

Original Article

# SEFGAST: Step-Up to Environment Friendly Green Automated Software Testing

Anithakrishna .G<sup>1</sup>, M. Mohankumar<sup>2</sup>

<sup>1</sup>Research Scholar, Department of CS, IT and CA, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India.

<sup>2</sup>Associate Professor, Department of CS, IT and CA, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India.

<sup>1</sup>anithamohanvm@gmail.com, <sup>2</sup>mohankumar07@gmail.com

**Abstract** - In the current world scenario, sustainable development has a magnificent role, especially in the two sizable fields of information and communication, viz., Green Technology and Green Computing. Green IT make ICT more sustainable and instigate solutions to set up and use hardware in an energy-efficient way. Green by IT focus on software-based solutions to make energy-efficient utilization of resource and to diminish the negative impact on the environment. The energy-efficient software development model is the right way toward sustainability. Developing a software product is not only bound to writing its core code, but it also includes many phases from planning to maintenance. To erect an energy-efficient green software, need to make the whole process green. This paper proposes a technique to apply green concepts on software testing to efficiently gather energy consumption related information. For the same need a closer look at green software and green software engineering, and through this proposed work applied green principles to measure, analyze and optimize operations on software testing.

**Keywords** - Green software, Testing, Sustainability, Green IT, Energy.

## I. INTRODUCTION

Green software development describes a paradigm in which software administrators, developers, and testers can make the solutions and techniques more energy efficient. Environment sustainable development deploy tools to optimize the automated processes to save energy. Along the software development life cycle, testing activities are needed time and during the initial development to detect and fix errors in and later to depend on maintenance type.

The proposed work is to improve energy transparency during the testing phase. Automated test scripts have a serious role in energy consumption, and combining test case coverage and energy measurement tool effectively localize code level energy consumption. The proposed technique analyzes energy values of different hardware components like CPU, DRAM, GPU etc., to compute energy utilized [21] by a test script at a particular timestamp. It will provide statistical proof of energy usage of the current automated test script and suggestions to

improve energy efficiency. The power reading defines energy consumption with respect to a timestamp, and the total energy consumption of the application is obtained by summing up all the measured values. The proposed work is a Linux based analysis tool that provides an estimation of energy consumption for a particular test script.

Green computing is a major concept that applies green metrics [5,14] to hardware and software components. Green software life cycle model, focus on energy [10,11] efficiency of software.

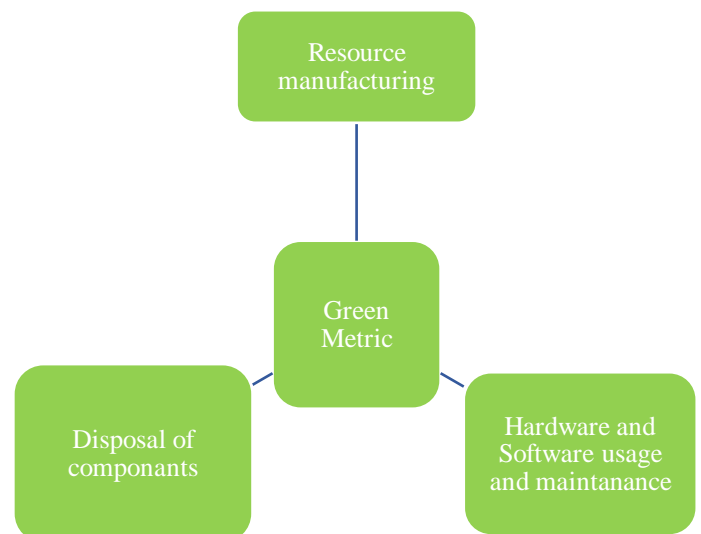


Fig. 1 Green approach

Usage of components refers to the use of computer components in an environment friendly [7] manner. Disposal refers to the recycling of E-waste in an effective manner. Resource manufacturing focus on the manufacturing process of computer resources with reduced [6] impact on the environment. Green software engineering focuses on developing sustainable software which follows green metrics [9,15]. Here the approach is credulous, which analyse the energy consumption at runtime to identify the portion of code which is avaricious to energy.



