International Journal of Scientific & Engineering Research, Volume 12, Issue 6, June-2021
ISSN 2229-5518

Software Development Life Cycle for Environmental Sustainability

Likhit J Jain[1], Nischala Prasad[1], Nishitha B N[1], Guru Darshan[1], Hemanth Kumar[1], Kushagra Galundia[1], Sheela S V[2]

[1]Student, Department of Information Science and Engineering, BMS College of Engineering, Bangalore, India.
[2]Professor, Department of Information Science and Engineering, BMS College of Engineering, Bangalore, India.

Abstract: The Software Industry is unequivocally the backbone of today's digital society. Since the late 90s, the software industry has generated enormous amounts of wealth. The benefits of technology are seen daily. However, this industry also plays a huge part in the ever-increasing carbon footprint as its development is highly energy-intensive. For instance, the carbon footprint of a single neural network model during training can emit as much as five cars' carbon footprint in its lifetime. Also, there is an exponential increase in the computational power required to run AI models. The industry has to focus on the environmental sustainability of software as an integral part. Along with traditional parameters like scalability and security, the software's energy efficiency must also be judged in performance reviews and the inclusion of green practices and targets. Computer hardware, starting from manufacturing to its disposal, leads to pollution of air, water and soil. This affects the present and the future. This paper discusses how the software industry can take steps to be more green and sustainable in its practices. We propose a Green Software Development Life Cycle (SDLC) which can make software products more sustainable. By focusing on Green IT and actively taking part in sustainability efforts, the rewards will outweigh the challenges in the long run.

Keywords: Environmental Sustainability, Software, Software Industry, Software Development Life Cycle, Green IT.

I. INTRODUCTION

Computer Software plays an important role in today's society and it is being utilized by all industries and fields, be it the healthcare industry or the financial industry. It's close to impossible to point out an area where it is not used. It has generally made our lives easier and massively increased human productivity. However, this comes with a hefty price, the price the environment pays. It has worsened a lot of the environmental problems. The use of energy in the form of electricity is the element of software that is most significant to global warming. The development of software can extract humongous amounts of energy: an AI model on a publicly available dataset of flowers' classification achieved an accuracy of 96.17% with 964J of energy. 2815J of energy was consumed in the next 1.74% increase in accuracy, and around 400% more energy was consumed for the last 0.08% increase in accuracy. Now consider the same example in the context of the bigger picture. Without hardware, the software has no utility. Because software and hardware are so interconnected, the software sector might be blamed for hardware's environmental effects.

Green IT can be defined as "the systematic application of practices that enable the minimization of the environmental impact of IT and allow for company-wide emission reductions based on technological innovations". It is the study and practice of utilising computers and resources in an efficient, responsible, and environmentally friendly manner, as they use a lot of valuable natural resources during their lifetime and can have long-term detrimental effects on the environment. According to UNESCO, "Sustainability is using the resources in a way that does not compromise the environment or deplete the materials for future generations". Sustainable engineering cannot be the sole responsibility of Environmental Engineering, but an approach from all aspects of engineering fields are required. Sustainability in engineering must be incorporated to advance the quality of life for mankind. The two big fields of sustainability in IT are Green IT and Green by IT. Green IT considers resources and energy consumption and Greening By IT has the objective of reducing the environmental impact on the whole (including other fields) utilizing ICT.

The operation systems, their runtime environment, and the software product itself all require a significant amount of energy. The runtime factors that determine the power wastage of processors for computing-intensive workloads in Multicore computer systems are dynamic, short-circuit, and leakage power. The bulk of power on-chip is consumed by circuit-level components such as the clock tree, registers, control and data channel logic circuits, and memory. Choosing

International Journal of Scientific & Engineering Research, Volume 12, Issue 6, June-2021
ISSN 2229-5518

desktops over laptops (laptops can use up to 80% less energy than a desktop); using unnecessarily large monitor sizes (a lot of employees' work does not require large monitors); and failing to conduct regular energy audits are some of the ways that software companies consume more energy than is required.

The software industry has three effects on the environment -

- Carbon emissions from manufacture, usage, and disposal;
- Indirectly beneficial or negative emission consequences; and
- Influencing behaviors and preferences

The software industry is responsible for around 730 million tonnes CO2-eq of total carbon footprint, or about 1.4% of world greenhouse gas emissions.This includes not just the power required by the system equipment during operation, but also all other parts of the system's life cycle, such as network, data centre, phone, computer, and other user equipment manufacturing. Construction of ICT-related structures, as well as employee travel and transportation, are also covered in the data.

