International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Comparative Analysis of Energy Efficiency in Desktop Web Browsers: Towards Sustainable Software Applications

P.S. Felix*[1], Dr. M. Mohankumar[2]

Submitted:13/03/2024 **Revised**: 28/04/2024 **Accepted**: 05/05/2024

Abstract: The pursuit of sustainability necessitates a multifaceted approach to mitigate various forms of pollution, particularly carbon emissions, prevalent in electricity generation and the usage of electrical and electronic devices. Electricity generation, primarily reliant on fossil fuels like coal, natural gas, and oil, contributes significantly to global greenhouse gas emissions, prompting exploration into cleaner, renewable energy sources such as solar, wind, hydro, and biomass. Similarly, electrical and electronic devices, crucial in modern lifestyles, exhibit varying levels of energy efficiency, denoted by star ratings, with higher ratings indicating lower power consumption. Beyond hardware considerations, software applications also play a pivotal role in energy consumption, with ongoing research focusing on enhancing their efficiency. This article compares the energy consumption of different desktop web browsers during routine internet usage tasks, aiming to identify the most energy-efficient option. To achieve this, the Green IT Hexagon methodology is applied, offering a comprehensive framework for evaluating software energy usage. This research article encompasses defining the scope, identifying usage scenarios, selecting representative websites, setting up a standardized test environment, measuring power consumption using a Power Reading Unit (PRU), and conducting detailed data analysis and comparison. This experiment underscores the importance of considering software energy efficiency alongside hardware considerations in the pursuit of sustainability, offering insights into optimizing energy usage in the digital realm.

Keywords: *green software engineering, green computing, sustainable software, green it, green environment*

1. Introduction

Sustainability is a global goal that requires reducing various types of pollution that harm the environment and human health. One of the main sources of pollution is carbon emission, which causes global warming and climate change. Carbon emission occurs when fossil fuels are burned to generate electricity or power vehicles and machines. Therefore, it is important to identify and address the major contributors to carbon emission and find ways to reduce them. One of the major contributors to carbon emission is electricity generation, which accounts for about 25% of global greenhouse gas emissions [14].

Electricity generation depends on the type and amount of fuel used, as well as the efficiency and capacity of the power plants. Coal, natural gas, and oil are the most common fossil fuels used for electricity generation, and they emit high levels of carbon dioxide and other pollutants. Renewable energy sources, such as solar, wind, hydro, and biomass, are cleaner and more sustainable alternatives, but they face challenges such as cost, availability, and reliability.

Another major contributor to carbon emission is the use of electrical and electronic devices, such as lights, fans, air conditioners, refrigerators, etc. These devices consume electricity and emit heat, which adds to the global warming effect [18]. To reduce electricity consumption and carbon emission, these devices are given star ratings to indicate their energy efficiency. The higher the star rating, the lower the electricity consumption and carbon emission. The public is encouraged to buy and use higher-rated devices for a more sustainable world.

The software plays a more significant role than hardware in determining the energy efficiency of electronic devices. Software applications are programs that run on devices and perform various tasks, such as browsing the internet, playing games, editing documents, and sending emails. Software applications vary in their energy consumption depending on their design, functionality, and performance. Some software applications use more energy than others because they require more processing power, memory, or network bandwidth [13].

Ongoing research is focused on enhancing the energy efficiency of computers and developing systems and algorithms related to energy-efficient technologies. “Green IT” [16] encompasses all IT solutions that conserve energy at multiple usage levels, including hardware, software, and services. The hardware should consist of energy-saving desktop PCs and thin-client structures. In this article, we compare the energy consumption of different desktop web browsers, which are software applications that allow users to access the internet. The internet is widely used for various purposes, such as education, entertainment, communication,

[1] *Karpagam Academy of Higher Education, Pollachi Main Road, Coimbatore, Tamil Nadu, India*
ORCID ID: 0000-0002-0592-0426
[2] *Karpagam Academy of Higher Education, Pollachi Main Road, Coimbatore, Tamil Nadu, India*
ORCID ID: 0000-0002-7021-2725
** Corresponding Author Email: psfelix@gmail.com*

and information.

Desktop web browsers are the main interface between users and the internet, and they differ in their features, speed, security, and compatibility. We collect data on the power consumption of the most popular web desktop browsers, such as Google Chrome, Microsoft Edge, and Internet Explorer, while performing search engine tasks for daily activities. We aim to identify which web browser is the most energy-efficient and environmentally friendly.