## II. LITERATURE REVIEW

Christof Ebert presented an article that proposes technical, social and environmental solutions [24] and a paradigm shift on reducing the energy consumption of software. It envisages ecologic behaviour as a business opportunity, thus adopting it as a tool for conserving our planet. Timo Johann and Markus hick presented a paper that highlighted the white box measurement approach in measuring energy consumption [25] of similar modules in the software. It demonstrates a method to develop and apply metrics [8,22] using the source code instrumentation technique to find the resource-intensive part for energy saving.

Devi J and Bhatia K conducted a study on the functioning of selenium tools in software automation and had an analysis of the [1] testing structure. K Lokeshwari and M Kannan conducted a study on three automated tools named Selenium, QTP and Load Runner. For factors like [3], efficiency, cost, platform support has major credit to selenium, whereas Load runner has a credit on programming language support. QTP has high credit on usability.

Kaur P and Agnihotri M analysed and presented a paper on green practice, which includes handling peripheral resources,[2] machines and handling of digital investment. High utilization of energy cause environmental problems like huge carbon emission [4], and the goal of green is to find a solution for energy consumption

Marcus Hahnel and Bjorn Dobel presented a paper to share their experience in using the Running Average Power Limit (RAPL) energy sensor available in recent intel CPU for determining the fine-grained power consumption of short-running code path while executing a single function within an application.

Roberto Verdecchia and Eva Kern conducted a study that illustrates an approach to identifying energy hotspots at the source code level. It takes an application and uses [19] spectrum-based analysis process to localize the portion. Rui Pereira and Marco Couto [16] implements a language-independent tool to locate energy-inefficient fragment in a program's source code, and it uses a spectrum-based fault localization method to create energy ranking of a source code fragment.

## III. METHODOLOGY

### A. Power and Energy

This section is defined to review the relationships between software and power. Power is defined as the rate of doing work measured in Watts. It is recorded on the computer over a small finite value of time. Power is the rate of energy consumption at a particular moment. In a generic way, energy efficiency can be measured as

$$\text{Energy Efficiency} = \frac{\text{Useful Work Done}}{\text{Energy Used}}$$

The quality aspect of the efficient model [17] belongs run time efficiency, CPU intensity, memory usage and idleness.

## IV. RELATED WORK

The two sorts of energy monitoring policies to observe the energy usage of a program are external and internal evaluators. External energy monitors utilize the readings such as voltmeter, ammeter etc. It evaluates the system as a whole and has limited utility to monitor individual programs because they are not comminuted to identify energy usage [12,13] at the component level.

Internal evaluators are integrated as power monitoring [20] within the system. This is achieved by measuring energy registers, process wakeups and CPU state transitions.

### A. Code Optimization

Code optimization improves energy consumption by reducing the work performed, which include dead code elimination, loop optimization, common subexpression elimination. An example of such a thing is demonstrated in figure 2. Here the code  $b*a$  is optimized and assigned to a new variable which avoids recomputing of same code multiple times and saves energy.

Before optimization

$c = b * a + f$

$x = b$

$d = b * a * e$

After optimization (Dead code and common sub expression elimination)

$t = b * a$

$c = t + f$

$d = t * e$

**Fig. 2 Sample code to show the role of code optimization**

Dead code elimination is implemented by removing the code set which is no longer used in the application. In the code snippet,  $x$  is assigned with a value that is no longer used in the application.

### B. Energy Model

In application model energy consumption of a program can breakdown as

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

$E_{\text{(active)}}$  is the application's running time,  $E_{\text{(wait)}}$  is the waiting time for other components, and  $E_{\text{(idle)}}$  is the time in which the system is not performing any work [23] for the specific application.

## V. APPROACH

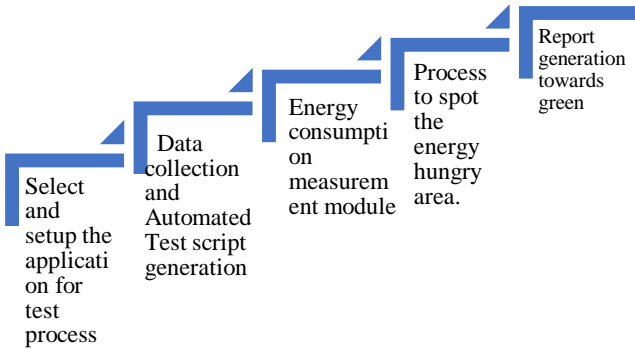
Step-up to Environment-Friendly Green Automated Software Testing aims at energy glassiness during the test phase for test script developers to localize specific energy-

hungry portions of the code. This paper presents Energy Spot Measuring Tool, which refers to runtime hardware energy consumption. More specifically, the approach consists of five steps.

