International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 9 Issue 8 ǁ 2021 ǁ PP. 12-19

Green Software

Abhishek D S, Anusha V, Bheemappa, Chaitra B R, Mallesha Holeyache, Vijaykumar, Dr. Sheela S V
Department of Information Science and Engineering.
BMS College of Engineering, Bangalore.

ABSTRACT
Building efficient software is one of the major concerns for all the companies. The software must be environmentally sustainable and should have minimal impact on the environment. Green software is one such software which can be built more efficiently with minimal impact. Though software products are insignificant goods, their use can bring about remarkable materials. Software is basic for reducing the use of natural resources in IT systems. While Green IT has found success in hardware efficiency in the previous years. It Is time for focusing on efficient resources in software.
Green software must achieve a various life cycle to complete its product namely- Design, implementation and usage. The present report focuses on green software and its lifecycle, green software metrics- refers to measuring the efficiency of the green software and green software practices- finding the best practices for efficiency of the software in industries and few impacts of green software.
Keywords*: Green software, sustainability, measurement, energy efficiency, green design, green metrics, practices.*

Date of Submission: 14-08-2021 Date of acceptance: 29-08-2021

I. INTRODUCTION

Nowadays software is ubiquitous everywhere, Minimal environment impact and sustainability are the important factors in development of software. When it comes to energy efficiency software plays an important role. Green infrastructure is incomplete without software. In previous years Green IT has focused on hardware efficiency to reduce power consumption. Any device that runs with software consumes energy, hence one to understand the energy consumption in these devices. Green software is built to ensure the minimal impact on the environment and provide sustainable energy. Software developers must know the techniques to save energy consumption allow a way for energy consumption while others do not. Few energy saving techniques include the computational efficiency, data efficiency, idle efficiency and the context awareness. Software developers have choices to choose one among these techniques and work on it. It is important to understand the impacts of power consumption to improve the energy efficiency of the software. Many tools are available to measure the power consumption, but these tools although they provide high-level power consumption they don't completely provide the grind details. Development of green software includes various processes like, design, implementation and usability. The aim of developing a green software is to provide efficient coding and reduce power consumption. Sustainability is one major factor while talking about the software. Recent study shows that many systems cannot support for higher resources demanded by software. Though many computers are in working condition, their software requirements make them unusable. Hence, sustainable software must be developed considering positive and negative impacts on the environment. Few attributes are provided to evaluate a sustainability of the software, Like the software must be able to make changes quickly, the software developed must on every system that it is installed to and the software must be easy to deploy and use. Other attributes include performance, usability and accessibility. Quantifying these attributes provides a sustainability metric which is used for measuring the efficiency of the software. This research focuses on how to obtain sustainability metrics and also few analysis on platforms. Green practices in software designing which is used to get knowledge on greening to get more environment friendly software. There are many such practices improving the efficiency of the software. Few important among those will be explained in detail later in this paper. This research focuses on the development of the green software and its life cycle. Sustainability metrics for the software and few green software practices. Section II presents the literature survey done on green software. Section III focuses on development of green software and its methodologies. Section IV explains the metrics used for measuring the software. Section V presents the best practices for green software followed by industries.

II. LITERATURE SURVEY

Nazakat Ali et al. presents that cyber-physical systems have changed the way of thinking of engineers in the development of software [1]. This paper addresses that energy consumption is one of the important factors in terms of software performance. Green software development addresses this problem. Green software is built to reduce energy consumption. However software has to be on hardware, dealing with hardware systems is also important. A number of researches have been conducted to measure the energy consumption of the software system. This paper has proposed an energy-aware framework that applies code-refactoring and dynamic adaptation strategies to develop a green software.

