Setting up/Measuring Scientific Experiments on Energy Efficiency: A Guide

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—The correct assessment of energy consumption in software systems is essential for understanding and improving their environmental impact. However, measuring and analyzing energy consumption can be a difficult undertaking. In this study, we investigate the replicability of energy consumption measurements in the broader context of software applications. In fact, despite the application of established best practices for measuring the energy consumption gathered from the literature, the application of such approaches to the *TicketMonster* application produced unexpected results when, leading us to reevaluate the entire procedure. By conducting interviews with seven experts in the field and a comprehensive analysis of our experiments, we identified and addressed methodological obstacles. This article discusses the challenges encountered during the initial process and proposes a refinement of the best practices, influenced by expert opinions. The findings contribute to enhancing the replicability of energy consumption measurements, providing a valuable resource for researchers and professionals in this field.**

***Index Terms*—energy consumption, software sustainability, reliable experiments**

I. Introduction

The increasing prevalence and complexity of software systems in contemporary society have accentuated the need for a comprehensive understanding of their environmental impact [1]. Among the various dimensions influencing the sustainability of software applications, energy consumption stands out as a critical factor [2] [3]. The energy demands of software systems not only impact operational costs but also contribute significantly to the carbon footprint, making correct measurement and analysis essential for informed decision-making and sustainable development [4]. Measuring energy consumption in software systems is a multifaceted challenge, requiring meticulous attention and methodological rigor [5]. The correct assessment of energy usage is pivotal not only for understanding the ecological footprint of software applications but also for optimizing their performance.

Despite the acknowledged significance of measuring energy consumption in software systems, a growing concern surrounds the replicability of such measurements. This paper delves into the nuanced landscape of energy consumption measurements within the broader context of software applications, spurred by anomalous results obtained during the measurement of the *TicketMonster*. These unexpected findings prompted a reevaluation of the best practices of the underlying measurement process. In the pursuit of advancing the correctness of energy consumption measurements, this study addresses the following research questions:

- Which are the best practices for correct energy consumption measurements?
- What challenges emerge when applying established measurement approaches to software systems?
- Which methodological enhancements can be introduced to improve the accuracy and replicability of energy consumption measurements in software systems?

The primary aim of this research is to contribute to the refinement of energy consumption measurement methodologies in software systems. Through a systematic investigation with seven experts in the field and the analysis of the challenges encountered in the measurement process, we seek to provide insights that will inform the development of more robust and correct approaches to measurements. The ultimate goal is to offer a valuable resource for researchers and professionals in the field, fostering a collective effort toward a more accurate energy measurement process. Thus, the contribution of the paper can be summarized as follows:

- General guidelines to measure the energy consumption of software systems gathered from the literature.
- Application of the guidelines to the *TicketMonster* application and analysis of the results.
- Interviews with experts in the field for the identification of the main challenges and best practices to perform energy measurements.
- Set of best practices to enlarge the state of the art and their first assessment on the case study.

The paper is structured as follows: Section II summarizes existing methodologies and approaches for measuring energy consumption of software system identifing common practices and guidelines that are applied and discussed in Section III. Section IV details the interview protocol used for engaging with domain experts, while Section V presents key insights from the experts regarding challenges and recommendations. The refined best practices are applied in Section VI. Thus, Section VII discuss the results, the threaths to validity and the lessons learned. Section VIII concludes the paper.

II. Related Work

In this section, we analyse the approaches that are currently used by researcher to measure the energy consumption on a software. In addition, the best practices for energy consumption measurement retrieved from the literatura are analysed and managed to provide general guidelines to perform a correct experiment. Consider that the steps and tools required for measuring energy consumption may vary according to the specific goals and scenarios, but these practices provide an overall idea of how to enhance the accuracy and significance of energy consumption measurements in software.

A. Energy Measurement Approaches

The energy utilization of software systems has emerged as a significant concern due to its environmental ramifications and its role in contributing to global carbon emissions [6].

Various *methodologies* have been devised by researchers for gauging and predicting the energy consumption of software systems. These methodologies can be broadly categorized into two classes [8]: hardware-based and hybrid approaches. Hardware-based methodologies involve the utilization of energy and power measurement instrumentation, such as *Watts Up? Pro* [10] and *Monsoon* [9], to directly measure the energy consumption of the hardware platform executing the target software system [11] [7]. While offering the most dependable results, this approach necessitates a physical connection to the hardware platform and supplementary tools for data collection and analysis. Additionally, it is unsuitable for assessing the energy consumption of software running remotely or in distributed environments, such as the cloud. To address this limitation, hardware-assisted software methodologies (hybrid) are employed. These approaches utilize mechanisms provided by hardware to measure power and energy consumption, along with associated metrics (e.g., clock frequency, power states, etc.). An example of such an approach is the utilization of hardware performance counters in CPUs [13]. The common software component used in hybrid approaches are software libraries such as *Intel RAPL* [14] [15] and *Alpaca* [16], are commonly adopted to access performance counters.

B. Energy Consumption Best Practices

In the realm of energy consumption *measurement*, substantial efforts have been dedicated to quantifying energy usage across diverse contexts, including data centers [17] [18] [19], networks [22] [23], and mobile devices and applications [20] [21] [24]. However, a comprehensive review of the existing literature reveals a limited number of proposals explicitly proposing best practices and guidelines for conducting such measurements [5] [12]. The authors in [5] discuss the growing significance of energy consumption awareness in software and hardware components. The paper introduces an approach for planning and conducting software energy consumption measurements, aiming to support evidence-based software engineering by providing reliable, comparable, and actionable results. Similarly, in his blog, Luís Cruz [12] presents a scientific guide to setting up energy efficiency experiments, pinpointing the most crucial strategies to minimize biases during data collection and providing strategies for analyzing data to determine normalcy in samples.