2. Literature Review

Energy-efficient software development is key to sustainability. This process extends beyond coding to include all phases from planning to maintenance [3]. The goal is to create green software by applying green principles throughout the process, including software testing. This approach allows for efficient collection of energy consumption data and optimization of software operations. The green concept is crucial in today's efficient software development, which involves various phases and activities. Software metrics are used to analyze and improve code. High-end games and applications requiring 3D and video editing necessitate a powerful Graphic Processing Unit (GPU) [4]. Monitoring GPU performance and power usage can identify energy-intensive areas in scripts, particularly for video rendering and game development.

This study conducts a systematic mapping of sustainability and software architecture research to assess the current state of the art and identify future research needs [5]. The results reveal a disproportionate focus on specific, mostly technical aspects of sustainability in existing works. This focus neglects the holistic perspective necessary to address sustainability, a multi-faceted concern.

The research aims to address this gap. Software architecture quality significantly impacts sustainability. Despite their potential, scenario-based evaluation methods are underutilized and not integrated with architecture-level metrics, limiting their effectiveness [6]. Current literature reviews lack critical reflection on the applicability of these methods for sustainability evaluation. This study aims to bridge this gap by reviewing scenario-based methods for sustainability and categorizing over 40 architecture-level metrics.

The treatment of sustainability as a software quality property and define a software sustainability assessment [7] method that helps make sustainability-driven design decisions. The method essentially relies on the definition of so-called 'decision maps', i.e. views aimed at framing the architecture design concerns around the four sustainability dimensions mentioned above-technical, economic, social and environmental sustainability. This research delves into sustainability in software engineering, with a focus on South Asian professionals. It highlights the industry's neglect of sustainability, despite its academic importance, leading to misunderstandings [8]. The study, involving a survey of 221 practitioners, found that while sustainability is valued, its integration in software development is unclear. The results underscore the need for explicit guidelines and heightened awareness for sustainable software development.

Sustainability in software design is a significant research challenge. Current research is focused on understanding sustainability and integrating it into the software development lifecycle, but there are few effective guidelines for designers. This paper introduces a Sustainability Design Catalogue (SSDC) [9] based on past and current research, aimed at helping developers elicit sustainability requirements and measure software sustainability. The SSDC, developed through analysis of four case studies and a proposed pilot framework, exemplifies the application and quantification of sustainability. Sustainability is a key topic in software and requirements engineering, but its identification in agile software development is less understood. This research aims to incorporate sustainability knowledge into agile frameworks like Scrum. Two case studies applied the Sustainability Awareness Framework [10], identifying over 20 potential sustainability effects for each system. The findings suggest that analyzing product backlogs for sustainability can aid early identification of effects, demonstrating the framework's practical utility.

Sustainability is gaining importance in software engineering, particularly in requirements engineering. Despite several approaches to manage sustainability impacts, there's a knowledge gap in identifying these effects in agile software development. This research aims to incorporate sustainability into Scrum, leading to the development of a Sustainability-Aware Scrum Framework [11]. The work offers recommendations for extending Scrum towards sustainability and presents a prototype tool for tracking sustainability impacts. Sustainable software engineering (SSE) [12] research aims to meet current needs without compromising future capacity, but its pillars need refinement. A scoping review and meta-synthesis of SSE research were conducted to improve existing models. The study found a focus on ecological sustainability and product sustainability, with sustainability being stratified and multisystemic. The findings call for more empirical evaluations and a multisystemic, stratified conceptualization of sustainability.

3. Methodology

The Green IT Hexagon methodology serves as a comprehensive approach for the assessment of green software. This methodology is designed with the primary objective of enhancing Software Energy Usage, thereby promoting more sustainable practices in the field of Information Technology. The methodology operates on the principle that the metrics for energy-efficient software are

contingent on the useful work performed by the software. Given the complexity of modern software, which comprises numerous modules each serving a unique purpose, the methodology acknowledges that there may be more than one applicable metric.

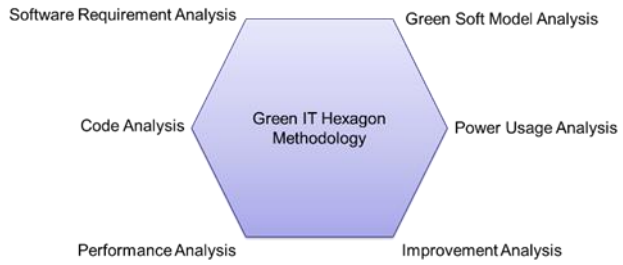

Fig. 1. Green IT Hexagon Methodology diagram

In the Green IT Hexagon methodology, these software parts can be evaluated either individually or in combination. This flexibility allows for a more nuanced understanding of the software's energy usage patterns and efficiency. However, for an accurate comparison of different software, it is recommended that the measured modules be as similar as possible. Figure 1 shows the Green IT Hexagon Methodology diagram.