Ayse Bener et al. explains how green software can be promoted after an efficient software is built [2]. These types of studies help to raise the awareness among the people for greener software. Later the study explains how important it is to consider the greenness of the software while in the design phase. As energy efficiency is important in these types of software, consideration of greenness also matters. The paper has also focused on the cost of the green software and the evolution of green software and its uses

Saeed Ullah Jan et. The idea is to develop a sustainable green software [3]. This article explains the factors that are available to develop a sustainable and green software for customer satisfaction. Efficient utilisation of time and computing resources is an important factor in the GSD environment. This factor helps in avoiding defects and adopting re-usability factors. Further this paper also explains Continuous validation, polymorphic design and rich communication and collaboration, Minimal documentation.

Eva Kern et al. states the two major fields of sustainability are Green IT and Green By IT [4]. This paper has focused on the reference model of green and sustainable software. The paper provides a short explanation of the reference model and the sub-model in development of software. In addition a model to measure the energy efficiency of the software and an example of measuring results is also described in the paper. Later the paper also gives a clear description of other measurements models for software energy consumptions.

Sanath. S. Shenoy et al. focuses on SDLC- softwares development life cycle of the green software [5]. The paper focuses mainly to the systematic execution and maintenance of software by dividing a development process into requirements, designing, implementation, testing, deployment and maintenance. This paper also discusses the changes in the existing SDLC model and suggests the appropriate steps to lower carbon emissions and power thus helping the industries to move towards a greener , sustainable software development.

Noraini Che Pa et al. presents the green dashboard system for measuring green software design which helps to assist the developers to consider while designing the software [6]. The paper addresses two contribution for the proposed system

1. Green software design metrics which can be used to reduce carbon dioxide emissions.
2. A web based system which helps to measure the sustainability of the software.

Timo Johann et al. explains the existing approaches for measuring the energy consumption of the software namely, Black box and white box measurement. Blach Box measurement is classified as a benchmarking measure of how the entire system performs and individual measurement refers to the concrete performance of the system [7]. As this method has few disadvantages White box measurement is implemented to tell which part of the software has potential to save energy.

Tommi Mikkonen et al. presents the typical industrial application leading to energy saving; the application is driven by an electric motor using a transceiver [8]. The author says that the whole system is guaranteed to run for more than ten years. As energy consumption is important in software development this system produces insignificant energy savings. Over ten years the consumption by system will be 29.5 kWy.

Muhammad Salam, Siffat Ullah khan presents SLR for developing a green and sustainable software the paper presents a list of success factors for developing a software [9]. The categorised factors were green software design and efficient coding, power saving software strategies and low carbon emissions. All these factors may lead vendors to develop a green software with minimal impacts.

Gang Hou et al. proposes a Energy Consumption Time State Transition Matrix as a green software model which can be effectively used for time and energy consumption while developing a software [10]. The model analysis is divided into two types namely,

1. Analysis of Energy Consumption Based on Path
2. Analysis of Energy Consumption Based on Model Checking.

The proposed system uses intelligent sweeping robot software design to show the analysis and design phase in software model. Hence the green evaluation on path analysis helps to increase the efficiency of the software.

Krisztina Erdélyi presents the activities of the software development life cycle in the point of how they contribute to the environment [11]. Firstly green software has to execute effective algorithms or test cases to measure the effective of them. Green software measurement is an

important factor while considering the development of software and also helps in energy consumption. Design phase shows the behaviour of the software in energy consumption.While Implementation the language and IDE's should have an effect to save energy. Finally Maintenance of the software should be more sustainable and have minimal impact on the environment.

Tribid Debbarma et al. explains different green measuring indicators in sustainable software that were rigorously evaluated and examined [12]. And also located the latest research in green metrics as well as other state-of-the-art measurements used in green software development and engineering. Green metrics in the sustainable software domain view energy metrics as one of the greatest measures for attaining sustainability in software, according to the extracted and processed data. Other measurements, such as environmental features and traits, aren't taken into account. Software sustainability is also taken into account in other sectors, such as mobile, network, and datacenters.