C. Energy Consumption Guidelines

Based on the insights from [12] and [5], we have compiled a list of best practices for measuring energy consumption of software. Those best practices are divided into a 3 process stages from the experiment preparation to the analysis of the results.

1) Experiment Preparation:

Define your goals - Clearly identify what you want to measure and why. For instance, are you comparing applications, optimising specific code sections, tracking trends, etc?

Choose your platform - Consider the hardware type carefully (desktop, mobile, etc) and the available measurement tools and techniques available to that platform

Control environment - Minimize external factors like network usage, background processes and fluctuating power sources. Standardize the hardware and software configurations to ensure consistency.

2) Measurement Process:

Baseline measurements - Measure the energy consumption of the idle system (OS only) to isolate the application's impact.

Use an appropriate tool - Perform extensive research on the tool you would like to use to measure energy consumption to ensure it's appropriate to your platform and accurate.

Control workload - Define realistic and repeatable user scenarios representative of typical application usage.

Repatition - Run an experiment multiple times. The number of times can vary, but running between 10 and 30 iterations is recommended to minimize randomness.

Capture relevant data - Log everything. Be sure to keep track of other metrics, including CPU, memory usage, temperature, etc., for correlation and analysis.

Sleep - Give a generous amount of time between experiment iterations to ensure one iteration will not affect the next.

3) Analysis and Interpretation:

Identify hotspots - Analyze data to pinpoint code sections or functionalities with high energy consumption. Tools like profilers can help here.

Consider power models - Understand the power consumption characteristics of your hardware components (CPU, display, network) for better interpretation.

Normalize results - Normalize energy consumption by workload duration or performance metrics for fairer comparisons

Document everything - Document your methodology, tools, configuration, and assumptions for reproducibility and future reference

Although we applied those best practices to software applications, we discovered a high variability in results when repeating the same experiments. Recognizing the need for a more nuanced understanding of energy consumption measurement methodologies, we decided to supplement the best practices found in the literature by conducting interviews with experts in the field.

III. Application to the Case Study

In this section, we apply the best practices retrieved in the literature to measure the energy consumption of the *TicketMonster* application. Thus, after presenting the case study, we delineate the experimental setup necessary to proceed with the measurement and analyse the results.

A. Experiement Preperation

This experiment aims to evaluate the best practices reported in the literature on a software case study. For this purpose, we selected the *TicketMonster* application. The *TicketMonster* monolith application is an example web application demonstrating ticket purchasing. It provides an online environment where users can purchase event tickets, and administrators can manage the related data [30]. The application was originally built to demonstrate combining *Red Hat* and *JBoss* technologies and frameworks. It contains a database of prepopulated Events, Event Categories and venues where a user can purchase several tickets, an Administration panel where an admin can monitor and update content, and a built-in bot that can randomly purchase many tickets.

The application is built using *Maven 3.8.4* (*open JDK 11*) and runs using *Wildfly* (formally *JBoss*) *23.0.2*, packaged within a single *Docker* Container. Thus, to ensure consistent and controlled experimental conditions, we containerized the *TicketMonster* application using *Docker* [25]. This isolation approach eliminates external interference and allows a fair comparison between experiments. In addition, to accurately measure the application's energy consumption, we utilized *Intel Power Gadget* [26], a robust power measurement tool compatible with *macOS* and *Windows*. This tool provides detailed real-time power data, enabling precise energy usage monitoring during the experiments. By creating a realistic workload scenario that simulates more demanding user browsing the website, purchasing a large volume of tickets simultaneously, we aim to stress test an application while monitoring its energy consumption to gain insight. We utilize a standardized *Macbook Pro* environment with controlled network access and minimize background processes and applications.

B. Measurement Process

During the measurement process, the previously defined scenario was divided into two stages. In the first stage, *Selenium* was used to open a browser window and perform various tasks such as navigating through events, selecting a certain event, purchasing 25 tickets, and repeating this process for two more events. This was simulated for five users executing these actions concurrently. Once the frontend execution was completed, the backend execution started, where five bookings were made, all five bookings were retrieved, and each one was updated one by one. Finally, all five bookings were deleted. This was repeated by 50 *Newman* instances in parallel to mimic realistic usage of the application. The entire scenario was repeated 30 times to minimize randomness.

A single experiment iteration consisted of two stages. The first stage was the warm-up phase, where the frontend and backend tasks were tested, and the time taken for each one was recorded. The recorded time was used in the following stages of the iteration. The second stage was the actual iteration, where various tasks were executed, such as starting up the *Docker* container and executing frontend baseline monitoring for a certain amount of time, followed by executing frontend tasks and monitoring for the same amount of time. This was followed by executing backend baseline monitoring and then backend tasks and monitoring for the same amount of time. Finally, the *Docker* container was shut down, and the iteration was ended. After each task within a stage, the experiment would be instructed to sleep for five minutes, so the frontend baseline monitoring would not affect the frontend monitoring task, etc.

During the monitoring phase, *Intel Power Gadget* was utilized via the command line from the experiment script. The script tracked and catalogued various data points, such as CPU utilization, CPU frequency, processor power, cumulative processor energy temperature, and more. All data per iteration was stored in an output folder, which could be later used for data analysis.

C. Analysis and Interpretation

After conducting the initial experiments, we observed several anomalous results that raised concerns about the reliability of the measurements. By exploring both aspects, we aim to understand the observed inconsistencies and their potential causes comprehensively.

