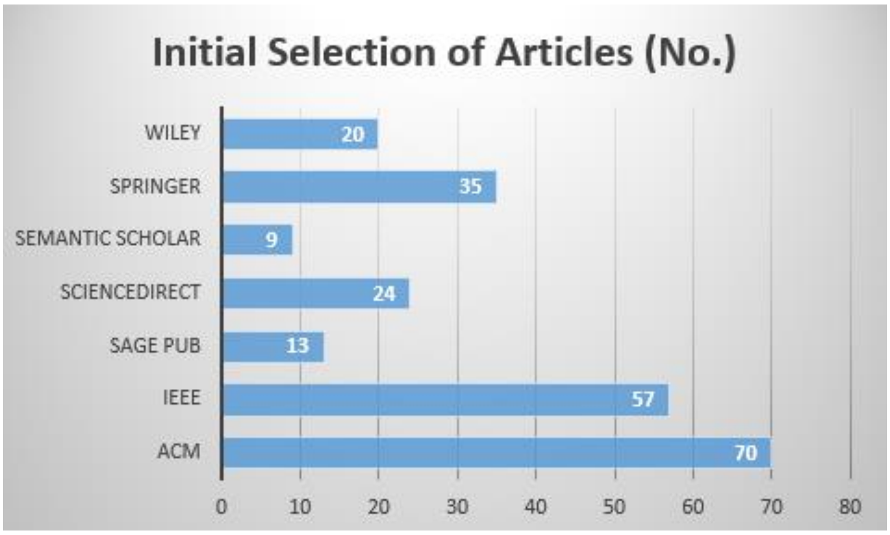

Figure 4: No of articles in initial articles set with article library sources

3.3 Snowballing

After getting the initial set of articles, I performed snowballing approach to enhance the number of relevant articles to capture the most insightful information for each aspect. I conducted two rounds of backward and forward snowballing.

From backward snowballing, I got 78 articles and from forward snowballing I got 25 articles that are relevant to my research study.

3.4 Final Selection of literature and Data Extraction

In this stage, I got a total of (228 from initial selection + 78 from backward snowball + 25 from forward snowball) 331. I used my research inclusion criteria (**IC01-IC05**) to make the final selection of articles. The inclusion criteria helped me to identify all crucial articles from a large number of articles. I got 37 articles after applying the research inclusion criteria (**IC01-IC05**).

The research inclusion criteria are mentioned below.

IC01: The articles are written in English and the title is relevant to the study topic.

IC02: The abstract and keywords are relevant to the study topic.

IC03: At least some discussions or implications on enabling the energy efficiency of software or some empirical evaluations.

IC04: Articles that highlight topic-related crucial information.

IC05: Articles that mention future research directions.

In below figure 5, I depicted the number of articles I chose from the article library sources (**Table 1: Article Library Sources List**).

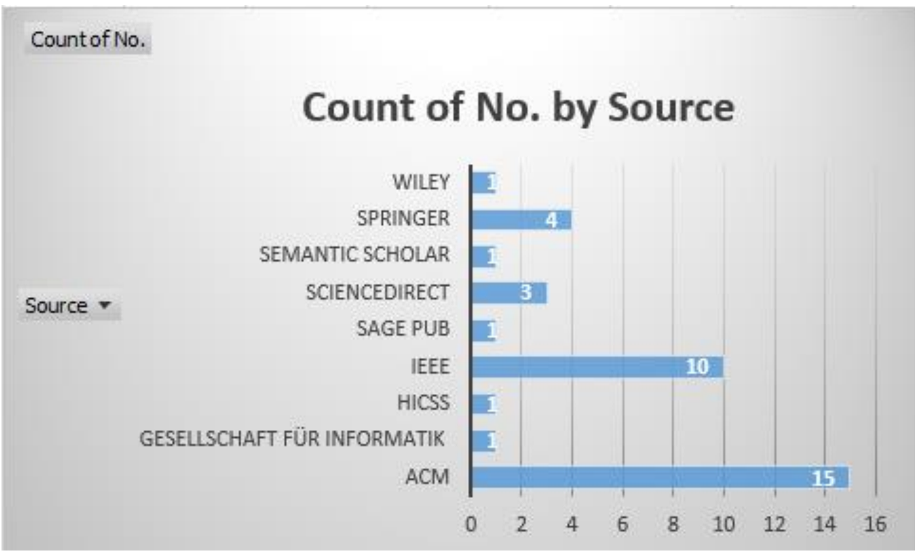

Figure 5: No of articles in final selection with article library sources

In data extraction, to perform the data extraction activities I used a strategy (**Table 3: Data extraction strategy**) to extract the necessary data from the final article sets.

Data	Articles Properties	Mapping to Research Questions
Relevant Information	Similar Studies, Studies have the relevant idea and thoughts	RQ1,RQ2
Elements	Studies have discussions or implications for enabling energy efficiency of a software	RQ1
Insights or recommendations	Experiment studies or Empirical evaluations,Studies have future research directions	RQ2

Table 3: Data extraction strategy

The data extraction strategy helped me to find the proper data or information for mapping with the research question's requirements.

3.5 Data Analysis

The data analysis piece of work was conducted with a qualitative data collection approach during the systematic literature review (SLR) process by applying the content analysis technique. The below data analysis and coding technique (**Table 4: Data Analysis and coding technique**) helped to identify the common patterns from the articles that were collected from the systematic literature review (SLR). The aim and research questions facilitated the content analysis activities to separate the study-related information and code that information. The below table (**Table 4: Data Analysis and coding technique**) depicts the detailed analysis and coding technique that was used in this study. The coding technique used in data analysis orchestrated the information properly for answering the research questions of this study.

Category	Description	Code
SDLC Phases aspect	The major aspects that are relevant to the each SDLC phase	Aspects
Affecting Elements and Factors	The elements or factors that are belong to a specific aspects	Elements
Implication and Insightful discussion with implication	The information of each element or factor to deal with software enegy efficiency	Guidelines

Table 4: Data analysis and coding technique

3.6 Mind Map

The mindmap depicted all key elements affecting the software's energy efficiency. All key elements were categorized in the mind map through software development life cycle (SDLC) phases. First, the mindmap depicted all aspects of each software development life cycle (SDLC) phase. After that, it depicted all elements of each aspect affecting the software's energy efficiency.