Step 3: Energy consumption measurement module, which prepares device under test for energy measurement and will carry out the measurement process by analyzing the CPU, Memory and Disk resources.

Step 4: Spotting the energy-hungry area of automated script

Step 5: Generation of energy usage report, which focuses on green metrics.



**Fig. 3 Overview of SEFGAST**







Step 1: Select and set up the specific application for which the quality analyst supposed to perform the test process

Step 2: Automated Test Script Generation. Here, consider the test script written by both experienced and fresh testers.

**VI. EXPERIMENTAL PROCEDURE**

We have conducted an experiment using Selenium, an automated testing tool and the proposed ESMT (Energy Spot Measuring Tool) to analyze energy consumption while performing the testing activities.

To carry out the experiment, need to select an automated testing tool. Software testing is a vital part of the software development life cycle which is used to ensure that the application is defect-free and to appraise the functionality of the software. The traditional method of software testing technique, which prepares and execute test cases manually, is a most stringent process which consumes more time taken to automated testing. The automated testing tools implement a test suite that can run or replay test cases without the intervention of human resources. A large collection of testing tools was available, and each has its own strength and limitations. A few tools like Selenium, Katalon, UFT, Test complete, Watir are topmost in this field. Various evaluation criteria have been applied for the selection of an appropriate testing tool.

<ul style="list-style-type: none"> <li>• It is an Open Source software</li> <li>• For web application.</li> <li>• Have record playback feature.</li> <li>• It support all programming languages.</li> <li>• used in Windows, Linux, and OS X platforms.</li> </ul> <p>Selenium </p>	<ul style="list-style-type: none"> <li>• It is not a free software</li> <li>• It is a functional automated platform</li> <li>• Desktop,web and mobile application</li> <li>• it support VB script, JScript, C++ script, C# script, Python, .NET, JAVA, Android, iOS, Visual C++, VB, HTML 5.</li> </ul> <p>Test Complete </p>	<ul style="list-style-type: none"> <li>• It is a free software.</li> <li>• It is a platform for web applications and mobile application.</li> <li>• Used in Windows Linux OS X.</li> <li>• It support only java</li> </ul> <p>katalon </p>
<ul style="list-style-type: none"> <li>• It is a free software.</li> <li>• For web applications.</li> <li>• It support only Ruby.</li> <li>• It provide cross browser testing.</li> <li>• Used in Windows Linux OS X platforms.</li> <li>• Not widely used as selenium.</li> </ul> <p>Watir </p>	<ul style="list-style-type: none"> <li>• It is not a free software.</li> <li>• It is a functional automated platform.</li> <li>• Desktop,web and mobile application.</li> <li>• It support only vb script.</li> </ul> <p>QTP </p>	<ul style="list-style-type: none"> <li>• Performance Testing tool</li> <li>• It is a desktop and web based applications</li> </ul> <p>LoadRunner </p>