Luca Ardito et al. presents Identifying and reworking code patterns that cause high-energy use (code level) or building the programme in such a manner that it can adapt at run time to varied energy demands or the available residual device energy are two complementing ways for developing green software (design level)[13].

1. Refactoring : Guidelines at the Code Level Predictive models incorporate information about both the power-consuming resources (such as the CPU) and the actions that cause their usage (such as disc transfers). As a result, from a developer's viewpoint, the next step is to discover the code patterns that indicate heavy use of certain resources and activities.
2. Self adaption : The basic concept is to give many configurations of the same programme that may be engaged at different times to achieve the greatest balance of functionality and energy consumption.

Bokolo Anthony Jnr et al. presents a model for understanding how software practitioners might use software applications in the software development process to embrace sustainable practices [14]. When software practitioners use software that reduces power use and CO2 emissions, they are practising sustainable practises. The adoption of sustainable practices by software practitioners is unquestionably helpful to humankind. The authors also explain how Green practices might be distributed when software practitioners employ software applications in software businesses.

Ana Carolina Moises et al. presents the academy's recommendations for sustainable software engineering have been implemented throughout the industry. Using a thorough literature analysis, 170 behaviours were discovered, 70 of which were linked to energy consumption practises that might be used during software development [15]. The findings show that those practises arose from grounded theory, are part of SWEBOK areas, and may be used in the workplace. The issue of sustainable software engineering has gotten a lot of attention recently, especially when it comes to determining the advantages of developing a sustainable software product via a sustainable software development method. Only a few studies covered the procedures used in the sector, according to the SLR offered in this study.

III. GREEN SOFTWARE METHODOLOGY

Green software must go through different phases of the software life cycle.

Fig. 1 - Green Software Development Model

1. REQUIREMENT

The software's shelf life must be considered at the Requirement phase. The shelf life of software refers to the time it takes for it to help society reach a specific objective. If a project's shelf life is determined, it will be easier to gather requirements in both the present and the future. User Interface requirements are also captured as part of the requirement gathering process.

When gadgets, displays, or computers are not in use, the requirement collection process should also involve turning them off or switching them to low power mode. This will assist in reducing the system's unneeded power use.

2. DESIGN

The major objective of the Design phase should be to achieve design simplicity. The system's design should be as straightforward as feasible. Complex designs may necessitate re-design and increased documentation effort utilizing computers and gadgets, resulting in increased power use, fuel use from travel, and other resource use. If the design is complicated, make an attempt to simplify it by modularizing it. Another crucial component in making a design sustainable is re-use. Reusable design can help you avoid having to re-implement components that are already in place. Extending re-usability may be accomplished through the usage of plug-in architecture, which is widely utilised to provide both reusability and extensibility in software systems.

3. IMPLEMENTATION

The implementation phase is critical since it is here that the requirements and design are combined to create a functional system. The initial stage in the implementation phase is generally to adhere to the organization's software development process's best practices. The usage of hardware-specific Application Programming Interfaces (APIs) should be avoided for long-term software implementation. These APIs tend to use more resources in terms of hardware, memory, disc space, and processing cycles, among other things. As a result, the programme is non-functional and unable to scale with current resources or older technology. One way to avoid a scenario like this is to profile resource-consuming APIs and use them sparingly.

4. *TESTING*

A software system's correctness is tested and validated during the testing process. At a more granular level, unit tests play a critical role in developing defect-free software. The fewer the flaws, the less software modification is necessary.

Automated testing should be promoted since it reduces manual mistakes. They also stress the reuse of test cases and the standardization of the testing procedure. This increases not just testing accuracy, but also productivity and minimizes the amount of power required by additional resources in the manual testing process.

5. DEPLOYMENT OR INSTALLATION

The real programme is prepared for installation in the productions environment during the deployment phase. The size of an installation package has a significant impact; the larger the installation, the longer it takes to install in the production environment. Standard data compression techniques should be used to reduce the size of a deployment or installation packages.