Fundamentally, the methodology introduces a universal metric for evaluating software energy efficiency. This clearly defined metric acts as a benchmark, enabling consistent assessment of energy usage and efficiency across diverse software systems. Such an approach fosters a standardized and impartial appraisal of software energy consumption, ultimately advancing the overarching objective of promoting energy efficiency and sustainability in software development. Commonly used approach to measure energy efficiency as below

$$Energy\ Efficiency = \frac{Useful\ Work\ Done}{Used\ Energy} \quad (1)$$

4. Existing Work

Energy monitoring policies for observing a program's energy usage can be categorized into two types: external and internal evaluators. External energy monitors employ tools like voltmeters and ammeters to evaluate the system as a whole. However, their ability to monitor individual programs is limited as they lack the granularity to identify energy usage at the component level. On the other hand, internal evaluators are integrated within the system like Power Reading Unit (PRU) for power monitoring. They measure energy registers, process wakeups, and CPU state transitions to provide a more detailed view of a program's energy consumption.

4.1. Energy Optimization in Software Development

In our previous article [17], we conducted an experiment to compare the energy efficiency of two sorting algorithms: Bubble Sort and Quicksort. Sorting algorithms are methods of arranging data in a specific order, such as ascending or descending. Sorting algorithms are widely used in computer science and engineering, as they can improve the performance and functionality of various applications and systems. However, sorting algorithms also consume energy, which can have environmental and economic impacts. Therefore, it is important to identify which sorting algorithm is the most energy-efficient for different types of data. We chose Bubble Sort and Quicksort as the two sorting algorithms to test, as they are among the most common and well-known sorting algorithms. Bubble Sort is a simple algorithm that works by repeatedly comparing and swapping adjacent elements in an array until the array is sorted. Quicksort is a more complex algorithm that works by dividing an array into two subarrays based on a pivot element, and then recursively sorting the subarrays. Both algorithms have different advantages and disadvantages, depending on the size and distribution of the data.

We measured the energy consumption of both algorithms by using a Power Reading Unit (PRU) [15], which is a device that can accurately measure the power consumption of the computer that run the algorithms. We used the same array of 100 thousand numerical data as the input for both algorithms, and we recorded the start time, end time, and time taken by each algorithm to sort the array.

The results of this experiment that compared the power consumption of two sorting algorithms: Quick Sort and Bubble Sort. The results showed that Quick Sort was faster and more energy-efficient than Bubble Sort, as it took only 0.088 seconds and consumed 0 Wh, while Bubble Sort took 1 minute and 17.522 seconds and consumed 1.18 Wh. The difference is because Quicksort has a lower time complexity than Bubble Sort, which means it performs fewer operations and comparisons to sort the array. Time complexity is a measure of how the running time of an algorithm changes with the size of the input. Quicksort has an average time complexity of O(n log n), which means it grows interatomically with the input size. Bubble Sort has a worst-case time complexity of $O(n^2)$, which means it grows quadratically with the input size. Based on our experiment, we concluded that Quicksort is a more energy-efficient sorting algorithm than Bubble Sort when the input data is large and random. However, this may not be the case for other types of data, such as data that is already sorted or nearly sorted.

4.2. Energy Model

One way to measure the energy consumption of a program is to use the equation proposed by [3]:

$$E_{\text{application}} = E_{\text{active}} + E_{\text{wait}} + E_{\text{idle}} \quad (2)$$

This equation shows that the total energy consumed by a program in an application model consists of three

components: E_{active}, E_{wait}, and E_{idle}. E_{active} is the energy consumed by the program when it is executing its tasks on the system. For example, if the program is sorting an array of data, E_{active} is the energy used to perform the sorting algorithm. E_{wait} is the energy consumed by the program when it is waiting for other components to finish their tasks or to provide input or output. For example, if the program is reading data from a file, E_{wait} is the energy used to wait for the file to be opened and read. E_{idle} is the energy consumed by the system when it is not doing any work for the program. For example, if the program is finished and the system is still on, E_{idle} is the energy used to keep the system running. By using this equation, we can analyse the energy efficiency of different programs and identify the factors that affect their energy consumption.