By enhancing software development and the SDLC process, the software industry may become more sustainable and ecologically friendly. Previously, software development centred on the environment received little attention. The objectives were primarily focused on meeting functional and performance requirements, while disregarding other critical factors such as energy usage, effective exploitation of IT resources, IT operating expenses, and overall negative environmental impact. However, due to the increased focus on IT in recent years, there is a compelling need to address these previously ignored elements. Sustainability and the environment must be kept in mind by all software companies. Positive and negative impacts over time should be continuously assessed, documented and should be further optimized by the companies. This paper focuses on how the software industry can make green and environmentally friendly decisions.

II. LITERATURE SURVEY

The software industry impacts the social, economic, and environmental aspects of a country. These impacts can be both positive or negative, direct or indirect. Therefore, we require more sustainable and innovative forms of production within the industry to deal with environmental challenges.

A proposal for a general software development process modification that may incorporate sustainability issues into software development processes has been put forth [1]. To accomplish sustainable software development, the model includes supporting tools, guidelines, teaching material, and many artefacts such as sustainability reviews and previews, continuing process evaluations, and a sustainability retrospective. Application of this model to an agile process is easy but faces a problem when it comes to complex processes. “The Greensoft Model” , has been regarded as a reference model for green and sustainable software engineering has been described. [2], which measures the energy efficiency of a software system. By using an HTML cache, approximately 8.6% of electricity consumed whilst generating 19% of reserved capacity is saved. An alternative model on energy consumption and cloud computing activities was used to study the empirical link between energy consumption and cloud computational activities, as well as system performance. [3].

Software companies also make use of immense hardware. Software and hardware together lead to a lot of power consumption and heating issues, adding an extra dimension when it comes to comparing competing software architectures. Two software architectures are compared with an emphasis on energy consumption: Half Synchronous/Half Asynchronous Concurrency Architecture (HS/HA), and the Leader/Followers Concurrency Architecture [4]. On further analysis of these architectures, it is seen that they produce almost similar throughput but the one which consumes less energy is the more optimal architecture, namely the HS/HA Architecture. A two-level power model has been suggested for estimating per-core power dissipation on “Chip Multiprocessors (CMP)” utilising just one “Performance Monitoring Counter (PMC)” and frequency information from CPUs [5], which aims to simplify the task for software developers, by developing a “Software Power Analyser (SPAN)” based model, which identifies the power behaviour of source code. The overall error percentage is under 3% after running different benchmarks on SPAN. Research on the development of environmentally friendly software can be performed by modifying the Software Development Life cycle (SDLC) to include environment-friendly decision making and processes [6]. Comparison of power consumption of computers and monitors manufactured pre-2000 and post-2000 was done showing how decisions taken in the software industry impact environmental conditions. Some modifications to the current software development are suggested that positively impacts the environment and

International Journal of Scientific & Engineering Research, Volume 12, Issue 6, June-2021
ISSN 2229-5518

helps the organization to build environmentally friendly software development centres. The biggest absorber of power, according to analysis, is the processor, and is the sole constituent that is majorly affected by the load, whereas all of the other components utilise roughly the same amount of power both when loaded and when idle. [7]. These results suggest that the processor should be the main focus of energy consumption optimization research.

To address problems of carbon emissions and power consumption of Information Technology (IT) companies, the environmental impact of software services are measured leading to awareness about the energy consumption of software units and software optimization can be targeted as needed. The three main problem areas are - process awareness, service awareness and people awareness - and a service-oriented approach is proposed to address this problem. The influence of enhancing software functionality and exploiting older systems on the environment has been examined [9], which delves into the tradeoffs between enhancing software functionality and lowering software energy usage. Data compression is the major focus, which is a characteristic that reduces the amount of I/O operations while enhancing CPU usage.