CDs and DVDs, as well as any other disposable media, are a major of source of e-Waste since they are programmed to be utilised based on factors such as a set number of instals, serial key on based installation, machine or system based installation. An alternative option would be to employ online licence verification based installation to dramatically reduce the amount of e-Waste generated by discarded media.

6. MAINTENANCE

When the majority of the features have been implemented and the programme has been running well in a production environment for time, it enters the maintenance phase. Fixing issues, fine-tuning the system, integrating new features, and so on are all part of this phase.

All of the previous phases of software development have an influence on this phase. Software maintenance necessitates the use of documentation. To avoid the consumption of paper, documentation should be in electronic format.

IV. GREEN SOFTWARE METRICS

Timo Johann et al. proposes the existing approaches for measuring the energy consumption of the software namely, Black box and white box measurement [7]. Green Metrics assumes that the functional component of an application has been chosen, and that the application structure, as well as the specific hardware and software infrastructure architecture, are all important.

We consider not just data for monitoring the program's execution and environment, but also metrics for evaluating the application.

Green Metrics are organised into the four clusters shown below..

1. IT Resource Usage Metrics

The quantity of energy spent by an application is determined by the nature of the programme as well as the system setup of the run-time environment. It shows how resource utilisation differs depending on whether the application is processor-intensive, data-intensive, or hybrid. As a result, an application's energy usage is determined by how well it utilizes its resources.

The list of resources contains CPUs in compute nodes and application servers, primary memory, storage devices, and I/O resources. By mapping the functions of the application into resource use, the energy consumption of the architecture running the specified programme can be linked.

All the metrics in this category are provided as percentages.

- CPU Usage Metric:

  This metric is related to a certain application's relative CPU utilization. As a result, the CPU Usage Metric shows how much time an application spends using a processor.

- Storage Usage Metric

  When reading or writing data to permanent storage, the local storage or remote Hard Disk Drives (HDD) are processed. The Storage Usage Metric denotes the total storage utilization percentages on the relevant storage device for data read and write activities.

- I/O Usage Metric

  Local storage or remote Hard Disc Drives (HDD) are used for reading or writing data to permanent storage. The Storage Utilization Metric displays the total storage usage percentages for data read and write activity on the relevant storage device.

- Memory Usage Metric

  This metric refers to the use of the main storage (RAM). The Memory Usage Metric represents the percentage of RAM occupied by specific application operations.

2. Life cycle Metrics

This metric describes an application in terms of the work required to develop it. These measures, on the other hand, support the design steps by allowing developers to set parameters which enable both energy consumption to be monitored while implementing applications.

- Life Cycle Cost Metric

This measurement traces the absolute lifecycle expenses of the process including reasonable demonstrating, investigation, design, advancement, sending, upkeep and evolution costs. These measurements, which depend on software improvement exertion assessment devices like SEER-SEM12, think about things like developer skill, code intricacy, level of abstraction and speculation, reusability and integration rates, prerequisites dependability, and the application's nearness to the organization's center business. For instance, improvement costs are based on engineers' endeavors (face to face/months) identified with the application life cycle. Expenses are caused as a result of these endeavors. This estimation is given in terms of energy saving in cost units. Accordingly this measure permits us to decide the relationship between endeavors needed to overhaul an application for upgrading energy use versus potential energy savings.

- Process Engineering Metric

This measure describes the development style used to create, code, and deliver an application. It takes into account both the quality of the platform used and the quality of the code written. The maturity of the development platform, the tools for developing and documenting the application, and the application engineering approaches employed are examples of the first class of criteria. These characteristics have an impact on energy consumption because they allow developers to create programmes faster, with fewer errors, and with higher efficiency in the code itself,they also allow energy-aware elements to be integrated in them, with monitoring code or performance indicators that identify energy leaks. Indexes of data utilisation, service consumption, branching probabilities, and failure probabilities in the application code are examples of factors of the second type. Net present value, profitability index, internal rate of return, payback time, and capital recovery factor are examples of project indicator measures. This metric is expressed worldwide as

an index derived from the weighted sum of factors of type a) and factors of type b). It is expressed in cost units and can be converted to an energy conservation index (*cost).