5. Approach

To conduct this experiment on the power consumption of desktop web browsers, we followed the systematic approach detailed below. Figure 2 shows the "Experimental Approach Steps"

Fig. 2. Experimental Approach Steps

(A) *Define the Scope*: The first step of this experiment is to define the scope of the study by selecting the web browsers that will be measured for their power consumption. The selection criteria should consider the most widely used browsers in the market, their popularity, and features. Additionally, it's advisable to include a browser that may not be commonly used but is still available in the default Windows operating system. This inclusion allows for an exploration of earlier browser versions and their design for energy efficiency.

(B) *Identify Usage Scenarios*: The next step is to identify the usage scenarios that will be used to test the power consumption of the web browsers. The usage scenarios should reflect the typical and realistic web browsing activities that users perform on a daily basis.

(C) *Select Websites*: The third step is to select the websites that will be used for the image searching scenario. The websites should be representative of the most visited and the most engaging websites on the internet. They should also be compatible and consistent with all the web browsers that are tested.

(D) *Set Up Test Environment*: The fourth stage involves establishing the test environment while maintaining uniform testing conditions across all web browsers. This encompasses both hardware and software components of the test system. The hardware includes computer components utilized for running the web browsers, along with the Power Reading Unit (PRU) employed for power consumption measurement. Meanwhile, the software component comprises the operating system and specific versions of web browsers used for testing purposes. It's crucial to ensure consistency and control in both hardware and software setups, as any variations could potentially impact the experiment's outcomes.

(E) *Measure Power Consumption*: The fifth step is to measure the power consumption of the web browsers during the execution of the image searching scenario as per the energy efficiency methodology. This is done by using the PRU measurement module, which is a hardware tool that can accurately measure the energy consumption of the chips at hardware runtime. The PRU measurement module provides insights into the energy usage patterns of the web browsers and the algorithms that they use. By measuring the power consumption in real-time, the experiment captures precise and reliable data on the energy efficiency of the web browsers.

(F) *Analyse and Compare*: The final step is to analyse the data and compare the power consumption of the web browsers. This is done by using statistical methods and graphical tools to evaluate the results and draw conclusions. The analysis considers the different aspects and factors that could affect the power consumption of the web browsers, such as the user actions, the web content, the browser features, and the algorithm performance. The comparison should highlight the strengths and weaknesses of each web browser and identify the most energy-efficient web browser under the image searching scenario.

6. Experimental Procedure

This experimental procedure aimed to investigate and compare the power consumption of popular web browsers during specific tasks, shedding light on the potential impact of browser usage on overall system power consumption. In this experimental procedure the set of steps and instructions are followed as per the approach defined above to conduct a scientific experiment. Details are as below:

(1) *Study Range*: This investigation selected the following internet browsers: Google Chrome, Microsoft Edge, and Safari. According to the 2024

report from Similarweb LTD [2], these are among the most popular web browsers globally, and their unique features and functionalities could influence their energy efficiency.

(2) *Scenario of Use*: The primary usage scenario for this study was determined to be image searching on a search engine. This involves commonly performed activities such as searching for images, scrolling through results for a minute to load more images related to the keyword "Flower", and other related web browsing activities that require user interaction, network communication, and graphic rendering. The aim is to replicate user behaviour and web content as accurately as possible in each browser.

(3) *Testing Environment*: The specifications of the computer, including the processor, RAM, and operating system details, were recorded to provide context for the experiment. The computer configuration is as follows:

The system operates on a 64-bit Windows 10 Pro operating system (Version 22H2, OS Build 19045.3930) and is powered by an Intel Core i3-3220 CPU with a clock speed of 3.30 GHz and 8 GB of RAM. The CPU is a dual-core processor that supports hyper-threading and has a 3 MB cache. The RAM is of the DDR3 type and operates at a frequency of 1600 MHz. The operating system is the most recent version of Windows 10, offering enhanced security, performance, and features.

(4) *Test Website*: The website www.google.com was chosen for testing as it is ranked number 1 on Similarweb LTD's website as of February 2024 [1]. After launching all three browsers, the www.google.com website should be opened. On this website, images of flowers should be searched for and the browser should be scrolled for a minute.