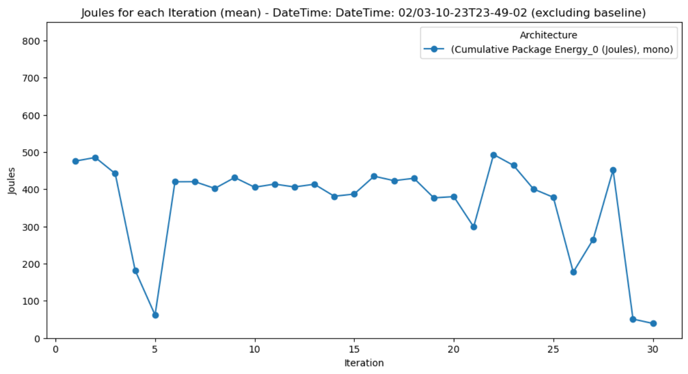

Fig. 1. Experiment on User Heavy Load Scenario October 4th

It has been observed that there are inconsistencies within experiments. Figures 1 and 2 illustrate the results of applying the best practices retrieved from the literature on two different experiments conducted under the same environment and process. As shown in the Figures, there are significant variations in energy consumption within the same experiment. For instance, in the experiment depicted in Figure 1, energy consumption ranges from 39.24 Joules (minimum value) to 493.32 Joules (maximum value) across iterations. This variability within the same experiment raises concerns about measurement errors or external factors affecting the results.

Similarly, Figure 2 shows considerable differences in energy consumption between iterations. The exact values can be seen

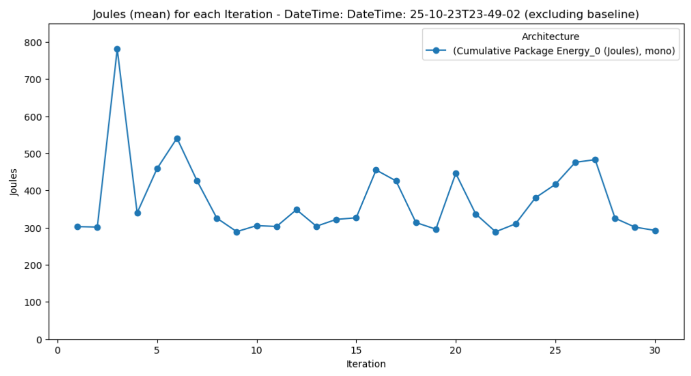

Fig. 2. Experiment on User Heavy Load Scenario October 25

in Table I, which indicates a minimum value of 288.81 Joules and a maximum value of 780.75 Joules, resulting in a large variance of 491.94 Joules between iterations. This substantial within-experiment variation raises major concerns about potential measurement errors or external factors influencing the results.

TABLE I
AVERAGED MONITOR VALUES WITH MIN AND MAX FOR INITIAL EXPERIMENT

Expeirment Date	**Number of Iterations**				
	10	**20**	**30**	**Min**	**Max**
October 4	405.42 J	380.21 J	39.24 J	39.24 J	493.32 J
October 25	305.45 J	446.12 J	780.75 J	288.81 J	780.75 J

Table I presents insights into every 10th iteration in the experiment, along with the Minimum (the lowest value in Joules across all iterations) and Maximum (the highest value in Joules across all iterations) values for both experiments. As highlighted in the Figures and Table I, there are significant differences between experiments conducted on different dates. For example, iteration 3 in Figure 2 consumed 780.75 Joules compared to 442.03 Joules in 1, resulting in a substantial 45% reduction. Such variability across experiments suggests systematic issues that require investigation.

One major issue was the high noise level in the energy consumption data. Despite implementing a sampling strategy with one-second intervals, the measured values fluctuated considerably within each iteration across different iterations. This variability made it difficult to draw meaningful conclusions about the energy efficiency of the application.

To investigate the source of these inconsistencies, we analyzed the experimental setup and identified potential sources of error. The main factors contributing to the noise in the measurements included:

1) *Background Processes* - The system was running several background processes that could not be shut down, which could interfere with the energy consumption monitoring.
2) *Hardware Variability* - The MacBook Pro used for the experiments had a variable core usage pattern, meaning it can automatically adjust the number of CPU cores being used based on the workload and other factors.
3) *Inconsistent Power Supply* - The machine used used an old charger. Although it was left plugged in throughout the experiment, the charger exhibited inconsistent charging behaviour, leading to fluctuating power delivery and affecting the overall energy consumption measurements.

The observed inconsistencies in energy consumption measurements across and within experiments raise significant concerns about the initial methodology's correctness. Thus, it is crucial to address these issues by investigating and implementing refined best practices.

IV. THE INTERVIEW PROTOCOL

In this section, we outline the semi-structured methodology employed for conducting interviews as part of our research. The selection of participants, the protocol followed during the interviews, and the distribution of interview questions are discussed to provide a comprehensive understanding of the data collection process.

A. *Questionnaire Design*

The semi-structured interview [29] is a meeting in which the interviewers ask open-ended questions encouraging free discussions in the direction the interviewee wishes to go. We opted for synchronous interviews [32] [33] to be conducted via *Microsoft Teams* in the English language. We designed a questionnaire and detailed the interview process to specify the steps to be taken and points to bring up during the interviews. The questions aimed to explore tools, measurement processes, challenges encountered during energy consumption measurements within the potential solutions, and the best practices to measure the energy consumption. The main questions driving the interviews are listed below:

1) Which tool or software do you typically use for measuring energy consumption in your projects?
2) Walk me through the typical steps you follow when measuring the energy consumption of a software application. How do you ensure accurate and reliable energy consumption measurements during the development or testing phase?
3) Have you encountered any challenges or issues when measuring energy consumption in software? Please describe them.
4) When faced with challenges in energy consumption measurements, what strategies or actions do you typically take to resolve the issues?
5) What best practices do you follow to ensure accurate and reliable energy consumption measurements throughout the software development lifecycle?