- Quality of Service Metric

This metric evaluates the quality measures of an application. It contains the following parameters.

1. Availability: This is the normal pace of accessibility of an assistance s, addressed by the likelihood P(s) that a solicitation is effectively satisfied inside a normal time. Bigger upsides of accessibility require more resources to guarantee the execution in the framework, along these lines consuming more energy. This measurement is given as a Percentage as Availability = Total number of successful assistance demands/Total number of administration requests.

2.Throughput: The quantity of administration demands served at a given time span. This measurement is given as a index. Reaction Time: It is the total time took to a help S to deal with client demands . Reaction time is the amount of the preparing time(Tp) and the transmission time (Tt) characterized as,

$$Tr(s) = Tp(s) + Tt(s)$$

• Cycle Time: The normal time taken by an assistance S, from the hour of conjuring to the season of consummation, delays. Which are expressed as follows,

$$Tp(s) = (Tcomp(s) - Tinv(s)) + \Sigma\ Tdelay.$$

Where,

Tcomp=T completion,

Tinv = T invocation.

3. Reliability: It is a chance of a service unalters functional to reach the required function within a specified time. This metric is given as an index defined as,Which implies,

$$r = e\text{^}(-M)$$

Where, M = number of failures.

4. Recoverability- It is the ability of a help to reestablish the ordinary execution after a disappointment inside a given time span. This measurement is given as a list. it is a QoS metric that can be estimated from logs utilizing measure mining.

5. Workload -this contains the sort and rate of demands which are shipped off the framework included with the execution of programming bundles and in-house application programs. This is given as a record.

6. Application Performance Metric- This metric allows for the estimation of energy utilization per figuring unit. This measurement permits specifically to look at changed IT administration focuses in a transparent mode. Also, this is given in the Computation Unit/KWh.

3 Energy Impact Metric

These measurements essentially identify with the effect of the application lifecycle on the climate, thinking about power, power supply, devoured material, and outflows.

- Consumables Metric

This record calculates the amount of consumable cons created by the programme during its work process. This measurement also takes into account the records generated by the application at both turn of ievents iand run time. Furthermore, the volume of information is displayed by the application. This measurement is provided in icost units.System Power Usage Metric

- System power

Framework power use alludes to the force utilization of a framework executing an application, having the power utilization of the PC framework also as the energy utilization of the related office and foundation, like the cooling framework, network gadgets, etc. This measurement is communicated in KWh.

This metric defines the index of carbon emissions which are being emitted by transportation, logistics, etc. And this is needed for the execution of the according services. This metric is provided in CO_2 units.

4. Organizational Factor Metric

- The measurements in this group consider the human elements associated with running and overseeing applications, just as normalization and consistence to decide when it is reasonable to put resources into more energy-proficient frameworks .On the other hand, requiring more noteworthy starting interests in creating and running less energy-squandering applications.

- Human Resources Metric in this bunch, which estimates the expense of HR that associated with keeping up and regulating an application..
- Compliance Metric, which estimates the exertion needed to agree with enactment or potentially consortium approaches

OVERVIEW OF GREEN METRIC

The cost unit are proportional to the amount of energy consumed.

Metric	Unit
IT Resource Usage Metrics	
CPU	Percentage
Storage	Percentage
Input/output	Percentage
Memory	Percentage
Lifecycle Metrics	
Lifecycle Cost	Cost
Process of Engineering	Index
Quality of Services	QoS Levels
Energy Impact Metrics	
Consumables	Cost
System Power	KWh
Supply Chain	CO2 units
Organizational Factors Metrics	
Human Resource usage	Cost
Compliance	Cost

Table. 1- Metrics and its Units.