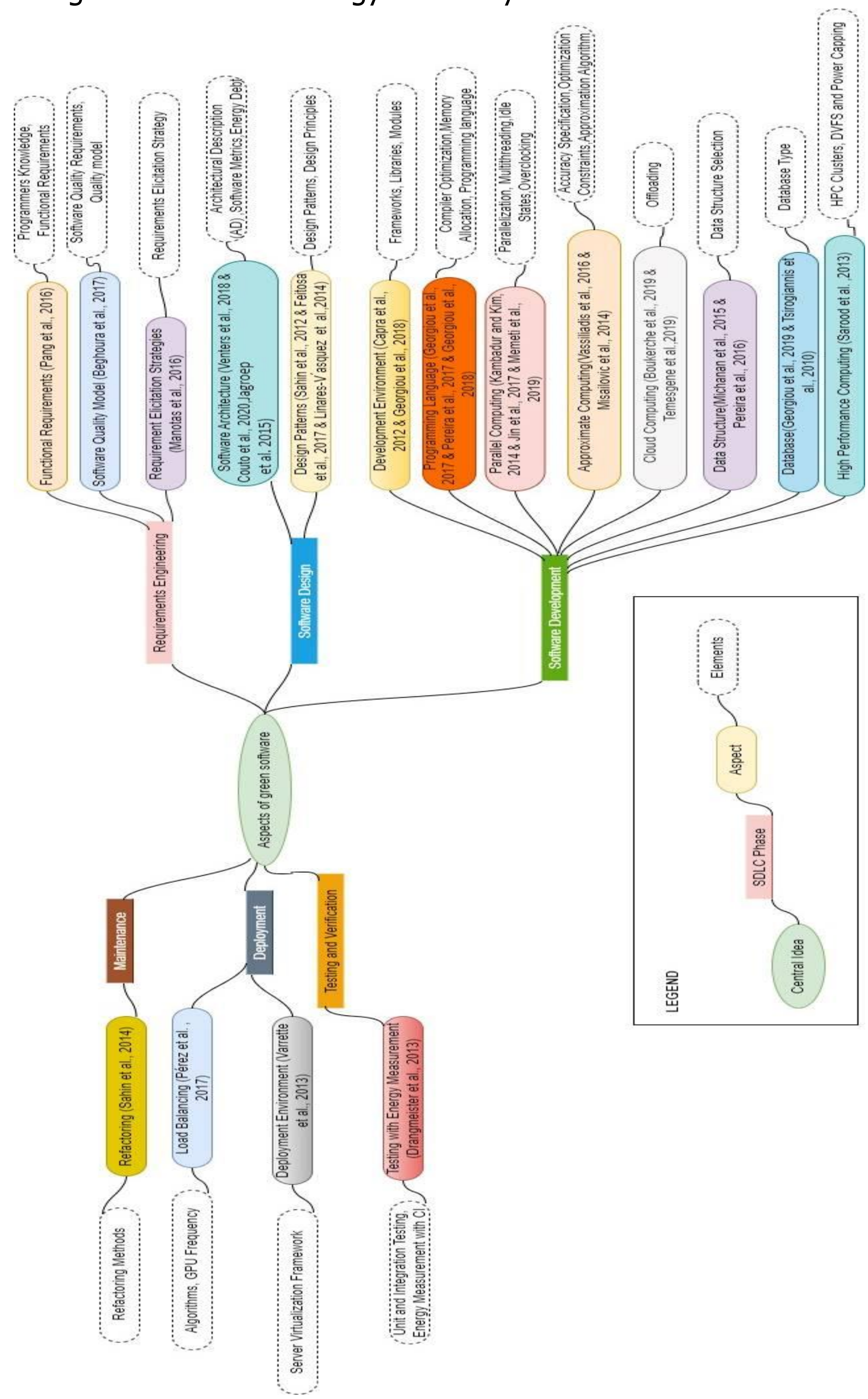

Figure 6: **Mind Map**

4 LITERATURE REVIEW RESULTS

The systematic literature review (SLR) with the snowball approach allowed me to gather crucial insights for fulfilling the study aim and research questions. After conducting the literature review, this study identified major elements that are affecting the energy efficiency of software. Besides, this study also highlighted guidelines for those elements that were gathered from the literature review work. The mind map (**Figure 6**: **Mind Map**) visualized the elements that are affecting the energy efficiency of software. The guidelines were also listed in a table accordingly. This categorization will also help software developers to easily correlate with development work. This section was structured as first highlighting all elements of a specific aspect with bold keywords and providing guidelines for those elements with again bold keywords.

4.1 Requirements Engineering

Requirements engineering is the phase where the requirements of the software are elicited and specified. This SDLC phase is crucial for emphasizing the green aspects of the software. This study identified several key elements in the requirements engineering phase that are affecting the energy efficiency of software. The programmer's knowledge has a great effect on the greenness of software according to Pang et al. [22]. They mentioned that programmers must have to know what they need to do for improving the energy efficiency of software. They also reported that the green requirements belonged to the corresponding **functional requirements**. Pang et al. [22] also stated that green requirements can be identified by correlating with relevant functional requirements. Also, Venters [34] et al. mentioned that the sustainability of software has been greatly affected by software quality attributes. Besides, Beghoura et al. [2] stated that the **quality model** needs to consider for improving the energy efficiency of software. Also, they mentioned that software quality requirements have an enormous effect on the energy efficiency of software. Manotas et al. [18] highlighted that the **requirements elicitation strategy** needs to consider for defining the green requirements and determining the energy usage acceptance criteria. In Table 5, the guidelines for the above elements are provided. The guideline (**GL01**) is described as "Software developers should specify the functional and non-functional requirements. Non-functional requirements are attached to the **functional requirements**. Mark energy requirements from non-functional requirements." according to Pang et al. [22]. The guideline (**GL02**) is highlighted as "Software developers should use green efficiency attributes in software **quality model** to facilitate the energy efficiency techniques in the software development stage." according to Beghoura et al. [2]. The guideline (**GL03**) explained as "Energy-related (green) requirements are stated usually in other words. Software developers should use an **elicitation strategy** to identify a task's energy usage acceptance level. For instance (e.g., mobile location applications should be able to run from available charging sources when driving a car or bike). So, the energy requirement is visible behind the stated requirement. Also, energy requirements need to define with their acceptance levels." mentioned by Manotas et al. [18].

SDLC Phase	Aspect Name	No.	Guidelines	Description	Reference
Requirement Engineering	Functional Requirements	GL01	Identify energy requirements by correlating them with functional requirements	Software developers should specify the functional and non-functional requirements. Non-functional requirements are attached to the functional requirements. Mark energy requirements from non-functional requirements.	Pang et al. [22]
	Software Quality Model	GL02	Use energy efficiency as a characteristic in software quality model	Software developers should use green efficiency attributes in software quality model to facilitate the energy efficiency techniques in the software development stage.	Beghoura et al. [2]

	Requirement Elicitation Strategies	GL03	Use a requirement elicitation strategy for identifying a task's energy usage acceptance level	Energy-related (green) requirements are stated usually in other words. Software developers should use an elicitation strategy to identify a task's energy usage acceptance level. For instance (e.g., mobile location applications should be able to run from available charging sources when driving a car or bike). So, the energy requirement is visible behind the stated requirement. Also, energy requirements need to define with their acceptance levels.	Manotas et al. [18]

Table 5: Requirements Enginerring Guidelines

4.2 Software Design

The Software design is the phase where the specification of technical artifact (software) is derived from the customer's requirements. The software design is the second phase of SDLC. To specifically identified crucial elements that are affecting the energy efficiency of software in the software design phase, this study categorized Software design as Software Architecture and Software Design Patterns (coding pattern).

Software Architecture: It is the structure of a software system. In another term, it is the backbone of a software system. Many research studies have identified that software architecture has a great impact on the energy efficiency of software [5,13,33]. Jagroep et al. [13] identified that **Architectural Description (AD)** has a great impact on the energy efficiency of software. Venters et al. [33] studied the impact of software architecture on the greenness of software. They reported that **software quality metrics** have great impacts impact on the energy efficiency of software. In other words, software quality metrics are needed to evaluate the software architecture for ensuring the greenness of software. Couto et al. [5] introduced **energy debt** as the new concept of energy efficiency of software. They noticed that as code smells, energy smells are the energy-greedy coding practices. In Table 6, the guidelines for the above elements are provided. The guideline (**GL04**) explained as "Software architects should modify **Architectural Description (AD)** (different views and others) to enhance greenness of software according to Jagroep et al. [13]. The energy consumption perspectives are also mentioned in the table". The guideline (**GL05**) explained "Usually, **software quality metrics** are used to estimate the quality of a system. For example, Bad coding practices will make it more complex to evolve software. Bad coding practices include redundant APIs. So, there is an issue with the architectural functional view." according to Venters et al. [33]. The guideline (**GL06**) explained as "Energy debts are raised by the energy smells. Energy smells are energy-greedy or inefficient coding practices. It is different from technical debt. The energy debt increases as the number of times a specific release performs tasks." mentioned by Couto et al. [5].