**Fig. 4 Lookup on various Testing Tools**

**Table 1. Merits and demerits of various software testing tools**

Tools	Merits	Demerits
<b>Selenium</b>	<ul style="list-style-type: none"> <li>• It can run under multiple browsers.</li> <li>• It supports Windows/Mac or Linux.</li> <li>• It supports parallel processing, which helps to save execution time.</li> <li>• It provides reduced hardware resource utilization. It provides a remote control server.</li> <li>• As an open-source, it allows to share and modify the code.</li> </ul>	<ul style="list-style-type: none"> <li>• Lack of technical supporting team.</li> <li>• It needs support from a third party framework for binding, result reporting etc.</li> <li>• It mostly demands high-level technical skills on the programming side.</li> <li>• No centralized management section.</li> </ul>
<b>Test Complete</b>	<ul style="list-style-type: none"> <li>• It provides cross-platform testing, which includes web, desktop and mobile applications.</li> <li>• It has a visualizer to capture screenshots of every single operation.</li> <li>• Minimum of middle-level technical skill is ok in the programming section</li> <li>• It provides good tech support.</li> </ul>	<ul style="list-style-type: none"> <li>• It supports only the Windows platform.</li> <li>• It is not a free software need to buy, but mostly affordable price compares to some other tools.</li> <li>• Lack of checkpoint options to handle debugging a little more easily.</li> </ul>
<b>Katalon</b>	<ul style="list-style-type: none"> <li>• It includes services in web applications, mobile applications and web services.</li> <li>• It provides cross-browser testing.</li> <li>• Test result reports can export as pdf.</li> </ul>	<ul style="list-style-type: none"> <li>• Java and Groovy are the only supporting programming languages.</li> <li>• It cannot be used for desktop applications.</li> </ul>
<b>Watir</b>	<ul style="list-style-type: none"> <li>• It supports multiple browsers.</li> <li>• It is open source and has all code access rights.</li> <li>• Code reusability is high.</li> </ul>	<ul style="list-style-type: none"> <li>• It supports only Ruby, so technical people should learn the Ruby language.</li> <li>• Script execution speed is low.</li> <li>• Using third-party tools for recording</li> </ul>
<b>Qty</b>	<ul style="list-style-type: none"> <li>• High technical knowledge in programming is not mandatory.</li> <li>• Its own built-in IDE.</li> <li>• Test reporting provided with details.</li> <li>• Very good object identification process and is very user friendly.</li> </ul>	<ul style="list-style-type: none"> <li>• Costly compared to other tools.</li> <li>• The execution process is comparatively slow.</li> <li>• Need to renew the license and buy add-ins supported by Qtp.</li> </ul>
<b>LoadRunner</b>	<ul style="list-style-type: none"> <li>• It can stimulate a large number of users concurrently.</li> <li>• It is highly developed and complex.</li> <li>• It reduces the hardware requirement.</li> <li>• It stimulates heavy load and multiple concurrent user's stress.</li> </ul>	<ul style="list-style-type: none"> <li>• It is expensive.</li> <li>• It focuses on performance testing not feasible for functional testing.</li> <li>• It is difficult to set up first.</li> <li>• Its controller section needs the specific windows OS.</li> </ul>

Selenium is a collection of tools developed by a number of professionals in a sequential development manner. Primarily in 2004, it was developed by Jason Huggins (selenium core). But for using selenium core, you need to install the whole application under test, which is inconvenient. To overcome this issue, Hammant invented

a system known as Selenium remote control (Selenium1). Then an engineer named Patrick Light tried to find a solution to minimize the execution time, and it led to the creation of a system known as the Selenium grid. The Selenium team merged Web Driver and Selenium RC to develop Selenium 2, which is more powerful. It is a suite of components: Selenium Driver, Selenium Grid.

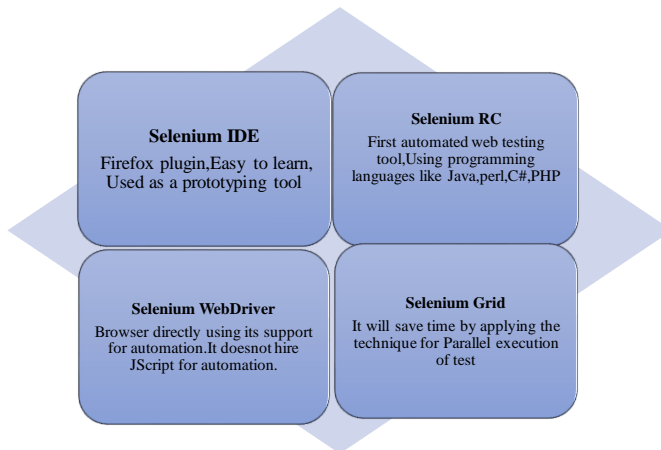


Fig. 5 Selenium components

As the first step, we have selected two demo web applications “tour travo” project and “harmony shop”, a smart shopping application for implementing the selenium testing. Configured test script by preparing web driver code depending on the logic that needed to implement. Included the essential codes that needed to be present in the script like package inclusion, variable instantiation, browser session, actual and expected value, acquisition of page title code etc.

```

1 |
2 | package com.demo.test;
3 |
4 | import org.testng.annotations.Test;
13 |
14 | public class DemoTest {
15 |     public String baseUrl = "http://demo.guru99.com/test/newtours/";
16 |     public WebDriver driver;
17 |
18 |     //Make a folder C:\SeleniumDemo and download the git repository
19 |
20 | @Test
21 |     public void verifyHomepageTitle() throws InterruptedException {
22 |
23 |         System.out.println("launching browser");
24 |         //Browser can be selected as Chrome or firefox. Swap between lines 24 and 25 for the same
25 |         BrowserType browserType = BrowserType.CHROME;
26 |         //BrowserType browserType = BrowserType.FIREFOX;
27 |         //Launch the browser and baseUrl
28 |         driver = LocalDriverManager.getDriver(browserType);
29 |         driver.get(baseUrl);
30 |
31 |         //Wait for 3sec for the page to load
32 |         Thread.sleep(3000);
33 |         String expectedTitle = "Welcome: Mercury Tours";
34 |         String actualTitle = driver.getTitle();
--
    
```