(5) *Energy Consumption*: Data A Power Reading Unit (PRU) [15] integrated into the experimental computer was used to measure energy consumption during the execution process. Before execution, the computer's idle time energy consumption was recorded to establish a baseline according to the defined energy model.

(6) *Data Analysis and Comparison*: During this phase, we analysed the power consumption data collected during the experiment, which was captured using the PRU-integrated computer. The data was converted into a visual representation and tabular column data. The results were derived from the various data collected.

7. Results

The outcomes of the experiment are transformed into a comprehensive visual format. This includes the creation of detailed charts and tables that provide a clear and in-depth understanding of the data. These visual representations allow for easier interpretation and comparison of the collected data, thereby offering a more thorough analysis of the experiment's results. The charts highlight key trends and patterns, while the tables present precise numerical values for a more exact understanding. This detailed presentation of results aids in a more comprehensive evaluation of the experiment.

7.1. Amperes Comparison

Figure 3 - The graph of Current Amperes Consumption demonstrates the power usage, quantified in Amperes (Amperes), by utilizing the formula for Amperes as shown below:

$$Amps = \frac{Volts}{Watts} \quad (3)$$

The values are derived for three distinct web browsers: Microsoft Edge, Google Chrome, and Internet Explorer. It is observed that the Amperes increase when a browser is opened, compared to the idle state.

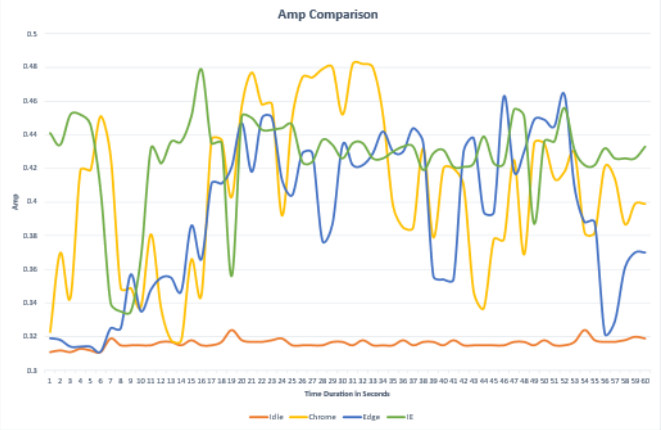

Fig. 3. Current Amperes consumption graph

For **Microsoft Edge**, the power consumption starts at a minimal level and gradually increases as images are searched and scrolled through. Once scrolling stops, the Amperes usage decreases.

Google Chrome also begins with minimal power consumption, but it experiences a quick surge in the initial seconds before dropping as the website loads. As image searching and scrolling commence, the Amperes usage increases again, fluctuating until scrolling ceases. After scrolling stops, the Amperes usage drops slightly but remains higher than that of Microsoft Edge when the browser is idle.

Internet Explorer requires a high Amperes level even to start the browser, similar to the pattern observed in Google Chrome. However, as the website loads and image searching begins, there is an immediate spike in power

consumption, which remains constant until scrolling stops. Notably, the Amperes usage at idle for Internet Explorer is higher than the other two browsers.

Table 1 - Current Amperes consumption table below provides a reference for the power consumption data for the first 20 seconds of the experiment.

Table 1. Current Amperes consumption table

Seconds	*Idle*	*Chrome*	*Edge IE*
1	0.311	0.323	0.3190.441
2	0.312	0.370	0.3180.434
3	0.311	0.343	0.3140.452
4	0.313	0.419	0.3140.452
5	0.312	0.419	0.3140.446
6	0.311	0.451	0.3110.407
7	0.319	0.427	0.3250.340
8	0.315	0.349	0.3250.335
9	0.315	0.349	0.3570.335
10	0.315	0.338	0.3350.367
11	0.315	0.381	0.3480.432
12	0.317	0.337	0.3550.423
13	0.317	0.318	0.3550.436
14	0.315	0.318	0.3470.436
15	0.318	0.366	0.3860.451
16	0.315	0.345	0.3660.479
17	0.315	0.437	0.4110.435
18	0.317	0.437	0.4110.435
19	0.324	0.403	0.4210.356
20	0.318	0.457	0.4470.451

7.2. Watts Comparison

Similar to the Amperes data Figure 4 Watts consumption graph depicts the power usage in watts during the use of three different browsers (Microsoft Edge, Google Chrome, and Internet Explorer) to search for images related to the keyword "Flower", as per the defined experimental procedure.