Software Design Patterns: The software design pattern is about how software's codes are efficiently orchestrated. Several studies identified that different software **design patterns** have different impacts on the energy efficiency of software [9,17,26]. Feitosa et al. [9] conducted an empirical study on the popular gang of four (GoF) patterns.

According to their findings gang of four (GoF) patterns are more energy-greedy design patterns compared to others. Linares-V´asquez et al. [17] performed an empirical on different mobile applications' design patterns and found that the APIs and usage patterns consume more energy compared to others. They identified different **design principles** that have great impacts on the energy efficiency of software. Sahin et al. [26] conducted a study to observe 15 design patterns' energy impacts on software. They used an application named "Proxy" for evaluating the impacts before and after applying a design pattern. It is pretty complex to estimate a design pattern before evaluation. They found that some design patterns are more energy efficient the others. In Table 6, the guidelines for the above elements are provided. The guideline (**GL07**) explained as "A **design pattern** has impacts on software energy efficiency. The impacts of design patterns are not constant. Avoid patterns that have a complex way of implementation. For example, the Decorator pattern is not an energy-efficient one. Decorator pattern creates complex objects except for inheritance.” according to Sahin et al. [26]. The guideline (**GL08**) explained as “**Design patterns** have similar or higher energy consumption levels in most cases. However, it is a good idea to use design patterns when pattern-related methods with a higher number of method invocations are required for implementing complex tasks.” mentioned by Feitosa et al. [9]. The guideline (**GL09**) explained “It is a pretty energy-consuming task for android applications to navigate views. So, using Model-View-Controller (MVC) pattern allows refreshing each view when need load contents. Which is an energy-expensive task. Software developers must avoid it unless they have no alternatives.” according to Linares-V´asquez et al. [17]. The guideline (**GL10**) explained as “Data hiding in an android application is energy consuming task. So, **Design principles** relevant to data orchestration can decrease the energy efficiency of an android application.” mentioned by Linares-V´asquez et al. [17].

SDLC Phase	Aspect Name	No.	Guidelines	Description	Reference
Software Design	Software Architecture	GL04	Architectural Description (AD) must have to alter with energy consumption perspectives.	Software architects should modify AD (different views and others) to enhance greenness of a software. The energy consumption perspectives are. 1.Identify requirements that ensure less energy consumption -Context View: How can the software achieve green goals? -Functional View: How much energy can a function consume? -Information View: What can we do to optimize information flow for reducing energy consumption? -Concurrency View: How we can arrange functional elements to be attached to processes and how the hardware will execute all these processes? -Development View: Can we choose green algorithms for the software? -Operational View: How run time issues can be solved to reduce energy consumption? -Deployment View: Which hardware aligned with what processes? 2. Make energy profile: Determine what level of energy the software consume.	Jagroep et al. [13]
				3. Check energy profile with requirements: Evaluate the requirements with the energy profile for further adjustments. 4. Determine the adjustments: Make some adjustments based on energy hotspot elements. 5. Check the adjustments: After the adjustments, check if the adjustments have positive effects on energy consumption.	

	Software Architecture	GL05	The software quality metrics can be used to evaluate the software architecture's greenness.	Usually, software quality metrics are used to estimate the quality of a system. For example, Bad coding practices will make it more complex to evolve software. Bad coding practices include redundant APIs. So, there is an issue with the architectural functional view.	Venters et al. [33]
	Software Architecture	GL06	Energy debts should be mitigated to ensure green software.	Energy debts are raised by the energy smells. Energy smells are energy-greedy or inefficient coding practices. It is different from technical debt. The energy debt increases as the number of times a specific release performs tasks.	Couto et al. [5]
	Design Patterns	GL07	Design patterns have a variety of impacts on software energy consumption. So, software developers should choose simple design patterns for developing software or system.	A design pattern has impacts on software energy efficiency. The impacts of design patterns are not constant. Avoid patterns that have a complex way of implementation. For example, the Decorator pattern is not an energy-efficient one. Decorator pattern creates complex objects except for inheritance.	Sahin et al. [26]
	Design Patterns	GL08	Nonpattern is better than the use of any design pattern. In complex cases of implementation, design patterns can be an energy-efficient choice.	Design patterns have similar or higher energy consumption levels in most cases. However, it is a good idea to use design patterns when pattern-related methods with a higher number of method invocations are required for implementing complex tasks	Feitosa et al. [9]
	Design Patterns	GL09	Android mobile applications should not use Model-View-Controller (MVC) pattern when it has many views.	It is a pretty energy-consuming task for android applications to navigate views. So, using Model-View-Controller (MVC) pattern allows refreshing each view when need load contents. Which is an energy-expensive task. Software developers must avoid it unless they have no alternatives.	Linares-Vásquez et al. [17]

	Design Patterns	GL10	Design principles and energy efficiency have a tradeoff. Software developers should decide carefully on which one they are willing to focus on.	Data hiding in an android application is energy consuming task. So, Design principles relevant to data orchestration can decrease the energy efficiency of an android application.	Linares-Vásquez et al. [17]

Table 6: Software Design Guidelines

4.3 Software Development

Software development is the phase where the specification of software turns into a technical artifact. Software development is the third phase of SDLC. To identify crucial elements that are affecting the energy efficiency of software in the software development phase, this study divided the software development phase into eight crucial aspects as Development Environment, Programming Language, Parallel Computing, Approximate Computing, Cloud Computing, Data Structure, Database, High-Performance Computing. High-Performance computing covers parallel computing by utilizing resources.

Development Environment: The software development environment consists of a set of tools used by software developers to develop software [10,4]. Also, the development environment mostly consists of **frameworks**, **libraries** and **modules** according to Georgiou et al. [10] However, Capra et al. [4] reported that the development environment strongly affects software energy consumption. They also reported that there is a trade-off between performance and energy efficiency. Besides, Georgiou [10] et al. claimed that the selection of **frameworks**, **libraries,** and **modules** in software development is also affected the energy consumption of software. In Table 7, the guidelines for the above elements are provided. The guideline (**GL11**) explained "The use of development environment (**frameworks, libraries**) for small and medium size applications can reduce the energy consumption. For large applications, the use of an application development environment (**frameworks, libraries**) is not a good choice for energy efficiency. Because the development environment is used for optimizing development efficiency. There is a trade-off between development efficiency and energy efficiency (based on application size)." stated by Capra et al. [4] The guideline (**GL12**) explained as "If software developers develop a new function from scratch, then it is quite difficult to ensure the energy efficiency of that." mentioned by Capra et al. [4] The guideline (**GL13**) explained as "Large applications are created by integrating **modules**. So, that can easily affect the overall software energy consumption." mentioned by Capra et al. [4] The guideline (**GL14**) explained as "**Libraries** that are written in native language have greater energy efficiency over other **libraries**." mentioned by Georgiou et al. [10].

Programming Language: The programming languages are used for creating technical artifacts. Several studies identified that different programming languages have various effects on the energy efficiency of software [6,10,11,23]. Georgiou et al. [11] conducted an empirical study to observe the energy consumption of several programming languages with similar implementations. They stated that several **programming languages** have various effects on software energy consumption. Georgiou et al. [10] conducted a study to evaluate the energy consumption behaviors of different programming languages. They claimed that **compiler optimization** has a great impact on software energy efficiency. Pereira et al. [23] conducted a study to observe the relationship between energy, time and memory on software energy. They identified that memory allocation has a great impact on software energy efficiency. In Table 7, the guidelines for the above elements are provided. The guideline (**GL15**) explained as "Software professionals must have to choose a **compile-time performance optimization flag** for ensuring energy efficiency." stated by Georgiou et al. [10]. The guideline (**GL16**) explained as "If we consider, the four **memory allocation** choices (malloc, new, array, and vector). The malloc is the best for energy-efficient choice." according to Abdulsalam et al. [1]. The guideline (**GL17**) explained as "C, C#, and JavaScript are the best **programming languages** for energy-efficient software development.” [10,11,23].