Fig. 6 Selenium Test Script

Then we have selected a desktop application, “MediMind” management Project for analyzing the performance. JMeter is a java application for load and performance testing. So configured the test script using JMeter.

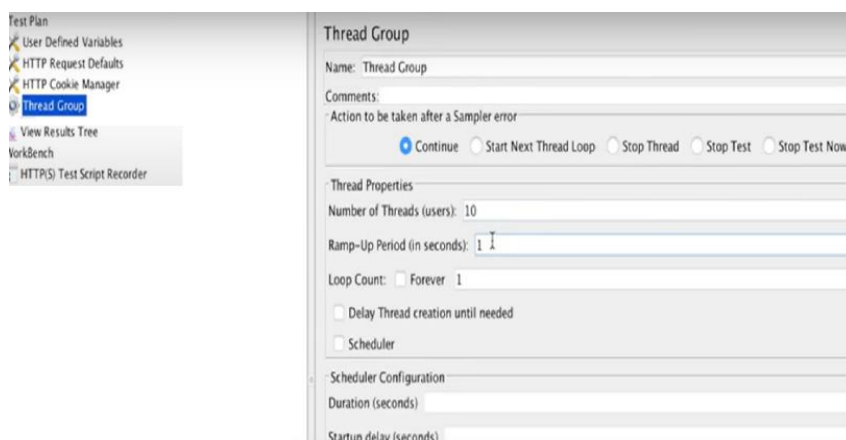


Fig. 7 Interface for JMeter

Here the desktop application is client-side, and users are interacting with the front end of the application, but there should be a back end where the application is interacting with the server or database at the backend. So, at the same time, multiple users interact with the back end. Configured test script to focus interaction of application

request and response to the server and overall performance of backend service.

While executing the test script, measure the CPU and memory utilization using the [18] Linux shell script.

```

3 PRGDIR=`dirname "$PRG"`
4 [ -z "$LT_HOME" ] && LT_HOME=`cd "$PRGDIR/.." ; pwd`
5 cd $LT_HOME
6
7 rm -rf data/cpu*
8 mkdir -p data
9 > data/cpu.csv
10 echo "writing to cpu.csv"
11 echo "TIME_STAMP, Usage%" | tee -a data/cpu.csv
12 while :
    writing to mem.csv
    TIME_STAMP,Memory Usage (MB)
    Available memory in MB: 3866
    15:16:52:1639648012+05:30, 112.11
    15:16:53:1639648013+05:30, 112.11
    15:16:54:1639648014+05:30, 112.11
    15:16:55:1639648015+05:30, 112.11
    15:16:56:1639648016+05:30, 112.11
    15:16:58:1639648018+05:30, 112.11
    15:16:59:1639648019+05:30, 112.11
    writing to cpu.csv
    TIME_STAMP, Usage%
    14:21:28:1639644688+05:30, 2.7
    14:21:29:1639644689+05:30, 2.7
    14:21:30:1639644690+05:30, 2.7
    14:21:32:1639644692+05:30, 2.7
    14:21:33:1639644693+05:30, 2.7
    14:21:34:1639644694+05:30, 2.7
    14:21:35:1639644695+05:30, 2.7
    14:21:36:1639644696+05:30, 2.7
    14:21:37:1639644697+05:30, 2.7
    14:21:39:1639644699+05:30, 2.7
    14:21:40:1639644700+05:30, 2.7
    14:21:41:1639644701+05:30, 2.7
    
```

Fig. 8 Terminal output of the script

ESMT, Energy Spot Measuring Tool, is a software-based tool proposed to estimate CPU and Memory utilization of the Test process. It is a Linux based tool to continuously monitor the Memory and CPU usage of a specific process and, based on this, generate a report which will carry us towards green metric implementation.

that will be saved in the folder chosen in 'Save File'. The button 'Start' is used to initiate running the test script. It is a toggle button that changes to 'Stop' once it is started. And can end the monitoring by clicking the 'Stop'.