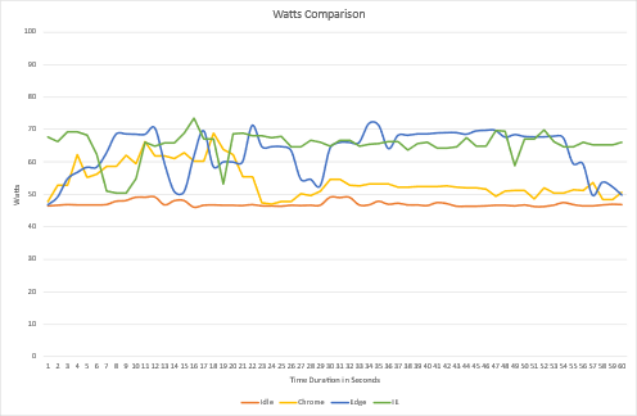

Fig. 4. Watts consumption graph

The power consumption in Watts (W) of the experimental computer device can be calculated using the formula:

$$P = VI \quad (4)$$

Where:

- P presents the power consumption in Watts (W)
- V is the voltage in Volts (V)
- I is the current in Amperes (A)

The PRU device utilizes this formula, enabling us to determine the power usage in Watts during the experiment for each browser. The results are presented in Table 2 below.

In this data, both **Microsoft Edge** and **Google Chrome** start their power consumption near the idle time watts. Google Chrome gradually increases its power usage, which after reaching a peak, drops down and maintains a level slightly higher than the idle time data. On the other hand, Microsoft Edge gradually increases its power consumption, fluctuates for a while, and then maintains a similar level until the scrolling is stopped. After stopping the scrolling, the power consumption drops down to a level near Google Chrome's power usage.

Internet Explorer starts with a high-power consumption, drops immediately, and then increases again once the search starts. It maintains a stable power consumption even after the scrolling is stopped, which is higher than the other two browsers.

Table 2 - Watts consumption table below provides a reference for the power consumption in watts for the first 20 seconds of the experiment.

Table 2. Watts consumption table

Seconds	*Idle*	*Chrome*	*Edge*	*IE*
1	46.5	47.9	46.9	67.7
2	46.7	52.9	49.3	66.2
3	46.9	52.9	54.8	69.2
4	46.8	62.2	56.8	69.2
5	46.8	55.3	58.4	68.3
6	46.8	56.3	58.4	62.3
7	46.9	58.6	62.7	51.1
8	47.9	58.6	68.5	50.5
9	48.1	62.0	68.6	50.5
10	49.1	59.4	68.5	54.9
11	49.1	66.3	68.5	66.0
12	49.2	61.8	70.4	64.8
13	46.8	61.8	59.4	65.9
14	48.1	61.1	50.9	65.9
15	48.1	62.8	50.9	68.9
16	46.1	60.3	61.7	73.5
17	46.7	60.3	69.6	67.0
18	46.8	68.8	58.6	67
19	46.7	63.8	60	53.3
20	46.7	62.3	60	68.7

7.3. Watt Hour Comparison

To determine the energy usage (measured in watt-hours or Wh) of an application was calculated using the following equation:

$$E = P \times t \quad (5)$$

In this above equation:

- E stands for the energy usage, which is measured in watt-hours (Wh),
- P is the power, which is measured in watts (W),
- t is the time, which is measured in hours.

The final key data point, Watt Hour, is compared in Figure 5 Watt hour comparison graph above. It clearly illustrates that upon the initiation of the application, there is an increase in power consumption compared to the idle time, which was logged at 81.173 WH. Over the course of a one-minute experiment, the data from the three browsers indicates that Google Chrome consumes the least power, followed by Microsoft Edge, with Internet Explorer consuming the most.

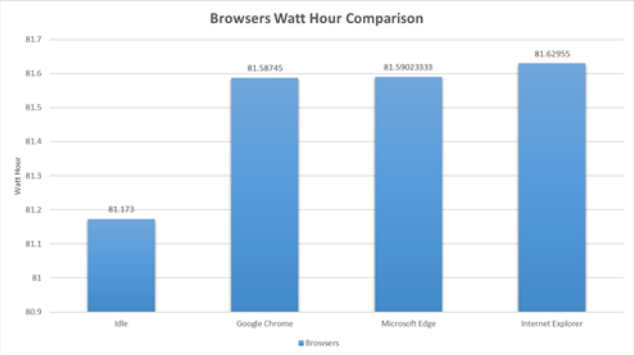

Fig. 5. Watt hour comparison graph

Table 3 - Watt Hour consumption table provided below offers a detailed reference for the consumption of Watt Hours for the first 20 seconds of the experiment.