Parallel Computing: Parallel computing is the computing mechanism where a big functional program is divided into smaller problems to execute simultaneously. To articulate the effects of parallel computing on software’s energy consumption, this study identified several elements that are affecting the software’s energy consumption. Several studies claimed that parallel computing has enormous effects on software’s energy consumption [14,15,19]. Jin et al. [14] conducted a study to perform parallel computing for identifying elements that are affecting the software’s energy consumption. They reported that **Parallelization** and **Multithreading** have great effects on the software’s energy consumption. Kambadur et al. [15] surveyed the measuring energy of 41 applications with 220 configurations. They reported that effective parallelization and compiler optimization reduces overall energy consumption. **Overclocking** has also effects on software’s energy efficiency. However, the CPU in **Idle state** reduces energy consumption. Memeti et al. [19] performed a literature review for software optimization methods in parallel computing using meta-heuristic algorithms and machine learning. They mentioned several approaches that used meta-heuristic algorithms and machine learning to optimize compile and run-time aspects. In Table 7, the guidelines for the above elements are provided. The guideline (**GL18**) explained as “Utilizing and effectively using CPU cores and threads can reduce the energy consumption of software.” according to Kambadur et al. [15]. The guideline (**GL19**) explained as “An equal number of cores and threads can increase the performance and energy efficiency of a **multithreaded** application.” claimed by Jin et al. [14]. The guideline (**GL20**) explained as “CPU **Idle** and sleep states help to reduce the energy consumption of software. In an idle time of software, the CPU idle and sleep states help to reduce energy consumption.” stated by Kambadur et al. [15]. The guideline (**GL21**) explained as “**Overclocking** can reduce the runtime of software execution but it has fewer effects on software energy efficiency.” stated by Kambadur et al. [15].

Approximate Computing: Approximate computing is a mechanism for developing a technical system to reduce computation complexity and save energy consumption of that technical system. Also, parallel computing aims at maximizing the parallel execution of computing tasks, approximate computing aims to reduce the number of computations needed according to Vassiliadis et al. [32]. Several studies identified that approximate computing has enormous effects on software's energy consumption [21,32]. Misailovic et al. [21] conducted a study to develop a software base framework for precise approximate computing. It is an optimization (Chisel) framework that chooses approximation kernel operations to minimize energy consumption and maintain the accuracy of computation. They claimed that for choosing precise approximation we need **Accuracy Specification** (hardware), **Optimization Constraints**. These have a great effect on software energy consumption. Also, they mentioned approximate computing (**Approximation Algorithm**) has a great effect on software energy consumption. Besides, Vassiliadis et al. [32] conducted a study to identify the computation significance of approximate computing automatically. They claimed that there is a trade-off between output quality and approximate computing for software. In Table 7, the guidelines for the above elements are provided. The guideline (**GL22**) explained as "By defining specific **accuracy specification** in approximate computing, the software can achieve a significant amount of energy efficiency." stated by Vassiliadis et al. [32]. The guideline (**GL23**) explained as "In approximate computing, we need to define optimization constraints properly that leads expected accuracy specification for energy efficiency." as stated by Misailovic et al. [21]. The guideline (**GL24**) explained as "If we use an optimization algorithm in approximate computing then we can achieve energy efficiency in computational works." according to Misailovic et al. [21].

Cloud Computing: Cloud computing refers to the cloud resources for performing computational tasks. Nowadays, offloading (cloud computing) is a quite popular mechanism for performing computational tasks in cloud computing [29]. Mobile and small devices have limited CPU and storage capacity to perform computational tasks and **offloading** can help in this issue according to Temesgene et al. [29]. Cloud computing has enormous effects on software's energy consumption according to Boukerche et al. [3]. Also, **offloading** can support sustainability (energy-efficient perspective) according to Temesgene et al. [29]. Boukerche et al. [3] conducted a study of energy-aware offloading aspects. They illustrated state-of-the-art protocols, algorithms and architectures for energy-efficient offloading towards green(energy-efficient) software development. Temesgene et al. [29] conducted an experimental study to observe how offloading of tasks helps with performance improvement and energy efficiency of resource-constrained devices. They showed that offloading provides better performance and reduces energy consumption. In Table 7, the guidelines for the above elements are provided. The guideline (**GL25**) explained as "In **offloading**, if we consider energy efficiency then there is an effect in task efficiency. Always there is a trade-off between the energy efficiency of offloading and task efficiency." according to Boukerche et al. [3]. The guideline (**GL26**) explained as "Mobile devices have fewer resources. So, **offloading** in the nearest location allows energy efficiency in computation." according to Temesgene et al. [29].

Data Structure: Data Structure is a mechanism for managing data. Several studies identified that the choice of **Data Structure** has a great effect on the energy efficiency of software [20,24]. Michanan et al. [20] conducted an empirical study for selecting automatic data structures based on tasks. They investigate several types of data structures (e.g., Interface based). However, they showed in their study that C5 data structures can be dynamically selected by using machine learning technologies. Besides, Pereira et al. [24] conducted an empirical study on Java Collection Framework (JFC) energy consumption evaluation. They claimed that Data Structure selection in java has a great effect on the energy efficiency of software. In Table 7, the guidelines for the above element are provided. The guideline (**GL27**) explained "To ensure green software (with energy efficient computing) we have to choose the right data structure for the right task. C5 **data structures** can be selected based on tasks and energy efficiency." according to Michanan et al. [20]. The guideline (**GL28**) explained as "The optimized Java Collection Framework (Sets, Lists, and Maps) can reduce energy consumption compared to the original implementation." according to Pereira et al. [24].

Database: The database is used to store the collection of data. Several studies identified that the choice of **Database** has a great effect on the energy efficiency of software [12,30]. The choice of Database for an application is a challenging task. Georgiou et al. [12] conducted a study to observe the effects of the database on software energy efficiency. They compared both relational databases and NoSQL databases. Tsirogiannis et al. [30] conducted a study to observe the power usage of different database workloads on a modern server. They claimed that database performance is related to energy efficiency. In Table 7, the guideline for the above element is provided. The guideline (**GL29**) explained as "Relational Databases are less energy efficient compared to NoSQL. If there is no special requirement on **Database** then the choice of NoSQL will be a good decision." according to Georgiou et al. [12] & Tsirogiannis et al. [30].

High-Performance Computing: High-Performance computing is used by supercomputer nodes and clusters to solve advanced computational problems. Sarood et al. [28] conducted an empirical study to observe the energy consumption behaviors in High-Performance Computing (HPC). They **claimed HPC clusters** allocation, dynamic voltage and frequency scaling (**DVFS**) and CPU **power capping** have a great effect on HPC application's energy consumption. In Table 7, the guideline for the above elements is provided. The guideline (**GL30**) explained as "Organizing HPC clusters with fewer nodes, utilizing DVFS low voltage and Power Capping can reduce the energy consumption in HPC applications." according to Sarood et al. [28].