For analyzing the role of code on energy utilization, considered test script written by a fresher and an experienced professional. By comparing the generated result, we could understand that role of software on energy utilization over hardware is not negligible.

**VII. RESULT**

Here in this section is a generated report based on results gathered through the experimental procedure. For evaluating the result, inspect the generated data and automated script. The first test run was to evaluate the test script written by the fresher for the tour travo demo project. Further, it is evaluated with the test script written by an experienced professional.

The result generated as CSV files are consolidated for fresher and more experienced with CPU Usage and Memory Utilization, and the graph is plotted.

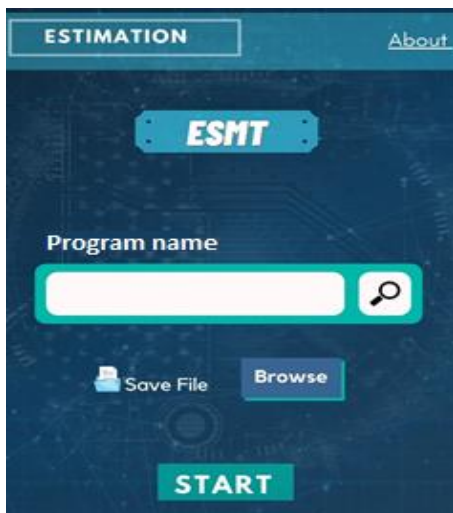
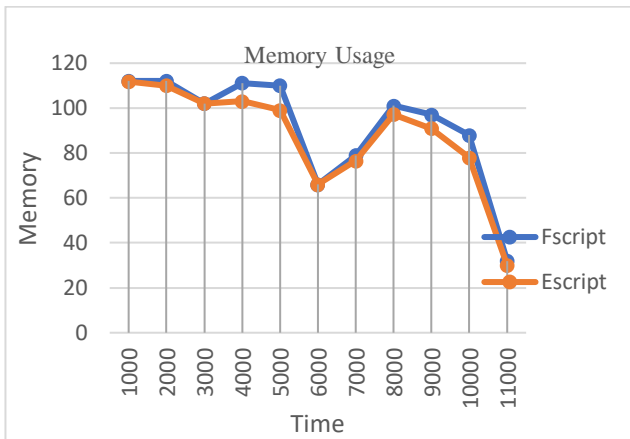
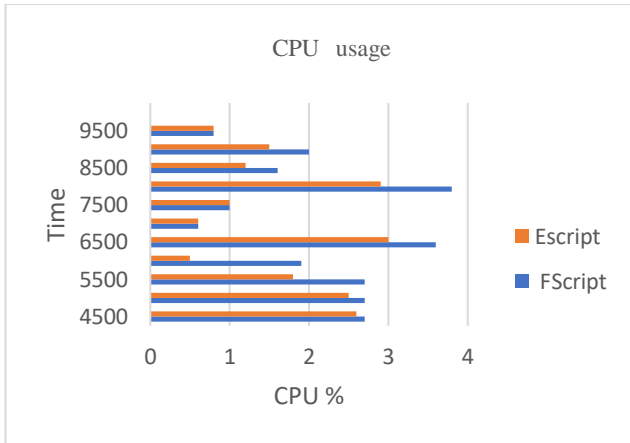