Greener initiatives are in the works. IT should incorporate cutting-edge, energy-efficient electronic gadgets and services, as well as every possible energy-saving option [10]. Sustainable IT encompasses a core set of elements that characterise SDD design, such as power management, virtualization, cooling technologies, recycling, waste disposal, and IT infrastructure optimization [11]. The focus has been on long-term IT projects to improve data efficiency. As a result, infrastructure, power, workload management, product design, virtualization, and cloud computing activities have received more strategic and tactical attention. The second wave of long-term IT services is just getting started, and it will be significantly more difficult to develop and deploy. It comprises determining the IT department's role in an organization's broader Corporate Social Responsibility (CSR) strategy, as well as researching current trends, issues, and future implications of Green Computing [12]. The most important activities in green computing are equipment recycling, paper consumption reduction, virtualisation, cloud computing, power management, and green production. Several major research papers relating to green computing were surveyed, which highlighted the relevance of sustainable development [13]. Green Computing research is being carried out in: Energy Consumption; E-Waste Recycling; Data Centre Consolidation and Optimization; Virtualization; IT Products and Eco-labelling. An optimization framework for managing green data centres has been proposed using multilevel energy reduction techniques. An methodology was used to evaluate the environmental effect of an individual's usage of Information and Communication Technology (ICT) over a year [14]. Calculating the energy and data consumption of an average user's use of a typical gadget, as well as the equivalent energy usage (and hence CO2 generated) at each stage in the data chain, we may calculate the embodied carbon of an average device.

The carbon footprint of the ICT industry was forecasted for the year 2020 [15]. According to projections, the software sector's carbon footprint (measured in CO2 emissions) will increase somewhat (at approximately 4% a year). The entire carbon footprint for the software business was reported based on a result of the reference research as shown in Figure 1. A methodology is described to measure software's carbon footprint by establishing a model enterprise [16]. On this basis, a concentration on the development period calculates the carbon footprint. The resultant carbon footprint is transformed per person per month into the kg of CO_2 emissions. There are two major parts to making software greener: activities to make the manufacturing process greener and actions to make the software product itself greener. Because one of the key goals of Green IT is to create environmentally friendly software, the approaches given here can assist developers and designers without requiring them to change their software engineering practises on a whole.

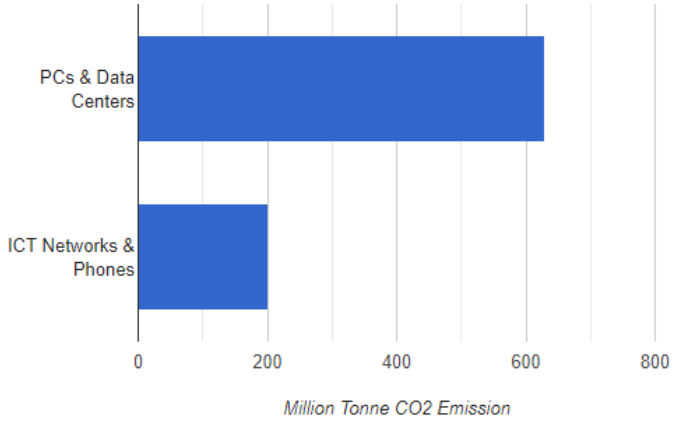

Fig. 1 Carbon Footprint of the ICT industry in 2020 in terms of CO_2 emissions.

III. PROPOSED FRAMEWORK

We attempt to rethink the Software Development Life Cycle (SDLC) in this paper by adjusting it to fit

International Journal of Scientific & Engineering Research, Volume 12, Issue 6, June-2021
ISSN 2229-5518

sustainability and include some ideas in each of the phases: Requirement Engineering, Design, Implementation, Testing, Deployment, Maintenance, and Disposal.

A. *Requirements Engineering:* The very first stage in building any software is requirements engineering. The main goal is to understand the problem that the system is trying to solve and deduce the most suitable methodology to obtain the solution. At this stage the proposed system should be examined to determine if the problem is being solved. If not, then the most energy efficient way forward is to not develop the product itself. It's also important to consider the product's shelf life, which aids in the development of software that can function on legacy hardware, since new hardware should be avoided when eliciting and understanding requirements. As a large quantity of electricity, paper, and bespoke hardware is utilised for this purpose, it cannot be deemed a good technique because the prototypes are only developed to understand the needs and then discarded. As a result, creating a reusable prototype that can be reused in the future is encouraged.

B. *Design:* The product's design should be maintained as simple as possible. Complex designs may demand re-design and greater documentation effort utilising computers and devices, resulting in greater power consumption and other resource use. If the design is complex, make an attempt to modularize it in order to lessen the complexity. In addition, repeated design changes can be costly in terms of time and resources utilised in the other phases of the SDLC. A design that is constantly changing may not be environmentally beneficial, as it could result in a lot of paperwork, usage of tools to develop or update the design, impact both old and new hardware and software, increased power usage, and so on. Based on the application, programming language, programmers should build efficient algorithms by developing a compact design of codes and data structures.