B. *Participant Selection and Interview*

To recruit participants for our interviews, we employed a targeted approach. We disseminated interview invitations through *LinkedIn* and the *X* platforms, accompanied by a visually representative image depicting the specific issue under investigation. This strategy aimed to attract individuals with expertise in the field to engage in discussions with our research team. A total of seven researchers responded to our call as reported in Table II, representing institutions such as the *Vrije Universiteit Amsterdam* (1), *University of Florence* (1), *Indian Institute of Technology Tirupati* (1), *University of L'Aquila* (2), *Fraunhofer IESE Research Institute* (1), *Dalhousie University* (1). The participant pool comprised a diverse group of specialists in energy measurement, including Ph.D. students (2), Post-Doctoral researchers (2), Professors (2), and Research Department Head (1).

Before the interviews, participants were furnished with a predetermined set of questions to familiarize themselves with the topics. Throughout the interview sessions, these questions were visibly presented to ensure a structured and consistent approach. Consent for recording the interviews was obtained from each participant to facilitate accurate transcription and subsequent analysis. The interview duration ranged from 15 to 40 minutes, allowing for an in-depth exploration of the subject matter. To mitigate potential biases, interviewers refrained from interrupting participants during their responses.

C. *Data analysis*

The gathered data were then analyzed according to an open coding approach [34] based on *grounded theory* [35]. Thus, in order to enhance the rigor and accuracy of our research on energy consumption measurements practices we employed a multiple steps process to analyze the data. Frist, we analysed the notes that were diligently taken throughout the meetings capturing key insights and observations. Secondly, we cross-verified our handwritten notes with the corresponding video recorded transcriptions. The outcome of those two phases was a list of code to be used for the analysis. Those codes include the tool used, the measurement process, the challenges faced, and the best practices. Finally, to leverage the capabilities of advanced language processing technology, we employed *ChatGPT 4.0* to generate summarized versions of the transcriptions. The command used for the summarization on *ChatGPT 4.0* is shown in the following and considers the code we generated in the first two steps of the process.

"Can you extract from the following text: 1. the tools used to measure the energy consumptio, 2. the process the interviewee use to measure the energy consumption, 3. the challenges they faced running experiments, 4. best practices."

The integration of both manual and technological verification methods reinforces the robustness of our data collection process, contributing to the overall reliability of our findings in the investigation of energy consumption measurements practices.

V. Interview Results

This section carefully presents the results of our interviews discussing the tools used by the interviewee, the common steps adopted for the correct measurement of the energy consumption of software applications, the main challenges reported by the interviewees, and the our peoposed refinement of the best practices.

A. *Tools Used*

The experts highlighted two main categories of tools: hardware devices such as smartplugs, and Watts Up and software tools that rely on metrics extracted from the operating system and kernel as RAPL, and Scaphandre.

Hardware devices, particularly those that measure voltage and current flowing into the plug, were considered theoretically more reliable for energy consumption measurement. On the other hand, software tools, while potentially less reliable in theory, offer the advantage of measuring the energy consumption of individual applications running on a system. They can also provide estimates of possible energy consumption.

The preference for working at a low level and direct interfacing with the kernel was expressed by some experts, emphasizing the ability to record multiple events simultaneously. In this context, Intel RAPL was mentioned as one of the tools used, along with built-in tools provided by operating systems for measuring memory and CPU consumption. Code Carbon was also cited for its support and flexibility. The experts discussed the development of their own tool suite.

Some interviewees expressed uncertainty about continuing the development of their own tools, considering a potential switch to frameworks provided by others. Their engagement with companies developing software in the field indicated a collaborative approach to advance research in energy consumption measurement.

B. *Measurement Process*

Insights derived from interviews with seven experts in energy consumption measurement unveil a sophisticated and meticulous strategy for evaluating software energy usage. The experts prioritize isolating application behavior by conducting experiments on a pristine Ubuntu distribution, minimizing background processes, and selectively activating only the necessary software components. This intentional approach aims to mitigate potential interference from unrelated system activities. A key concern for the experts is the practical challenge of effectively turning off all background processes, highlighting the difficulties in achieving this goal. To bolster the reliability of energy consumption measurements, the experts advocate for repetitiveness in the measurement process, recognizing the inherent variability and potential unreliability of the tools employed. The experts also emphasize the significance of stabilizing hardware conditions and temperatures to prevent distortions in measurements. Implementing warm-up and cool-down phases before and after each experiment contributes to maintaining consistent conditions and ensuring the validity of results. Routine operational checks on the operating system and deploying applications for preliminary runs are part of the experts' process to validate experiment reproducibility. A comprehensive data analysis phase involves assessing the

TABLE II
THE INTERVIEWEES (ALPHABETICALLY ORDERED)

Name	Country	University/Company	Role	Area of Expertise
Carlo Centofanti	Italy	University of L'Aquila	Post-doctoral Researcher	Multi-access, Edge Computing, Edge Native applications, Cloud Computing, SDN
Sridhar Chimalakonda	India	Indian Institute of Technology Tirupati	Associate Professor	Software Engineering, Educational Technologies, Human-Computer Interaction, Computing Research for Society
Rafiullah Omar	Italy	University of L'Aquila	Ph.D. Student	Green AI
Vincenzo Stoico	Netherlands	Vrije Universiteit Amsterdam	Post-doctoral Researcher	Software Performance, Energy Consumption, Software Engineering, Embedded Systems, Model-Driven Engineering
Saurabh Rajput		Dalhousie University	Ph.D. Student	
Roberto Verdecchia	Italy	University of Florence	Assistant Professor	Software Engineering, Software Architecture, Technical Debt, Software Testing, Green Software
Joachim Weber	Germany	Fraunhofer IESE Research Institute	Research Department Head	

normal distribution of measurements, and a tuning phase, comprising 10 to 30 iterations, is conducted to refine the experimental setup. Table III summarize the common steps of the measuring process employed by the interviewee.