SDLC Phase	Aspect Name	No.	Guidelines	Description	Reference
Software Development	Development Environment	GL11	The use of a development environment (frameworks, libraries) can reduce the energy consumption of small and medium size applications. But not for large applications.	The use of development environment (frameworks, libraries) for small and medium size applications can reduce the energy consumption. For large applications, the use of an application development environment (frameworks, libraries) is not a good choice for energy efficiency. Because the development environment is used for optimizing development efficiency. There is a trade-off between development efficiency and energy efficiency (based on application size).	Capra et al. [4]
	Development Environment	GL12	Developing new functions from scratch is not energy efficient.	If software developers develop a new function from scratch, then it is quite difficult to ensure the energy efficiency of that.	Capra et al. [4]
	Development Environment	GL13	Integrating modules can increase software's overall energy consumption.	Large applications are created by integrating modules. So, that can easily affect the overall software energy consumption.	Capra et al. [4]
	Development Environment	GL14	Energy efficient choice is choosing native language written libraries.	Libraries that are written in native language have greater energy efficiency over other libraries.	Georgiou et al. [10]
	Programming language	GL15	Using compiler optimization flags can have a great effect on software energy consumption.	Software professionals must have to choose a compile-time performance optimization flag for ensuring energy efficiency.	Georgiou et al. [10]
	Programming language	GL16	The malloc is the most energy-efficient memory allocation choice.	If we consider, the four memory allocation choices (malloc, new, array, and vector). The malloc is the best for energy-efficient choice.	Abdulsalam et al. [1]

	Programming language	GL17	The compiled programming languages are better than semi-compiled programming languages in terms of energy efficiency. The semi-compiled programming languages are better than interpreted programming languages in terms of energy efficiency.	C, C#, and JavaScript are the best programming languages for energy-efficient software development.	Georgiou et al. [10], Pereira et al. [23] & Georgiou et al. [11]
	Parallel Computing	GL18	Effective parallelization and compiler optimizations can reduce software energy consumption.	Utilizing and effectively using CPU cores and threads can reduce the energy consumption of software.	Kambadur et al. [15]
	Parallel Computing	GL19	Multithreaded applications need to be used with an equal number of cores and threads.	An equal number of cores and threads can increase the performance and energy efficiency of a multithreaded application.	Jin et al. [14]
	Parallel Computing	GL20	Idle and sleep states can reduce the energy consumption of the processor. That also increases software energy efficiency.	CPU Idle and sleep states help to reduce the energy consumption of software. In an idle time of software, the CPU idle and sleep states help to reduce energy consumption.	Kambadur et al. [15]
	Parallel Computing	GL21	Overclocking can minorly improve software energy efficiency.	Overclocking can reduce the runtime of software execution but it has fewer effects on software energy efficiency.	Kambadur et al. [15]
	Approximate Computing	GL22	There is a trade-off between energy efficiency and accuracy specification.	By defining specific accuracy specification in approximate computing, the software can achieve a significant amount of energy efficiency.	Vassiliadis et al. [32]
	Approximate Computing	GL23	Optimization constraints must satisfy the accuracy specification.	In approximate computing, we need to define optimization constraints properly that leads expected accuracy specification for energy efficiency.	Misailovic et al. [21]
	Approximate Computing	GL24	An optimization algorithm needs to use for obtaining energy savings.	If we use an optimization algorithm in approximate computing then we can achieve energy efficiency in computational works.	Misailovic et al. [21]

	Cloud Computing	GL25	The energy efficiency in offloading depends on remote (cloud resources).	In offloading, if we consider energy efficiency then there is an effect in task efficiency. Always there is a trade-off between the energy efficiency of offloading and task efficiency.	Boukerche et al. [3]
	Cloud Computing	GL26	Offloading in the nearest location (closest to the end device) has a positive impact on mobile device performance (Also it has energy efficiency)	Mobile devices have fewer resources. So, offloading in the nearest location allows energy efficiency in computation.	Temesgene et al. [29]
	Data Structure	GL27	The choice of data structure has an enormous effect on the energy consumption of software (computing).	To ensure green software (with energy-efficient computing) we have to choose the right data structure for the right task. C5 data structures can be selected based on tasks and energy efficiency.	Michanan et al. [20]
	Data Structure	GL28	The Java Collection Framework (Sets, Lists, and Maps) has also a great impact on the energy consumption of software.	The optimized Java Collection Framework (Sets, Lists, and Maps) can reduce energy consumption compared to the original implementation.	Pereira et al. [24]
	Database	GL29	NoSQL Database is a good choice for ensuring energy-efficient software development.	Relational Databases are less energy efficient compared to NoSQL. If there is no special requirement on Database then a choice of NoSQL will be a good decision.	Georgiou et al. [12] & Tsirogiannis et al. [30]
	High-Performance Computing (HPC)	GL30	Different mechanisms can be used to ensure energy efficiency in HPC applications.	Organizing HPC clusters with fewer nodes, utilizing DVFS low voltage and Power Capping can reduce the energy consumption in HPC applications	Sarood et al. [28]

Table 7: Software Development Guidelines

4.4 Testing and Verification

Testing and Verification is the phase where the requirements and expected functionalities of the software are tested and verified. This phase in SDLC also has a great impact on the software's energy efficiency. Drangmeister et al. [7]

claimed that this is a crucial phase after the development phase where software's energy consumption can be measured and monitored. This study identified several key elements in this phase that are affecting the energy efficiency of software. Besides, this study also gathered crucial guidelines for those elements. Drangmeister et al. [7] noticed that the **energy consumption measurement with continuous integration (CI)** can provide overall energy consumption insights of software. They also noticed that activities in **Unit and Integration Testing** have a great effect on improving software's energy efficiency. In Table 8, the guidelines for the above elements are provided. The guideline (**GL31**) explained as "The overall **energy consumption measurement** can be considered with **continuous integration (CI)**. The method proposed by Drangmeister et al. [7] can be used for monitoring and optimizing the energy consumption of software". The guideline (**GL32**) explained as "Software developers should be concerned about software's energy efficiency in unit and integration test steps." proposed by Drangmeister et al. [7].

SDLC Phase	Aspect Name	No.	Guidelines	Description	Reference
Testing and Verification	Testing with Energy Measurement	GL31	The energy efficiency measurement with continuous integration can help to measure the overall energy consumption of software.	The overall energy consumption measurement can be considered with continuous integration (CI). The method proposed by Drangmeister et al. [7] can be used for monitoring and optimizing the energy consumption of software.	Drangmeister et al. [7]
	Testing with Energy Measurement	GL32	Energy measurement in unit and integration test steps can be used to rate software's energy efficiency.	Software developers should be concerned about software's energy efficiency in unit and integration test steps.	Drangmeister et al. [7]

Table 8: Testing and Verification Guidelines

4.5 Deployment

In the Deployment phase, the software is deployed in its expected running environment. This phase in SDLC also has a great impact on the software's energy efficiency. This study identified several key elements in this phase that are affecting the energy efficiency of software. Besides, this study also gathered crucial guidelines for those elements. Varrette et al. [31] noticed that the choices of the **virtualization frameworks** have great effects on the software's energy efficiency. Besides, Pérez et al. [25] also noticed that the choices of **load balancing algorithms** and the amount of **GPU frequencies** have great effects on the software's energy efficiency. In Table 9, the guidelines for the above elements are provided. The guideline (**GL33**) explained as "**Virtualization frameworks** like Xen, KVM, and VMware ESXi are not optimal for energy-efficient software. Software may use more energy with the overhead of virtualizations." claimed by Varrette et al. [31]. The guideline (**GL34**) explained as "Not all but a specific **load balancing algorithm** helps to reduce the energy consumption of software with higher **GPU frequencies** in an efficient manner." claimed by Pérez et al. [25].