C. *Testing:* Performance and resource profiling should be emphasised as a required component in the testing process. Without them, additional hardware, processor cycles, and memory may be required. It might not be able to work properly with current or older systems that have little or moderate hardware resources. Automated testing should be encouraged because it reduces manual errors. They also promote test case reuse and testing process standardisation, which improves not just testing accuracy, but also productivity and minimises the amount of power consumed by additional resources in the manual testing process.

D. *Deployment*: After design, implementation is the stage in which the design is turned into a set of programmes and programme units. At this stage, software developers should select the most appropriate programming fashion for the application and, based on that, a programming language. The actual software is prepared for installation in the production environment during the deployment phase. The size of the installation package has a significant impact; larger the installation package, the longer it takes to instal in the production environment. This increases the storage and maintenance costs and necessitates a large amount of disc space for storage. Standard data compression techniques should be used to reduce the size of deployment or installation packages. Standalone Installation could also be avoided as they can consume a large amount of disk space. Modern software installation methods can also be used, such as centrally deploying a single copy of a programme and allowing users to access it only when necessary.

E. *Maintenance*: Software maintenance or evolution is the process of altering a system after it has been released in order to accommodate newer versions or additions. The maintenance stage of the software engineering process is the most expensive. The cost is proportional to the amount of energy wasted. In most cases, software quality deteriorates during the maintenance phase as bugs are fixed without understanding the scenarios completely. There are several circumstances in which interim solutions are applied, resulting in rework and the development of new software system problems. When new features need to be incorporated in an existing system, problems such as non-reproducible defects and resource leaks or needless resource demands on hardware, processor, memory, and so on arise; the quality of code is compromised. Maintenance is typically viewed as a lower-skilled operation than software development, and is often delegated to lower-level employees who may lack experience and knowledge of old and obsolete programming languages. As a result, it is the obligation of the top management to provide maintenance employees with tutorials and courses to help them get more familiar with old and new

International Journal of Scientific & Engineering Research, Volume 12, Issue 6, June-2021
ISSN 2229-5518

programming languages. This will speed up the maintenance procedure while also lowering costs and enhancing energy efficiency.

F. *Disposal*: Disposal is the ultimate phase of the software system. This phase begins when the software system's lifespan expires. The practise of reusing software code for future projects in order to save in-house software development costs is known as software recycling. Hardware recycling refers to the process of reusing and recycling equipment and resources before discarding them. To conserve natural resources, products that can be used frequently should be purchased. Efforts should be made to avoid disposal, which produces a significant amount of e-waste. If inefficient or old hardware is identified, it can be reused in other software systems with reduced resource needs, or it can be donated to universities with research interests or educational institutions for instructional reasons.