C. Challenges Faced when Measuring

One prominent issue is the reliance on metrics generated by the kernel in software-based systems. These metrics may lack accuracy, and transitioning between systems, even within the same family of Intel processors, can yield varying results for identical workloads. The use of stress tools to emulate constant CPU workloads also showed inconsistencies in experiment results. The absence of a standard process for energy consumption measurement raises questions about the reliability of the obtained data. Experts ponder whether the variations stem from the processes followed or the tools utilized. In revisiting the RAPL original paper, concerns arise about its universality across different Intel processors and architectures. The limitations of RAPL, which reports only on CPU and memory, become apparent with the advent of GPUs, highlighting the need for broader coverage in energy consumption measurements. Another challenge identified is the difficulty in attributing energy consumption to specific software components, especially considering the diverse range of devices running the front end of modern software. This variability makes it challenging to estimate and predict energy consumption accurately. Issues with measurement tools were also raised during the interviews. For instance, the minimum resolution of 1 millisecond in Intel RAPL measurements proves insufficient for unit testing, particularly when dealing with methods running at much shorter intervals. The lack of tools providing accurate process-wise energy consumption data remains a significant hurdle. The sampling rate of measurement tools poses additional challenges. If the rate is too high, it can increase overheads in readings, while a low sampling rate may lead to inaccurate results. Striking a balance is crucial for obtaining meaningful data without compromising efficiency. Time synchronization emerged as another concern, especially when using hardware tools. Ensuring alignment between power readings, calculations, and system state requires meticulous attention to detail. Compatibility issues with existing libraries were mentioned, emphasizing the need for less granular analysis tools like Watts Up Pro or soldering in certain scenarios. Experts also highlighted challenges related to background software processes and behaviors that may go unnoticed, necessitating a more comprehensive understanding of how software interacts with infrastructure and resources. Table IV summarize the main challenges arised by the interviewee.

D. Best Practices

Based on the interview results from expert in the field of measuring software energy consumption, we compiled the following list of best practices to enlarge the set found in the literature:

*1) **Experiment Preparation**:*

Pre-Experiment Tuning - Conduct a preliminary phase to identify and address potential issues like background processes, hardware fluctuations and tool interference.

Controlled Testing Environment - Minimize external factors by standardizing hardware configuration, software versions and network conditions.

Platform-Specific Tool Selection - Carefully choose tools based on your market platform (desktop, mobile, embedded), measurement goals (fine-grained analysis, overall trends) and desired granularity (application, code section).

Standardize Experiment Procedures - Define and document a clear measurement process to ensure repeatability and minimize bias across experiments

*2) **Measurement Process**:*

Performance-Aware Measurement - Measure not only energy consumption but any other possible metrics (CPU, memory, Temperature etc) to understand software-resource interactions.

TABLE III
COMMON ENERGY CONSUMPTION MEASUREMENT PROCESS STEPS

Aspect	Description
Isolation of Application Behavior	Use plain Ubuntu distribution or a clean OS image for specific experiments.
Minimizing Interference	Terminate unrelated background processes. Wait for hardware stabilization post-process termination. Calculate baseline power. Stabilize temperature to prevent measurement distortion.
Data Analysis and Tuning	Analyze data for normal distribution. Includ tuning phase with experiment-specific cool-down periods. Conduct 10-30 iterations of experiments.

TABLE IV
CHALLENGES IN MEASURING ENERGY CONSUMPTION

Aspect	Description
Lack of Standard Process	Absence of a standardized process for energy consumption measurement. Uncertainty whether variations result from the process employed, the background processer, or tools used.
Attribution Challenges	Difficulty in attributing energy consumption to specific software components. Complexity due to the diverse devices running the front end of modern software.
Process-Wise Measurement	Absence of accurate tools for process-wise energy consumption measurement.
Tool Sampling Rate	Challenges in balancing the sampling rate to avoid increasing overheads while ensuring accurate readings.
Universality of Metrics	Questions about the universality of metrics, especially in the context of different Intel processors and architectures. Limitations of metrics reporting only CPU and memory, neglecting GPUs.

Linux Preference - Opt for Linux-based testing environments whenever possible due to its inherent flexibility and control over system resources compared to other operating systems.

Automation - Experiments can take a very long time (for us, a single experiment can take up to 26 hours), automate as much as you can to avoid intervening in the process.

Controlled Sleep Intervals - Between experiments and experiment iterations, sleep for a generous amount of time (e.g. 5 minutes) to ensure stable and consistent testing environments for subsequent iterations and runs.

*3) **Analysis and Interpretation**:*

Experiment Repetition and Validation - Repeat experiments using different hardware to validate results and reduce bias.

Tool Limitation Awareness - Understand the inherent limitations and potential inaccuracies of each chosen tool to interpret results cautiously.

VI. BEST PRACTICES FIRST ASSESSMENT

This section details the improved experimental setup and analysis approach employed to measure the *TicketMonster* application's energy consumption, incorporating valuable insights from expert interviews and addressing limitations identified in previous attempts. The main goal of this refined methodology is to achieve more consistent results.