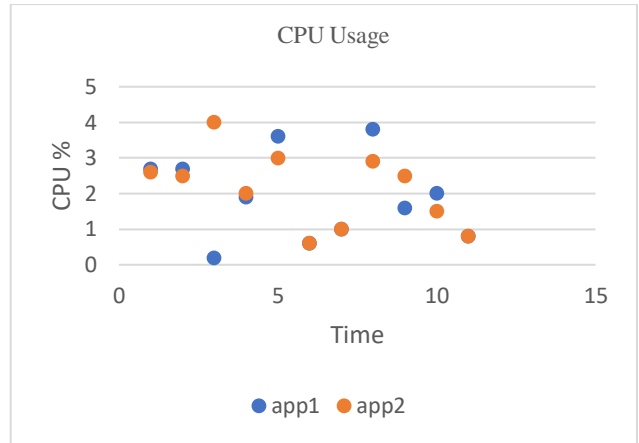
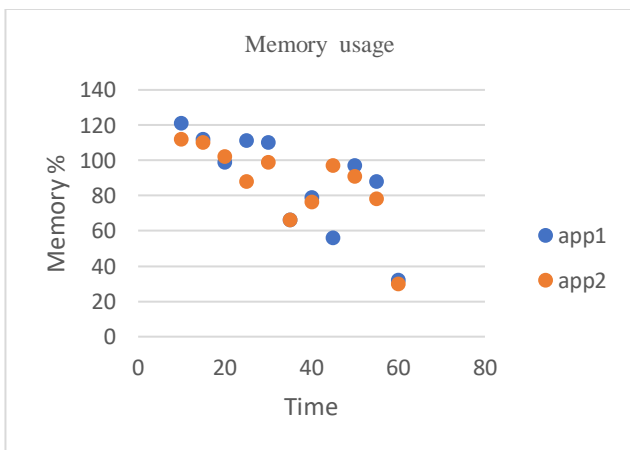
Fig. 9 Interface for ESMT

The User interface is designed using Python. The test script can be chosen using the browse option of 'Program name'. Shell script generates the test result as a CSV file

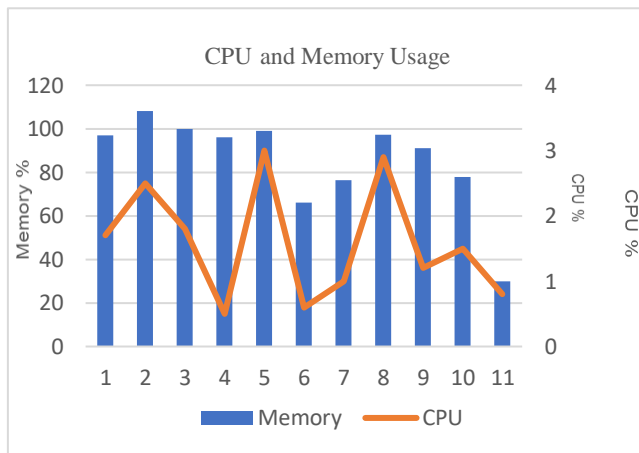


**Fig. 10 Comparison of Memory and CPU usage of Test Script written by Fresher and Experienced Professional for Tour travo project**

We have considered two web applications and one desktop application for monitoring and comparing the energy efficiency while executing the test script. Energy Spot Measuring Tool analyzed the performance while executing the corresponding test script. The CPU based energy consumption represents a major part of energy usage. Peak and continuous memory usage have an influence on energy consumption. In the study, energy, time and memory are the optimization aspiration.



**Fig. 11 Energy and Memory graphical data for tour travo and harmony shop**



**Fig. 12 Energy and graphical memory data for MediMind Project**

From the experiment, I analyzed various test scripts and testing tools and identified that each has a different impact on energy consumption, time and memory.

**VIII. CONCLUSION AND FUTURE WORK**

Since now run through green computing has become essential for environmental sustainability. Each one can confer their own part to reduce global warming. A secure environmental development in the software industry and the society which is utilizing these applications should have awareness about it.

Nowadays, minimizing the environmental impact on software development has an important role. The energy-efficient software needs certain metrics to measure software energy usage and need to minimize it. This tool aims at energy transparency during the testing phase of the Software Development Life Cycle. The future of this work is in extending the scope of the tool by detailing the hoggish energy area of the test script.