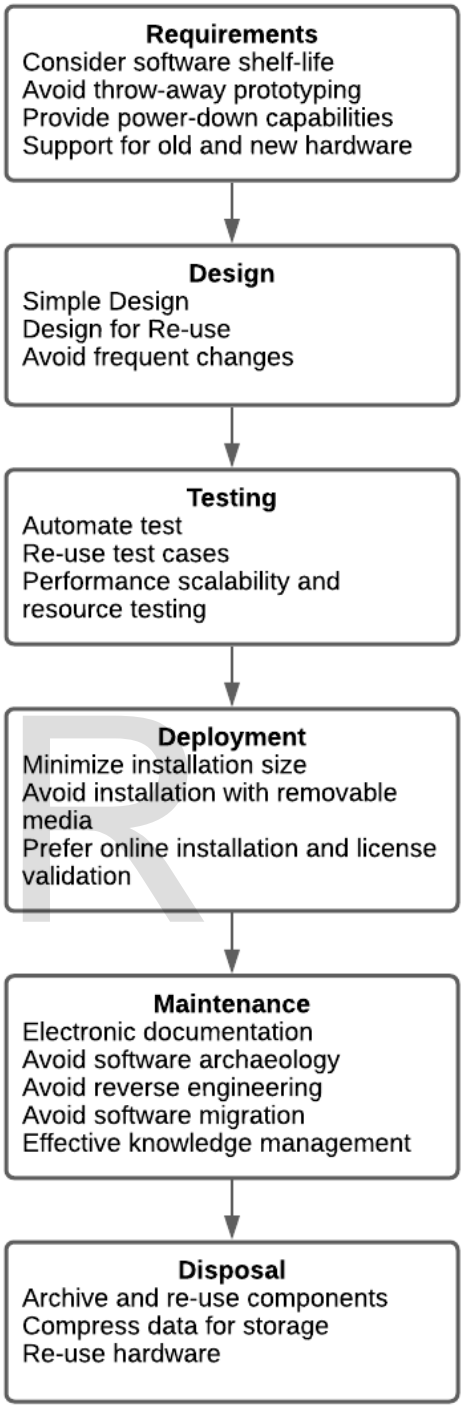

Figure 2: A Life Cycle Model for Software Sustainability

International Journal of Scientific & Engineering Research, Volume 12, Issue 6, June-2021
ISSN 2229-5518

IV. FURTHER GUIDELINES FOR DEVELOPERS

According to optimization recommendations, graphic design components account for 54% of the average website nowadays. Optimising graphics saves power as a result of lower network demand. Graphic design components, unlike photographs or charts, are images used to create borders, tabs, buttons, or a logo that provides an organisation or a corporation its identity. Graphic design elements are often smaller and have fewer colours. As a consequence, the number of colours in an image is reduced from RGB to a colour palette with a restricted number of colours, resulting in a significant reduction in file size (see Table 1). The photos in question do not need to be stored in the Portable Network Graphics (PNG) format, which enables indexed palettes, as they do in JPEG. Images are compressed more successfully in the PNG format than in the GIF format, resulting in reduced file sizes. It's also possible to remove words from a logo. The logo is then utilised as a backdrop graphic below the HTML text, with no content. Using an indexed palette for the design components, switching from JPEG to PNG is a viable technique to reduce transmission and save around 84 percent in both instances (with and without text). Furthermore, removing the text from PNG and JPEG files saves roughly 50-60% of the original file size. When converting from JPEG (with text) to PNG (without text), the overall savings is roughly 92%. As a result, as shown in Table 1, optimising graphical design components such as those mentioned above is advised in order to lower a website's network load and, as a result, its power consumption.

	File format	RGB living colour	J2 colour palette	7 colour pal
File size with text	JPEG 85%	8.152	n/a	n/a
	GIF	n/a	2,345	2,248
	PNG	9,036	1,320	1,290
File size without text	JPEG 85%	3,895	n/a	n/a
	GIF	n/a	1,528	1,431
	PNG	3,425	666	639

Table 1: File Sizes of different Graphical Design Elements at different optimization levels

Suggestions for Website Optimization Included in Development Tools: A possible technique to lessen the network load caused by a website is to optimise text-based source files made out of it. One option is to reduce the size of CSS files. CSSTidy (http://csstidy.sourceforge.net/) is an excellent tool for minimization.Extraneous characters (such as whitespaces and line breaks) are removed, as are values and attributes (such as default values), single properties (such as margin, border, and font) are shortened, and abbreviations are used. There are a host of other options for keeping CSS to a bare minimum.

V. IN-OFFICE BEST PRACTICES

Meeting rooms might be arranged so that natural light, rather than power-hungry lighting systems, are used for illumination. Natural ventilation or the use of energy-efficient fans can be used instead of air conditioners. Although this would not be practicable in locations with difficult climatic conditions, it would save not just energy but also the environment from dangerous components used in air conditioning.

Each of the SDLC phases may require travel, which should be avoided completely. For business meetings, air travel is a common means of transportation. Jet fuels are notoriously bad for the environment, thus preparation for other routes of transportation should be prioritised. Travel should be avoided by employing current communication by means such as the telephone, real-time content sharing software, video streaming, and email.

In software companies, storing and maintaining project artefacts, source, and product deliverables are commonplace. These are usually kept in huge data centres that require a specialised space with adequate cooling. The data centres not only consume a lot of energy, but they also need to be maintained on a regular basis. Peer-to-peer storage, often known as distributed storage, is a strategy to avoid using separate data centres. This will cut power usage, storage costs, and maintenance costs.

It is possible to avoid the use of paper by storing information in an electronic format. If authorisation is necessary, digital signatures can be employed. Any type of design document, architectural document, flow diagrams, and so on may be kept in electronic form or with the use of standardised tools that specialise in completing these activities. Scratch pads made of electronic material can be used instead of paper. In the process of information transfer or technological training, electronic books related to the product and technology must be preferred over physical books since they use far less paper.