A. Experiment Preparation

We established a standardized testing environment to minimise external influences by separating the server and client components into distinct machines. This eliminated potential interference and fostered a clean and isolated testing environment. For further isolation, the client machine was remotely controlled using *SSH* over a secured Wi-Fi network. Additionally, we ensured consistency and reduced background processes by utilizing a fresh installation of *Ubuntu Server*

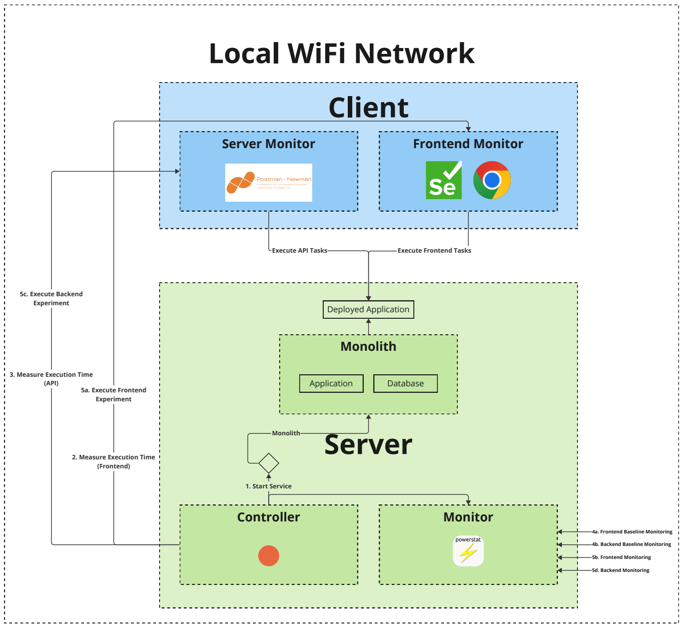

Fig. 3. Architecture of the refined methodology

20.04 LTS on a *Dell XPS 13 7390* laptop. In addition, We switched from *Intel Power Gadget*, which is incompatible with *Linux*, to *Powerstat* [31], a power consumption monitoring tool designed explicitly for Linux-based systems that utilize the Intel RAPL interface. *Powerstat* provides more granular and consistent power measurements, further enhancing the accuracy and reliability of our results. Figure 3 provides an overview of the architecture of the refined experiment implementation. It shows the separation of client and server, which would prevent any frontend testing tasks like opening a *Chrome* window from affecting the experiment results.

B. Measurement Process

We meticulously defined and documented a clear measurement process for optimal repeatability and reduced bias across experiments. Even in this case we considered the same user heavy scenario analysed in Section III.

Also in this case, we created an automated script written in *Bash*, *Python*, *Selenium* and *Newman* minimized human intervention and ensured accuracy. This script triggered the user scenarios on the client machine over *SSH*, collected real-time energy consumption data from the serverside using *Powerstat*, and logged additional metrics (CPU, memory, temperature, etc.) with timestamps for thorough analysis. Additionally, we implemented generous sleep intervals (5 minutes) between all stages of an iteration, as previously depicted in the initial methodology, to ensure a stable testing environment throughout the process.

C. Analsis and Interpretation

This section discusses the variability observed in individual experiment iterations for the refined experiment based on improved best practices and expert interviews. The aim is to compare this data against the initial experiment results, as shown in Figures 1 and 2, by maintaining the same simulation scenario of a user-heavy workload.

Figure 4 shows the results of the same scenario performed on the improved experiment setup, as described in the previous subsection. The graph visually depicts the energy consumption pattern across 30 iterations and emphasizes the consistency achieved through the refined methodology. Despite having different absolute values, the scenario maintains a stable pattern with minimal variation across iterations. Compared to the initial experiment, the variation between Joules values between iterations is far less in the refined experiment.

TABLE V
AVERAGED MONITOR VALUES WITH MIN AND MAX FOR REFINED EXPERIMENT

Expeirment Date	**Number of Iterations**				
	10	**20**	**30**	**Min**	**Max**
December 6	66.27 J	65.55 J	65.50 J	61.54 J	74.03 J

Table V displays selected iterations' minimum and maximum energy consumption values. The refined experiment showcased a minimum energy consumption of 61.54 J and a maximum of 74.03 J. In contrast, I showed a minimum of 39.24 J, a maximum of 493.32 J in one experiment, a minimum of 288.81 J, and a maximum of 780.75 J in another. These results indicate a significant decrease in energy consumption fluctuations, which further supports the improved consistency offered by the refined approach.

The refined methodology successfully addresses the inconsistency issue identified in the previous attempt, as indicated by the minimal fluctuation within the iterations and reduced high and low values. This is a promising indication of the potential to achieve reliable and consistent energy consumption measurements across various applications.

VII. DISCUSSION AND LESSON LEARNED

Based on the initial and refined methodologies and insights from the interviews, this section discusses the improvements observed, implications for understanding *TicketMonster*'s energy consumption, and potential threats to validity.

A. Experimental Results and Comparison

A key difference between the initial and refined methodology is the resulting graphs depicting energy consumption across different iterations. Figures 1, and 2 show the energy (in Joules) for each iteration. However, these graphs exhibited inconsistency in terms of energy consumption values and even trend patterns. This inconsistency made it difficult to draw definitive conclusions about the relative efficiency of the architecture under test.

The refined methodology, as seen in Figure 4, produced a single graph that illustrates the same scenario as the previous iterations. This graph shows a high level of consistency in terms of energy consumption trends across all iterations, with minimal differences in values between them. This suggests that the refined methodology has significantly improved the quality and reliability of the collected data.

B. Improvements and Implications

The following text describes the improvements and implications of a refined methodology that led to better results in measuring the energy consumption of the *TicketMonster* application. The new approach included a clean Ubuntu server distribution, hardware separation, and minimized software installations, which had several advantages.

Adopting a clean Ubuntu server distribution minimized background processes, ensuring that energy consumption measurements were not interfered with. It also standardized the environment, reducing potential biases introduced by specific application configurations or OS customizations. Finally, it minimized potential noise sources in the data, such as unexpected software interactions or resource demands from unnecessary applications, leading to more reliable measurements.