Here, Percentage calculated for 100% and Cost - It is expression of cost units and can be converted to an energy conservation index.

V. GREEN SOFTWARE PRACTICES

There are best practices for increasing the software efficiency Few listed below are few from both industry and academic literature.

1. Efficient data traffic

A smaller amount of data is sent through the communication line which leads to leaving more idle time for all devices resulting in energy saving.

2. Decrease algorithmic complexity

Though all algorithms are used for the same functionality, their performance differs significantly. The amount of data sent or the amount of time spent on computing can be different from one another.

3. Power Down Peripherals

When peripherals are not in use they can be shut down. Hence, shutting down the peripherals lead to conservation of energy.

4. Load Balancing

Two processing units can get the same work done as a single clocked processor which doesn't produce much heat in turn which saves energy.

5. Optimal use of peripherals

Using peripherals in an effective way also saves energy. Utilizing specific properties and characteristics is one effective way of increasing efficiency. A device driver is often used for peripheral behavior, but using peripherals optimally is limited to application software.

VI. CONCLUSION

This paper studies green software and its methodologies. Every software is built in different phases, each phase has its own way of developing a software, the working of phase is mentioned in this paper. Measuring the energy efficiency of the green software is also important after development. There are various metrics and their attributes explained in the paper. At last green practices which are followed by few industries though all those practices have drawbacks they can be used in increasing the efficiency of the software by energy consumption. Many researchers have provided various ways of increasing the efficiency of the software. Future works include in with drawbacks.

REFERENCES

[1]. G. Sissa, "Green software," The European Journal for the Informatics Professional, vol. II, pp. 53-63,2010.
[2]. Timo Johann, Markus Dick, Stefan Naumann, Eva Kern., "How to Measure Energy-Efficiency of Software: Metrics and Measurement Results", First International Workshop on Green and Sustainable Software (GREENS), 2012.
[3]. P. Bozzelli, Q. Gu, and P. Lago, "A systematic literature review on green software metrics," VU University Amsterdam, 2013.
[4]. S. Agarwal, A. Nath, and D. Chowdhury, "Sustainable Approaches and Good Practices in Green Software Engineering"., International Journal of Research & Reviews in Computer Science, vol. 3,2012.
[5]. Biswajith Saha, "Green Computing", International journal of computer science and engineering, May 2018.
[6]. David Gil, jose Luis fernandez-Aleman, "The effects of Green software : A study of impact factors on the correctness of software", September 2018.
[7]. Nazakat Ali, Jang-Eui Hoing, "Introducing Green software development to software product lines",KSII The 13th Asia Pacific International Conference on Information Science and Technology(APIC-IST), 2018.
[8]. Eva Kern, Stefan Nauman, Achim Guldner, Timo Johann, "Green software and green software engineering - definitions, measurements and quality aspects", Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, 2013.
[9]. Giuseppe Lami Luigi Buglione Fabrizio Fabbrini "Derivation of Green Metrics for Software", International Conference on software process improvement and capabilities, 2013.
[10]. Tribid Debbarma; K Chandrasekaran, "Green measurements metrics and towards a sustainable software : systematic literature review", International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2016.
[11]. Márcio Welter; Fabiane Barreto Vavassori Benitti; Marcello Thiry, "Green metrics to software development organizations: A systematic mapping", XL Latin American Computing Conference (CLEI), 2014.
[12]. S Ergasheva, I Khomyakov, A Kruglov, G Succi, "Metrics of energy consumption in software systems: a systematic literature review", 3rd International Conference on Power and Energy Engineering , 2019 .
[13]. Lami G., Buglione L., Fabbrini F., "Derivation of green metrics for software", 13th International SPICE Conference, 2012
[14]. Noraini Che Pa, Faizal Karim, Sa'adah Hassan., "DashBoard system for measuring green software metrics", 3rd International Conference on Science in Information Technology (ICSITech) , 2017.