Table 3. Watt Hour consumption table

Browsers	***Current Consumed (Wh)*** *Watts x Hrs*
Idle	81.173
Google Chrome	81.587
Microsoft Edge	81.590
Internet Explorer	81.629

The results of the experiment reveal distinct patterns in power consumption among the three web browsers: Microsoft Edge, Google Chrome, and Internet Explorer. When examining the Amperes usage, it's evident that all browsers exhibit an increase in power consumption when active compared to their idle states. Microsoft Edge shows a gradual rise in Amperes as images are searched and scrolled through, while Google Chrome experiences a quick surge followed by fluctuation until scrolling ceases. In contrast, Internet Explorer requires high Amperes even at startup, with immediate spikes upon searching and scrolling. Similarly, when analysing Watts consumption, Microsoft Edge and Google Chrome start near idle power but maintain different consumption patterns throughout usage. Internet Explorer, however, starts with high consumption, drops briefly, then stabilizes at a level higher than the other browsers. The comparison of Watt Hour usage further confirms Google Chrome as the most energy-efficient, followed by Microsoft Edge, and with Internet Explorer consuming the most power over a one-minute experiment.

8. Conclusion and Future Work

This study aimed to evaluate the energy efficiency of three prominent desktop web browsers—Microsoft Edge, Google Chrome, and the now-discontinued Internet Explorer—specifically for image searching tasks. Utilizing a methodology focused on Power Usage Analysis, we measured power consumption in terms of Amperes, Watts, and Watt-Hours using the Power Reading Unit (PRU). The findings revealed that both Microsoft Edge and Google Chrome exhibited relatively lower power consumption, with Google Chrome emerging as the most energy-efficient among the trio. In contrast, Internet Explorer consistently demonstrated higher power usage throughout the experiment. These results underscore the significance of considering software energy efficiency in web browsers to align with broader sustainability objectives, offering valuable insights for both users and developers seeking environmentally conscious digital solutions.

By integrating energy-efficient practices into software development processes, we can effectively reduce carbon emissions and promote greener, more sustainable software solutions. This approach not only benefits end-users but also contributes to a broader movement toward eco-friendly technology. The experiment highlights the importance of Power Usage Analysis in software design, with newer iterations of desktop browsers like Microsoft Edge and Google Chrome incorporating this critical aspect to enhance energy efficiency. This shift reflects a growing awareness and commitment to sustainable software development practices, shaping a more environmentally conscious digital landscape.

As part of future work, we intend to perform more intensive search on open-source browsers with more GUI related images, animated images, and GPU based image. And also, we planned to expand our research to include an

investigation into the energy efficiency profiles of software developed using commonly utilized technologies. This expansion aims to delve deeper into understanding how different software development approaches and technologies impact power consumption. Each experiment will highlight the significance of utilizing the Power Reading Unit (PRU) in accurately measuring power consumption data during the development of applications. By conducting these experiments, we aim to provide insights into how software developers can optimize their applications for energy efficiency, thereby contributing to sustainability goals. This future work will not only benefit developers by providing them with valuable information on energy-efficient software development practices but also contribute to the broader discourse on green IT and sustainable software engineering.

Author contributions

P.S. Felix was instrumental in the conceptualization and methodology of the project. They performed the experiments, conducted the literature review, and designed the experimental procedures. He also took the lead in drafting the original manuscript, creating visualizations, conducting investigations, and extracting and collecting data. **M. Mohankumar** played a crucial role in validating the research, contributing to visualization and data analysis, and participating in investigations. Additionally, he was responsible for reviewing and editing the manuscript.

Conflicts of interest

The authors have stated that there are no conflicts of interest related to this research. They affirm that their work was conducted independently and without any potential biases or financial influences that could have impacted the study's results.