Separating the server and client onto distinct machines ensured a clean and controlled experiment environment, eliminating interference between server and client processes. It also allowed for more focused monitoring, as the server-side monitoring excluded frontend actions like opening Selenium,

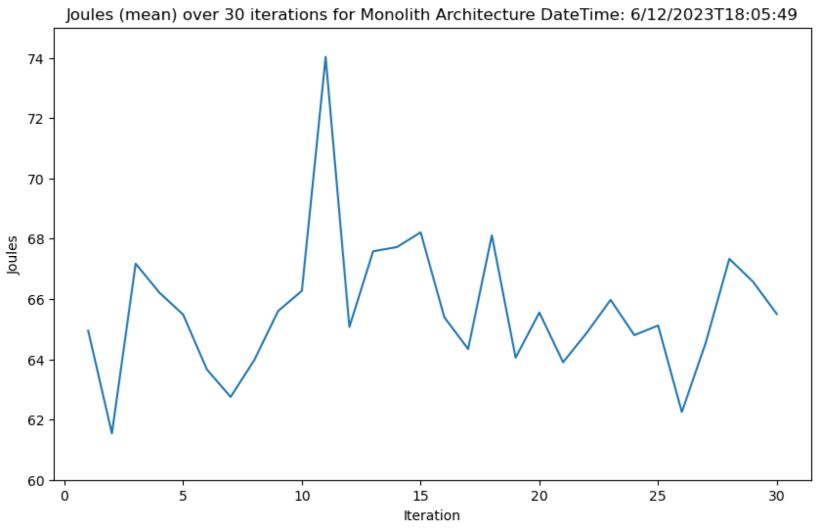

Fig. 4. Refined methodology December 6th

which provided more precise measurements. Furthermore, it simplified the experimental setup and data analysis, removing potential confounding factors due to interactions between server and client components.

Minimizing software installations on the server resulted in a lower baseline energy consumption, making detecting changes due to experiment activities easier. It also enhanced the signal-to-noise ratio, making the signals related to the experiment more prominent, leading to cleaner and more interpretable data. Finally, it improved the validity of the measurements, making them more representative of the actual energy consumption of the *TicketMonster* application.

Overall, the refined methodology, particularly the adaptation of a clean Ubuntu server distribution, hardware separation, and reduced noise levels, contributed significantly to achieving more reliable, consistent, and generalizable results in measuring the energy consumption of the *TicketMonster* application. These findings align with the valuable insights and best practices shared by interviewed experts, highlighting the importance of careful methodology design and consideration of environmental factors for accurate software energy consumption analysis.

C. Threats to Validity

The internal validity of our study may be influenced by several factors. Firstly, the limitation in the number of interview participants to seven may impact the generalizability of our findings. The small sample size could potentially hinder the representation of diverse perspectives within the software systems domain. While the insights gained from these participants contribute valuable qualitative data, it is crucial to acknowledge the inherent limitation of the sample size in drawing broader conclusions. Additionally, the internal validity is affected by the experimental design employed in our study. The refinement of best practices presented as a contribution in our paper has been tested only once in the case study. This single experimentation limits our ability to generalize the findings across various contexts and scenarios. The variations observed in different iterations of the same experiment may be subject to unique circumstances that are not accounted for in a broader context.

In terms of external validity, the generalization of our findings is further constrained by the application of the refined best practices to only one software system. While the case study provides valuable insights into the specific context under consideration, it may not be representative of the broader software systems landscape. Therefore, caution should be exercised when extrapolating the results to different software environments. Moreover, the lack of experimentation of the new best practices on different hardware platforms poses a threat to the external validity of our study. Software performance can be influenced by hardware configurations, and the absence of testing across diverse hardware platforms limits the generalizability of our findings. It is essential to acknowledge that the application of best practices may yield different results when executed in varied hardware environments, and our study does not account for this potential variability.

VIII. Conclusion

This study delves into the intricate landscape of energy consumption measurements within the broader context of software applications, prompted by unexpected results during the measurement of the *TicketMonster*. Despite the acknowledged significance of measuring energy consumption in software systems, concerns about the replicability of such measurements persist. The research questions explored best practices, challenges in applying established measurement approaches, and methodological enhancements to improve the accuracy and replicability of energy consumption measurements. Through a meticulous investigation involving seven domain experts, challenges and methodological obstacles were identified and addressed. The contributions of this paper encompass general guidelines derived from existing literature, their application and analysis in the context of the *TicketMonster* case study, insights obtained from expert interviews, and a set of refined best practices. This collective contribution aims to advance the state of the art in energy consumption measurement methodologies for software systems, providing valuable insights for researchers and professionals in the field. Despite the rigor applied in this study, internal validity is limited by a small sample size in interviews, potentially affecting the generalizability of findings. Additionally, the experimental design, with refined best practices tested only once in the case study, may limit generalization across various contexts. External validity is further constrained by the focus on a single software system, cautioning against broad extrapolation of results to different software environments. The absence of experimentation on various hardware platforms poses a threat to external validity, recognizing the potential variability in results across diverse hardware configurations. In conclusion, this research contributes to the ongoing discourse on energy consumption measurement in software systems, offering valuable insights and recommendations. Future work could expand on these findings, considering a broader range of software systems, larger participant samples, and experimentation across diverse hardware platforms to enhance the robustness and generalizability of the proposed best practices.

ACKNOWLEDGMENT

This work has been supported by the *omissis*, funded by the *omissis*

REFERENCES

[1] G. A. García-Mireles, M. Á. Moraga, F. García, C. Calero, and M. Piattini, "Interactions between environmental sustainability goals and software product quality: A mapping study,", Information and Software Technology, 95, 108-129, 2018.

[2] C. C. Venters, R. Capilla, S. Betz, B. Penzenstadler, T. Crick, S. Crouch, ... and C. Carrillo, "Software sustainability: Research and practice from a software architecture viewpoint," Journal of Systems and Software, 138, 174-188, 2018.