SDLC Phase	Aspect Name	No.	Guidelines	Description	Reference
Deployment	Deployment Environment	GL33	The virtualization frameworks are not optimal choices for reducing the energy consumption of software.	Virtualization frameworks like Xen, KVM, and VMware ESXi are not optimal for energy-efficient software. Software may use more energy with the overhead of virtualizations.	Varrette et al. [31]
	Load Balancing	GL34	A specific load-balancing algorithm can reduce the energy consumption of software.	Not all but a specific load balancing algorithm helps to reduce the energy consumption of software with higher GPU frequencies in an efficient manner.	Pérez et al. [25]

Table 9: Deployment Guidelines

4.6 Maintenance

Software maintenance is the phase where software is evolved. This phase has a great effect to improve the software's energy efficiency. This study identified key elements in this phase that are affecting the energy efficiency of software. Besides, this study also gathered crucial guidelines for the element. Sahin et al. [27] noticed that **refactoring methods** have a great effect on improving the software's energy efficiency. In Table 10, the guideline for the above element is provided. The guideline (**GL35**) explained as "Refactoring has a great impact on software energy efficiency. A **refactoring method** can increase or decrease the energy consumption of software." claimed by Sahin et al. [27].

SDLC Phase	Aspect Name	No.	Guidelines	Description	Reference
Maintenance	Refactoring	GL35	Different refactoring methods can reduce and increase the energy efficiency of software.	Refactoring has a great impact on software energy efficiency. A refactoring method can increase or decrease the energy consumption of software.	Sahin et al. [27]

Table 10: Maintenance Guidelines

5 CONCLUSIONS

This study depicted the key elements affecting software's energy efficiency. Those key elements were categorized in a mind map(**Figure 6**) through software development life cycle (SDLC) phases. The snowball approach helped to get more depth information about a topic. Also, this guided that information can be compared in a backward and forward manner. This study also provided several crucial guidelines(**Table 11**) for these above elements.

The results of this study will help a software professional to check all the elements that are affecting the energy efficiency of software from the mind map (**Figure 6**) and take decisions on each step of software development based on the guidelines provided for these elements (**Table 11**). However, this study highlights the importance of developing a framework for helping software professionals to enable the greenness of software. The framework should assist software professionals on their development choices based on guidelines. In other words, it will provide the recommendations for enabling greenness of software.

REFERENCES

1. Abdulsalam, S., Lakomski, D., Gu, Q., Jin, T., Zong, Z.: Program energy efficiency: The impact of language, compiler and implementation choices. In: International Green Computing Conference. pp. 1–6 (2014). https://doi.org/10.1109/IGCC.2014.7039169
2. Beghoura, M.A., Boubetra, A., Boukerram, A.: Green software requirements andmeasurement: Random decision forests-based software energy consumption profiling. Requir. Eng. **22**(1), 27–40 (Mar 2017). https://doi.org/10.1007/s00766-015-0234-2, https://doi.org/10.1007/s00766-015-0234-2
3. Boukerche, A., Guan, S., Grande, R.E.D.: Sustainable offloading in mobilecloud computing: Algorithmic design and implementation. ACM Comput. Surv. **52**(1) (Feb 2019). https://doi.org/10.1145/3286688, https://doi.org/10.1145/3286688
4. Capra, E., Francalanci, C., Slaughter, S.A.: Is software ”green”? application development environments and energy efficiency in open source applications. Inf. Softw. Technol. **54**(1), 60–71 (Jan 2012). https://doi.org/10.1016/j.infsof.2011.07.005, https://doi.org/10.1016/j.infsof.2011.07.005
5. Couto, M., Maia, D., Saraiva, J.a., Pereira, R.: On energy debt: Managing consumption on evolving software. In: Proceedings of the 3rd International Conference on Technical Debt. p. 62–66. TechDebt ’20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3387906.3388628, https://doi.org/10.1145/3387906.3388628
6. Couto, M., Pereira, R., Ribeiro, F., Rua, R., Saraiva, J.a.: Towards a green rankingfor programming languages. In: Proceedings of the 21st Brazilian Symposium on Programming Languages. SBLP 2017, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3125374.3125382, https://doi.org/10.1145/3125374.3125382
7. Drangmeister, J., Kern, E., Dick, M., Naumann, S., Sparmann, G., Guldner, A.: Greening software with continuous energy efficiency measurement. INFORMATIK 2013–Informatik angepasst an Mensch, Organisation und Umwelt (2013)
8. Erd´elyi, K.: Special factors of development of green software supporting eco sustainability. In: 2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY). pp. 337–340 (Sep 2013). https://doi.org/10.1109/SISY.2013.6662597

9. Feitosa, D., Alders, R., Ampatzoglou, A., Avgeriou, P., Nakagawa, E.Y.: Investigating the effect of design patterns on energy consumption. Journal of Software: Evolution and Process **29**(2), e1851 (2017)
10. Georgiou, S., Kechagia, M., Louridas, P., Spinellis, D.: What are your programming language's energy-delay implications? In: Proceedings of the 15th International Conference on Mining Software Repositories. p. 303–313. MSR '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3196398.3196414, https://doi.org/10.1145/3196398.3196414
11. Georgiou, S., Kechagia, M., Spinellis, D.: Analyzing programming languages' energy consumption: An empirical study. In: Proceedings of the 21st Pan-Hellenic Conference on Informatics. pp. 1–6 (2017)
12. Georgiou, S., Rizou, S., Spinellis, D.: Software development lifecycle for energy efficiency: Techniques and tools. ACM Comput. Surv. **52**(4) (Aug 2019). https://doi.org/10.1145/3337773, https://doi.org/10.1145/3337773
13. Jagroep, E.A., van der Werf, J.M.E., Spauwen, R., Blom, L., van Vliet, R.,Brinkkemper, S.: An energy consumption perspective on software architecture. In: European Conference on Software Architecture. pp. 239–247. Springer (2015)
14. Jin, C., de Supinski, B.R., Abramson, D., Poxon, H., DeRose, L., Dinh, M.N.,Endrei, M., Jessup, E.R.: A survey on software methods to improve the energy efficiency of parallel computing. The International Journal of High Performance Computing Applications **31**(6), 517–549 (2017)
15. Kambadur, M., Kim, M.A.: An experimental survey of energy management acrossthe stack. In: Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications. pp. 329–344 (2014)
16. Kocak, S.A.: Green software development and design for environmental sustainability. In: 11th International Doctoral Symposium an Empirical Software Engineering (IDOESE 2013). Baltimore, Maryland. vol. 9 (2013)
17. Linares-Va´squez, M., Bavota, G., Bernal-Ca´rdenas, C., Oliveto, R., Di Penta,M., Poshyvanyk, D.: Mining energy-greedy api usage patterns in android apps: An empirical study. In: Proceedings of the 11th Working Conference on Mining Software Repositories. p. 2–11. MSR 2014, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2597073.2597085, https://doi.org/10.1145/2597073.2597085
18. Manotas, I., Bird, C., Zhang, R., Shepherd, D., Jaspan, C., Sadowski, C., Pollock,L., Clause, J.: An empirical study of practitioners' perspectives on green software engineering. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). pp. 237–248 (2016). https://doi.org/10.1145/2884781.2884810