References

[1] Similarweb LTD's Top Websites - [Online]. Available: https://www.similarweb.com/top-websites/

[2] Similarweb LTD's Top Browsers - [Online]. Available: https://www.similarweb.com/browsers/

[3] Stefan Naumann, Markus Dick, Eva Kern, Timo Johann, The GREENSOFT Model: A reference model for green and sustainable software and its engineering, Sustainable Computing: Informatics and Systems, Volume 1, Issue 4, 2011, Pages 294-304, ISSN 2210-5379, https://doi.org/10.1016/j.suscom.2011.06.004

[4] Achim Guldner, Rabea Bender, Coral Calero, Giovanni S. Fernando, Markus Funke, Jens Gröger, Lorenz M. Hilty, Julian Hörnschemeyer, Geerd-Dietger Hoffmann, Dennis Junger, Tom Kennes, Sandro Kreten, Patricia Lago, Franziska Mai, Ivano Malavolta, Julien Murach, Kira Obergöker, Benno Schmidt, Arne Tarara, Joseph P. De Veaugh-Geiss, Sebastian Weber, Max Westing, Volker Wohlgemuth, Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components—Green Software Measurement Model (GSMM), Future Generation Computer Systems, Volume 155, 2024, Pages 402-418, ISSN 0167-739X, https://doi.org/10.1016/j.future.2024.01.033

[5] Vasilios Andrikopoulos, Rares-Dorian Boza, Carlos Perales, Patricia Lago, "Sustainability in Software Architecture: A Systematic Mapping Study", 10.48550/arXiv.2204.11657

[6] Heiko Koziolek, "Sustainability Evaluation of Software Architectures: A Systematic Review", QoSA-ISARCS '11: Proceedings of the joint ACM SIGSOFT conference - QoSA and ACM SIGSOFT symposium - ISARCS on Quality of software architectures - QoSA and architecting critical systems - ISARCS, June 2011, Pages 3–12 10.1145/2000259.2000263

[7] 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 2019 - Proceedings (pp. 61-64). Article 8797634 Institute of Electrical and Electronics Engineers Inc.. 10.1109/ICSE-SEIS.2019.00015

[8] Noman, H.; Mahoto, N.A.; Bhatti, S.; Abosaq, H.A.; Al Reshan, M.S.; Shaikh, A. An Exploratory Study of Software Sustainability at Early Stages of Software Development. Sustainability 2022, 14, 8596. 10.3390/su14148596

[9] Oyedeji, S.; Seffah, A.; Penzenstadler, B. A Catalogue Supporting Software Sustainability Design. Sustainability 2018, 10, 2296. 10.3390/su10072296

[10] P. Bambazek, I. Groher and N. Seyff, "Application of the Sustainability Awareness Framework in Agile Software Development," 2023 IEEE 31st International Requirements Engineering Conference (RE), Hannover, Germany, 2023, pp. 264-274, 10.1109/RE57278.2023.00034.

[11] P. Bambazek, I. Groher and N. Seyff, "Requirements Engineering Knowledge as a Foundation for a Sustainability-Aware Scrum Framework," 2023 IEEE 31st International Requirements Engineering Conference (RE), Hannover, Germany, 2023, pp. 311-316,10.1109/RE57278.2023.00041.

[12] S. McGuire, E. Schultz, B. Ayoola and P. Ralph, "Sustainability is Stratified: Toward a Better Theory of Sustainable Software Engineering," 2023 IEEE/ACM 45th International Conference on Software

Engineering (ICSE), Melbourne, Australia, 2023, pp. 1996-2008, 10.1109/ICSE48619.2023.00169.

[13] Kern, E. (2018). Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges. In: Otjacques, B., Hitzelberger, P., Naumann, S., Wohlgemuth, V. (eds) From Science to Society. Progress in IS. Springer, Cham. https://doi.org/10.1007/978-3-319-65687-8_23

[14] T. Debbarma and K. Chandrasekaran, "Green measurement metrics towards a sustainable software: A systematic literature review", 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE) 10.1109/ICRAIE.2016.7939521

[15] Calero C., Piattini M., Green in Software Engineering, (2015), pp. 1-327, 10.1007/978-3-319-08581-4

[16] Shalabh Agarwal, Asoke Nath, and Dipayan Chowdhury, "Sustainable Approaches and Good Practices in Green Software Engineering", International Journal of Research and Reviews in Computer Science (IJRRCS) 2012, ISSN: 2079-2557

[17] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. 2016. An empirical study of practitioners' perspectives on green software engineering. In Proceedings of the 38th International Conference on Software Engineering (ICSE '16). Association for Computing Machinery, New York, NY, USA, 237–248. https://doi.org/10.1145/2884781.2884810

[18] Shaiful Alam Chowdhury and Abram Hindle. 2016. GreenOracle: estimating software energy consumption with energy measurement corpora. In Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16). Association for Computing Machinery, New York, NY, USA, 49–60. https://doi.org/10.1145/2901739.2901763