[3] N. Condori-Fernandez, and P. Lago, "Characterizing the contribution of quality requirements to software sustainability," Journal of systems and software, 137, 289-305, 2018.

[4] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: quantifying the carbon footprint of computation," Advanced science, 8(12), 2100707, 2021.

[5] L. Ardito, R. Coppola, M. Morisio, M. Torchiano, and M. Risi, "Methodological Guidelines for Measuring Energy Consumption of Software Applications," Sci. Program., 2019.

[6] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H. H. S. Lee, G. Y. Wei, ... and C. J. Wu, "Chasing carbon: The elusive environmental footprint of computing," In 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 854-867). IEEE, 2021.

[7] A. Noureddine, R. Rouvoy, and L. Seinturier, "A review of energy measurement approaches," ACM SIGOPS Operating Systems Review, vol. 47, no. 3, pp. 42–49, 2013.

[8] T. A. Ghaleb, "Software energy measurement at different levels of granularity," In 2019 International Conference on Computer and Information Sciences (ICCIS) (pp. 1-6). IEEE, 2019.

[9] "Monsoon Power Meter," https://www.msoon.com/LabEquipment/PowerMonitor, accessed on January, 2024.

[10] J. M. Hirst, J. R. Miller, B. A. Kaplan, and D. D. Reed, Watts up? Pro AC power meter for automated energy recording: A product review. Behavior Analysis in Practice, 6, 82-95, 2013.

[11] N. Kulikov, E. Yaitskaya, A. Shvedova, and V. Zhalnin, "Power Consumption Meter for Energy Monitoring and Debugging," In Proceedings of International Scientific Conference on Telecommunications, Computing and Control: TELECCON 2019 (pp. 499-511). Singapore: Springer Singapore, 2021.

[12] L. Cruz, "Green Software Engineering Done Right: a Scientific Guide to Set Up Energy Efficiency Experiments," http://luiscruz.github.io/2021/10/10/scientific-guide.html, 2021.

[13] X. Wu and V. Taylor, "Utilizing Hardware Performance Counters to Model and Optimize the Energy and Performance of Large Scale Scientific Applications on Power-Aware Supercomputers," 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Chicago, IL, USA, 2016, pp. 1180-1189, doi: 10.1109/IPDPSW.2016.78.

[14] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "RAPL in Action: Experiences in Using RAPL for Power measurements," ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), 3(2), 1-26, 2018.

[15] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using RAPL," ACM SIGMETRICS Performance Evaluation Review, 40(3), 13-17, 2012.

[16] L. Goldberg, J. Katticaran, and A. Mhaidli, "Energy profiling with Alpaca," In Companion Proceedings of the 2016 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity (pp. 69-70), October 2016.

[17] J. Arjona Aroca, A. Chatzipapas, A. Fernández Anta, and V. Mancuso, "A measurement-based analysis of the energy consumption of data center servers," In Proceedings of the 5th international conference on Future energy systems (pp. 63-74), 2014.

[18] L. Gyarmati, and T. A. Trinh, "How can architecture help to reduce energy consumption in data center networking?," In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (pp. 183-186), 2010.

[19] C. Dupont, M. Sheikhalishahi, F. M. Facca, and F. Hermenier, "An energy aware application controller for optimizing renewable energy consumption in data centres," In 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC) (pp. 195-204). IEEE, 2015.

[20] E. Oliver, "Diversity in smartphone energy consumption," In Proceedings of the 2010 ACM Workshop on Wireless of the Students, by the Students, for the Students (pp. 25-28), 2010.

[21] C. Wilke, C. Piechnick, S. Richly, G. Püschel, S. Götz, and U. Aßmann, "Comparing mobile applications' energy consumption," In Proceedings of the 28th Annual ACM Symposium on Applied Computing (pp. 1177-1179), 2013.

[22] C. Lange, D. Kosiankowski, C. Gerlach, F. J. Westphal, and A. Gladisch, "Energy consumption of telecommunication networks," In 2009 35th European Conference on Optical Communication (pp. 1-2). IEEE, 2009.

[23] J. Baliga, R. Ayre, K. Hinton, and R. S. Tucker, "Energy consumption in wired and wireless access networks," IEEE Communications Magazine, 49(6), 70-77, 2011.

[24] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, and S. Tarkoma, Modeling, profiling, and debugging the energy consumption of mobile devices. ACM Computing Surveys (CSUR), 48(3), 1-40, 2015.

[25] "Docker," https://www.docker.com/, accessed on January, 2024.

[26] "Intel Power Gadget," http://tinyurl.com/ydhrmvyp, accessed on August, 2023.

[27] "Selenium," https://www.selenium.dev/, accessed on January, 2023

[28] "Newman CLI," https://github.com/postmanlabs/newman, accessed on January, 2023

[29] C. Robson, "Real world research: A resource for social scientists and practitioner-researchers," Wiley-Blackwell, 2002

[30] Red Hat, I. 1.2. "About the TicketMonster Application Red Hat JBoss Developer Studio 8.0 — Red Hat Customer Portal," http://tinyurl.com/33txkvre

[31] "PowerStat," https://github.com/ColinIanKing/powerstat, accessed on January, 2023

[32] S. E. Hove, and B. Anda, "Experiences from conducting semi-structured interviews in empirical software engineering research," In 11th IEEE International Software Metrics Symposium (METRICS'05) (pp. 10-pp). IEEE, 2015.

[33] P. E. Strandberg, "Ethical interviews in software engineering," In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1-11). IEEE, 2019.

[34] J. Saldana, “The coding manual for qualitative researchers,” sage, 2021
[35] K. J. Stol, P. Ralph, and B. Fitzgerald, “Grounded theory in software engineering research: a critical review and guidelines,” In Proceedings of the 38th International conference on software engineering (pp. 120-131), 2016.