## REFERENCES

- [1] Devi, J., Bhatia, K., & Sharma, R., A Study on Functioning of Selenium Automation Testing Structure. *International Journal of Advanced Research in Computer Science and Software Engineering*, 7(5) (2014) 855-862.
- [2] Kaur, P., & Agnihotri, M., Efficient Variable Neighbourhood Search Performance Based Joint Optimization Task Allocation for the Multicore Processor. In 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), (2016) 745-751. IEEE.
- [3] Kannan, M., & Lokeshwari, K., Comparison of Software Testing Tools with Respect to Tools and Technical Related Parameters. *International Journal of Advanced Research in Computer Science*, 8(9) (2017) .
- [4] Hemanandhini, I. G., & Ranjani, C., A Study On Various Techniques for Energy Conservation in Data Centers for Green Computing. *International Journal of Engineering Trends and Technology (IJETT)*, 46 (2017) .
- [5] Brown, D. J., & Reams, C., Toward Energy-Efficient Computing. *Communications of The ACM*, 53(3) (2010) , 50-58.
- [6] Abhishek, D. S., Anusha, V., Bheemappa, C. B., Vijaykumar, D., & Sheela, S. V. *Green Software*.
- [7] Gelenbe, E., & Caseau, Y., The Impact of Information Technology on Energy Consumption and Carbon Emissions. *Ubiquity*, (2015) 1-15.
- [8] Johann, T., Dick, M., Naumann, S., & Kern, E., How to Measure Energy-Efficiency of Software: Metrics and Measurement Results. In 2012 First International Workshop on Green and Sustainable Software (GREENS) , (2012) 51-54. IEEE.
- [9] Hähnel, M., Döbel, B., Völp, M., & Härtig, H., Measuring Energy Consumption for Short Code Paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review*, 40(3) (2012) 13-17.
- [10] Hilty, L. M., Arnfalk, P., Erdmann, L., Goodman, J., Lehmann, M., & Wäger, P. A., The Relevance of Information and Communication Technologies for Environmental Sustainability–A Prospective Simulation Study. *Environmental Modelling & Software*, 21(11) (2006) 1618-1629.
- [11] Hilty, L. M., Aebischer, B., & Rizzoli, A. E., Modelling And Evaluating the Sustainability of Smart Solutions. *Environmental Modelling & Software*, 56 (2014) 1-5.
- [12] Aebischer, B., & Hilty, L. M., The Energy Demand Of ICT: A Historical Perspective and Current Methodological Challenges. in *ICT Innovations for Sustainability*, (2015) 71-103.
- [13] Nixon, J. S., & Devaraj, A. F. S. *Green Computing: Awareness, Current Issues and Best Practices*.
- [14] Kern, E., Dick, M., Naumann, S., Guldner, A., & Johann, T., Green Software and Green Software Engineering–Definitions, Measurements, and Quality Aspects. In *First International Conference on Information and Communication Technologies For Sustainability (ICT4S2013)*, ETH Zurich , (2013b) 87-9.
- [15] Verdecchia, Roberto, Patricia Lago, Christof Ebert, and Carol De Vries. *Green IT and Green Software*, IEEE Software , 30(6) (2021) 7-15.
- [16] Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J. P., & Saraiva, J., Spelling Out Energy Leaks: Aiding Developers to Locate Energy-Inefficient Code. *Journal of Systems and Software*, 161 (2020) 110463.
- [17] Bangash, A. A., Sahar, H., & Beg, M. O., A Methodology For Relating Software Structure with Energy Consumption. In 2017 IEEE 17th International Working Conference on Source Code Analysis and Manipulation (SCAM) , (2017) 111-120. IEEE.
- [18] Bourdon, A., Noureddine, A., Rouvoy, R., & Seinturier, L., Linux: Understanding Process-Level Power Consumption. in *Green Computing Middleware (GCM'2011)*.
- [19] Verdecchia, R., Guldner, A., Becker, Y., & Kern, E., Code-Level Energy Hotspot Localization Via Naive Spectrum Based Testing. In *Advances and New Trends in Environmental Informatics*, (2018)111-130. Springer, Cham.
- [20] Pang, C., Hindle, A., Adams, B., & Hassan, A. E., What Do Programmers Know About Software Energy Consumption?. *IEEE Software*, 33(3) (2015) 83-89.
- [21] Hindle, A., Green Mining: A Methodology of Relating Software Change and Configuration to Power Consumption. *Empirical Software Engineering*, 20(2) (2015) 374-409.
- [22] Steigerwald, B., & Agrawal, A., *Developing Green Software*. Intel White Paper, 9 (2011).
- [23] Mahmoud, S. S., & Ahmad, I., A Green Model for Sustainable Software Engineering. *International Journal of Software Engineering and its Applications*, 7(4) (2013) 55-74.
- [24] Verdecchia, R., Lago, P., Ebert, C., & De Vries, C., Green IT And Green Software. *IEEE Software*, 38(6) (2021) 7-15.
- [25] Johann, T., Dick, M., Naumann, S., & Kern, E., How to Measure Energy-Efficiency of Software: Metrics and Measurement Results. In 2012 First International Workshop on Green and Sustainable Software (GREENS), (2012) 51-54. IEEE.