19. Memeti, S., Pllana, S., Binotto, A., Ko lodziej, J., Brandic, I.: Using meta-heuristicsand machine learning for software optimization of parallel computing systems: a systematic literature review. Computing **101**(8), 893–936 (2019)
20. Michanan, J., Dewri, R., Rutherford, M.J.: Predicting data structures for energyefficient computing. In: 2015 Sixth International Green and Sustainable Computing Conference (IGSC). pp. 1–8 (2015). https://doi.org/10.1109/IGCC.2015.7393698
21. Misailovic, S., Carbin, M., Achour, S., Qi, Z., Rinard, M.C.: Chisel: Reliability-andaccuracy-aware optimization of approximate computational kernels. ACM Sigplan Notices **49**(10), 309–328 (2014)
22. Pang, C., Hindle, A., Adams, B., Hassan, A.E.: What do programmers knowabout software energy consumption? IEEE Software **33**(3), 83–89 (2016). https://doi.org/10.1109/MS.2015.83
23. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J.: Energy efficiency across programming languages: how do energy, time, and memory relate? In: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering. pp. 256–267 (2017)
24. Pereira, R., Couto, M., Saraiva, J.a., Cunha, J., Fernandes, J.a.P.: The influence of the java collection framework on overall energy consumption. In: Proceedings of the 5th International Workshop on Green and Sustainable Software. p. 15–21. GREENS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2896967.2896968, https://doi.org/ 10.1145/2896967.2896968
25. P´erez, B., Stafford, E., Bosque, J.L., Beivide, R.: Energy efficiency of load balancing for data-parallel applications in heterogeneous systems. J. Supercomput. **73**(1), 330–342 (jan 2017). https://doi.org/10.1007/s11227-016-1864-y, https: //doi.org/10.1007/s11227-016-1864-y
26. Sahin, C., Cayci, F., Guti´errez, I.L.M., Clause, J., Kiamilev, F., Pollock, L., Winbladh, K.: Initial explorations on design pattern energy usage. In: 2012 First International Workshop on Green and Sustainable Software (GREENS). pp. 55–61 (2012). https://doi.org/10.1109/GREENS.2012.6224257
27. Sahin, C., Pollock, L., Clause, J.: How do code refactorings affect energy usage?In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2652524.2652538, https://doi.org/10.1145/2652524.2652538
28. Sarood, O., Langer, A., Kal´e, L., Rountree, B., de Supinski, B.: Optimizing powerallocation to cpu and memory subsystems in overprovisioned hpc systems. In: 2013 IEEE International Conference on Cluster Computing (CLUSTER). pp. 1–8 (2013). https://doi.org/10.1109/CLUSTER.2013.6702684

29. Temesgene, D., Porras, J., Parkkila, J.: Offloading for mobile device performanceimprovement. In: Proceedings of the 52nd Hawaii International Conference on System Sciences (2019)
30. Tsirogiannis, D., Harizopoulos, S., Shah, M.: Analyzing the energy efficiency of adatabase server. pp. 231–242 (06 2010). https://doi.org/10.1145/1807167.1807194
31. Varrette, S., Guzek, M., Plugaru, V., Besseron, X., Bouvry, P.: Hpc performanceand energy-efficiency of xen, kvm and vmware hypervisors. In: 2013 25th International Symposium on Computer Architecture and High Performance Computing. pp. 89–96 (2013). https://doi.org/10.1109/SBAC-PAD.2013.18
32. Vassiliadis, V., Riehme, J., Deussen, J., Parasyris, K., Antonopoulos, C.D., Bellas,N., Lalis, S., Naumann, U.: Towards automatic significance analysis for approximate computing. In: 2016 IEEE/ACM International Symposium on Code Generation and Optimization (CGO). pp. 182–193. IEEE (2016)
33. Venters, C.C., Capilla, R., Betz, S., Penzenstadler, B., Crick, T., Crouch, S., Nakagawa, E.Y., Becker, C., Carrillo, C.: Software sustainability: Research and practice from a software architecture viewpoint. Journal of Systems and Software **138**, 174–188 (2018). https://doi.org/https://doi.org/10.1016/j.jss.2017.12.026, https://www.sciencedirect.com/science/article/pii/S0164121217303072
34. Venters, C.C., Kocak, S.A., Betz, S., Brooks, I., Capilla, R., Chitchyan, R., Duboc,L., Heldal, R., Moreira, A., Oyedeji, S., Penzenstadler, B., Porras, J., Seyff, N.: Software sustainability: Beyond the tower of babel. In: 2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS).
pp. 3–4 (2021). https://doi.org/10.1109/BoKSS52540.2021.00009
35. Grosskop, K., Visser, J.: Chapter 5 - energy efficiency optimization of application software. In: Hurson, A. (ed.) Green and Sustainable Computing: Part II, Advances in Computers, vol. 88, pp. 199–241. Elsevier (2013).https://doi.org/https://doi.org/10.1016/B978-0-12-407725-6.00005-8, https://www.sciencedirect.com/science/article/pii/B9780124077256000058
36. Marantos, C., Salapas, K., Papadopoulos, L., Soudris, D.: A flexible tool for estimating applications performance and energy consumption through static analysis. SN Computer Science 2(1), 1–11 (2021)
37. Salam, M., Khan, S.U.: Developing green and sustainable software: Success factors for vendors. In: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS). pp. 1059–1062 (2016).https://doi.org/10.1109/ICSESS.2016.7883248

APPENDIX 1. Guidelines

SDLC Phase	Aspect Name	No.	Guidelines	Description	Reference
Requirement Engineering	Functional Requirements	GL01	Identify energy requirements by correlating them with functional requirements	Software developers should specify the functional and non-functional requirements. Non-functional requirements are attached to the functional requirements. Mark energy requirements from non-functional requirements.	Pang et al. [22]
	Software Quality Model	GL02	Use energy efficiency as a characteristic in software quality model	Software developers should use green efficiency attributes in software quality model to facilitate the energy efficiency techniques in the software development stage.	Beghoura et al. [2]
	Requirement Elicitation Strategies	GL03	Use a requirement elicitation strategy for identifying a task's energy usage acceptance level	Energy-related (green) requirements are stated usually in other words. Software developers should use an elicitation strategy to identify a task's energy usage acceptance level. For instance (e.g., mobile location applications should be able to run from available charging sources when driving a car or bike). So, the energy requirement is visible behind the stated requirement. Also, energy requirements need to define with their acceptance levels.	Manotas et al. [18]
Software Design	Software Architecture	GL04	Architectural Description (AD) must have to alter with energy consumption perspectives.	Software architects should modify AD (different views and others) to enhance greenness of a software. The energy consumption perspectives are. 1.Identify requirements that ensure less energy consumption -Context View: How can the software achieve green goals? -Functional View: How much energy can a function consume? -Information View: What can we do to optimize information flow for reducing energy consumption? -Concurrency View: How we can arrange functional	Jagroep et al. [13]

				elements to be attached to processes and how the hardware will execute all these processes? -Development View: Can we choose green algorithms for the software? -Operational View: How run time issues can be solved to reduce energy consumption? -Deployment View: Which hardware aligned with what processes? 2. Make energy profile: Determine what level of energy the software consume.	
				3. Check energy profile with requirements: Evaluate the requirements with the energy profile for further adjustments. 4. Determine the adjustments: Make some adjustments based on energy hotspot elements. 5. Check the adjustments: After the adjustments, check if the adjustments have positive effects on energy consumption.	
	Software Architecture	GL05	The software quality metrics can be used to evaluate the software architecture's greenness.	Usually, software quality metrics are used to estimate the quality of a system. For example, Bad coding practices will make it more complex to evolve software. Bad coding practices include redundant APIs. So, there is an issue with the architectural functional view.	Venters et al. [33]
	Software Architecture	GL06	Energy debts should be mitigated to ensure green software.	Energy debts are raised by the energy smells. Energy smells are energy-greedy or inefficient coding practices. It is different from technical debt. The energy debt increases as the number of times a specific release performs tasks.	Couto et al. [5]
	Design Patterns	GL07	Design patterns have a variety of impacts on software energy consumption. So, software developers should choose simple design patterns for developing software or	A design pattern has impacts on software energy efficiency. The impacts of design patterns are not constant. Avoid patterns that have a complex way of implementation. For example, the Decorator pattern is not an energy-efficient one. Decorator pattern creates complex objects except for inheritance.	Sahin et al. [26]