VI. CONCLUSION

Small steps taken from our end can contribute vastly in building a sustainable environment. The software industry being a huge contributor towards carbon emissions must be handled with care. Application of all the methods mentioned could help bring the carbon footprint down and help build a sustainable green software industry. Reduction in direct and indirect carbon emission associated with the software industry. 1.4% of the total greenhouse gas emission comes from IT related industries. Tweaking the Software development life cycle with the suggestions provided could help reduce significant

International Journal of Scientific & Engineering Research, Volume 12, Issue 6, June-2021
ISSN 2229-5518

amounts of power,paper and time. Making use of HTML cache reduces electricity consumption by 8.6%.. Changing RGB living colours to colours-palette from 9036 to 1320 for the file format PNG. This could be reduced to 1290 with the use of 7 colour palette. A simple switch from JPEG to PNG can show promising results. Removal of text results could help electricity consumption by 50% to 60% approximately. Reduction of travel, reduced lighting in offices and minimal use of Air conditioning could reduce power consumption.

REFERENCES

[1] Dick, M., & Naumann, S. (2010). Enhancing Software Engineering Processes towards Sustainable Software Product Design. In EnviroInfo (pp. 706-715).

[2] Kern, Eva & Dick, Markus & Naumann, Stefan & Guldner, Achim & Johann, Timo. (2013). Green software and green software engineering - definitions, measurements, and quality aspects.

[3] F. Chen, J. Schneider, Y. Yang, J. Grundy and Q. He, "An energy consumption model and analysis tool for Cloud computing environments," 2012 First International Workshop on Green and Sustainable Software (GREENS), 2012, pp. 45-50, DOI: 10.1109/GREENS.2012.6224255.

[4] Zhong, Benjamin & Feng, Ming & Lung, Chung-Horng. (2010). A Green Computing Based Architecture Comparison and Analysis. 10.1109/GreenCom-CPSCom.2010.110.

[5] Wang, S., Chen, H., & Shi, W. (2011). SPAN: A software power analyzer for multicore computer systems. Sustainable Computing: Informatics and Systems, 1(1), 23–34. https://doi.org/10.1016/j.suscom.2010.10.002.

[6] S. S. Shenoy and R. Eeratta, "Green software development model: An approach towards sustainable software development," 2011 Annual IEEE India Conference, 2011, pp. 1-6, DOI: 10.1109/INDCON.2011.6139638.

[7] Capra, E., Formenti, G., Francalanci, C., & Gallazzi, S. (2010). The Impact of MIS Software on IT Energy Consumption. ECIS.

[8] Lago, P., & Jansen, T. (2010, December). Creating environmental awareness in service-oriented software engineering. In International conference on service-oriented computing (pp. 181-186). Springer, Berlin, Heidelberg.

[9] Koçak, S. A., Miranskyy, A., Alptekin, G. I., Bener, A. B., & Cialini, E. (2013, February). The impact of improving software functionality on environmental sustainability. In First International Conference on Information and Communication Technologies for Sustainability (ICT4S) (pp. 95-100)

[10] Agarwal, S. (2014). Impact of green computing in IT industry to make eco friendly environment. Journal of Global Research in Computer Science, 5(4), 05-10.

[11] Soomro, T. R., & Sarwar, M. (2012). Green computing: From current to future trends. World Academy of Science, Engineering and Technology, 63, 538-541.

[12] Harmon, R. R., & Auseklis, N. (2009, August). Sustainable IT services: Assessing the impact of green computing practices. In PICMET'09-2009 Portland International Conference on Management of Engineering & Technology (pp. 1707-1717). IEEE.

[13] Saha, B. (2018). Green computing: Current research trends. International Journal of Computer Sciences and Engineering, 6(3), 467-469.

[14] P. Cooper, T. Crick, T. Tryfonas and G. Oikonomou, "Whole-Life Environmental Impacts of ICT Use," 2015 IEEE Globecom Workshops (GC Wkshps), San Diego, CA, 2015.

[15] Malmodin, J., Bergmark, P., & Lundén, D. (2013). The future carbon footprint of the ICT and E&M sectors. on Information and Communication Technologies, 12.

[16] Kern E, et al, Impacts of software and its engineering on the carbon footprint of ICT, Environ Impact Assess Rev (2014), http://dx.doi.org/10.1016/j.eiar.2014.07.003.