			system.		
	Design Patterns	GL08	Nonpattern is better than the use of any design pattern. In complex cases of implementation, design patterns can be an energy-efficient choice.	Design patterns have similar or higher energy consumption levels in most cases. However, it is a good idea to use design patterns when pattern-related methods with a higher number of method invocations are required for implementing complex tasks	Feitosa et al. [9]
	Design Patterns	GL09	Android mobile applications should not use Model-View-Controller (MVC) pattern when it has many **views.**	It is a pretty energy-consuming task for android applications to navigate **views.** So, using Model-View-Controller (MVC) pattern allows refreshing each view when need load contents. Which is an energy-expensive task. Software developers must avoid it unless they have no alternatives.	Linares-Vásquez et al. [17]
	Design Patterns	GL10	Design principles and energy efficiency have a tradeoff. Software developers should decide carefully on which one they are willing to focus on.	Data hiding in an android application is energy consuming task. So, Design principles relevant to data orchestration can decrease the energy efficiency of an android application.	Linares-Vásquez et al. [17]
Software Development	Development Environment	GL11	The use of a development environment (frameworks, libraries) can reduce the energy consumption of small and medium size applications. But not for large applications.	The use of development environment (frameworks, libraries) for small and medium size applications can reduce the energy consumption. For large applications, the use of an application development environment (frameworks, libraries) is not a good choice for energy efficiency. Because the development environment is used for optimizing development efficiency. There is a trade-off between development efficiency and energy efficiency (based on application size).	Capra et al. [4]
	Development Environment	GL12	Developing new functions from scratch is not energy efficient.	If software developers develop a new function from scratch, then it is quite difficult to ensure the energy efficiency of that.	Capra et al. [4]
	Development Environment	GL13	Integrating modules can increase software's overall energy consumption.	Large applications are created by integrating modules. So, that can easily affect the overall software energy consumption.	Capra et al. [4]

	Development Environment	GL14	Energy efficient choice is choosing native language written libraries.	Libraries that are written in native language have greater energy efficiency over other libraries.	Georgiou et al. [10]
	Programming language	GL15	Using compiler optimization flags can have a great effect on software energy consumption.	Software professionals must have to choose a compile-time performance optimization flag for ensuring energy efficiency.	Georgiou et al. [10]
	Programming language	GL16	The malloc is the most energy-efficient memory allocation choice.	If we consider, the four memory allocation choices (malloc, new, array, and vector). The malloc is the best for energy-efficient choice.	Abdulsalam et al. [1]
	Programming language	GL17	The compiled programming languages are better than semi-compiled programming languages in terms of energy efficiency. The semi-compiled programming languages are better than interpreted programming languages in terms of energy efficiency.	C, C#, and JavaScript are the best programming languages for energy-efficient software development.	Georgiou et al. [10], Pereira et al. [23] & Georgiou et al. [11]
	Parallel Computing	GL18	Effective parallelization and compiler optimizations can reduce software energy consumption.	Utilizing and effectively using CPU cores and threads can reduce the energy consumption of software.	Kambadur et al. [15]
	Parallel Computing	GL19	Multithreaded applications need to be used with an equal number of cores and threads.	An equal number of cores and threads can increase the performance and energy efficiency of a multithreaded application.	Jin et al. [14]
	Parallel Computing	GL20	Idle and sleep states can reduce the energy consumption of the processor. That also increases software energy efficiency.	CPU Idle and sleep states help to reduce the energy consumption of software. In an idle time of software, the CPU idle and sleep states help to reduce energy consumption.	Kambadur et al. [15]
	Parallel Computing	GL21	Overclocking can minorly improve software energy efficiency.	Overclocking can reduce the runtime of software execution but it has fewer effects on software energy efficiency.	Kambadur et al. [15]
	Approximate	GL22	There is a trade-	By defining specific accuracy	Vassiliadis

	Computing		off between energy efficiency and accuracy specification.	specification in approximate computing, the software can achieve a significant amount of energy efficiency.	et al. [32]
	Approximate Computing	GL23	Optimization constraints must satisfy the accuracy specification.	In approximate computing, we need to define optimization constraints properly that leads expected accuracy specification for energy efficiency.	Misailovic et al. [21]
	Approximate Computing	GL24	An optimization algorithm needs to use for obtaining energy savings.	If we use an optimization algorithm in approximate computing then we can achieve energy efficiency in computational works.	Misailovic et al. [21]
	Cloud Computing	GL25	The energy efficiency in offloading depends on remote (cloud resources).	In offloading, if we consider energy efficiency then there is an effect in task efficiency. Always there is a trade-off between the energy efficiency of offloading and task efficiency.	Boukerche et al. [3]
	Cloud Computing	GL26	Offloading in the nearest location (closest to the end device) has a positive impact on mobile device performance (Also it has energy efficiency)	Mobile devices have fewer resources. So, offloading in the nearest location allows energy efficiency in computation.	Temesgene et al. [29]
	Data Structure	GL27	The choice of data structure has an enormous effect on the energy consumption of software (computing).	To ensure green software (with energy-efficient computing) we have to choose the right data structure for the right task. C5 data structures can be selected based on tasks and energy efficiency.	Michanan et al. [20]
	Data Structure	GL28	The Java Collection Framework (Sets, Lists, and Maps) has also a great impact on the energy consumption of software.	The optimized Java Collection Framework (Sets, Lists, and Maps) can reduce energy consumption compared to the original implementation.	Pereira et al. [24]
	Database	GL29	NoSQL Database is a good choice for ensuring energy-efficient software development.	Relational Databases are less energy efficient compared to NoSQL. If there is no special requirement on Database then a choice of NoSQL will be a good decision.	Georgiou et al. [12] & Tsirogiannis et al. [30]
	High-Performance Computing (HPC)	GL30	Different mechanisms can be used to ensure energy efficiency in HPC applications.	Organizing HPC clusters with fewer nodes, utilizing DVFS low voltage and Power Capping can reduce the energy consumption in HPC applications	Sarood et al. [28]

Testing and Verification	Testing with Energy Measurement	GL31	The energy efficiency measurement with continuous integration can help to measure the overall energy consumption of software.	The overall energy consumption measurement can be considered with continuous integration (CI). The method proposed by Drangmeister et al. [7] can be used for monitoring and optimizing the energy consumption of software.	Drangmeister et al. [7]
	Testing with Energy Measurement	GL32	Energy measurement in unit and integration test steps can be used to rate software's energy efficiency.	Software developers should be concerned about software's energy efficiency in unit and integration test steps.	Drangmeister et al. [7]
Deployment	Deployment Environment	GL33	The virtualization frameworks are not optimal choices for reducing the energy consumption of software.	Virtualization frameworks like Xen, KVM, and VMware ESXi are not optimal for energy-efficient software. Software may use more energy with the overhead of virtualizations.	Varrette et al. [31]
	Load Balancing	GL34	A specific load-balancing algorithm can reduce the energy consumption of software.	Not all but a specific load balancing algorithm helps to reduce the energy consumption of software with higher GPU frequencies in an efficient manner.	Pérez et al. [25]
Maintenance	Refactoring	GL35	Different refactoring methods can reduce and increase the energy efficiency of software.	Refactoring has a great impact on software energy efficiency. A refactoring method can increase or decrease the energy consumption of software.	Sahin et al. [27]

Table 11: Guidelines