N° d'ordre NNT : 2017LYSE1256

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

École Doctorale ED 512
Informatique et Mathématiques de Lyon (InfoMaths)

Spécialité de doctorat : Informatique

Soutenue publiquement le 23/11/2017, par :
Hayri ACAR

Software development methodology in a Green IT environment

Devant le jury composé de :

Oussalah Mourad Chabane, Professeur des Universités, Université de Nantes	Rapporteur
Pierson Jean-Marc, Professeur des Universités, Université Paul Sabatier Toulouse 3	Rapporteur
Bellatreche Ladjel, Professeur des Universités, ENSMA Poitiers	Examinateur
VARGAS-SOLAR Genoveva, Chargé de Recherche CNRS, LIG Grenoble	Examinatrice
Ghodous Parisa, Professeur des Universités, Université Lyon 1	Directrice de thèse
Gelas Jean-Patrick, Maître de Conférence, ENS Lyon	Co-directeur de thèse
Isiklar Alptekin Gülfem, Associate Professor, Galatasaray University	Co-directrice de thèse
Lefèvre Laurent, Chargé de Recherche INRIA, ENS Lyon	Invité

UNIVERSITE CLAUDE BERNARD - LYON 1

Président de l'Université	**M. le Professeur Frédéric FLEURY**
Président du Conseil Académique	M. le Professeur Hamda BEN HADID
Vice-président du Conseil d'Administration	M. le Professeur Didier REVEL
Vice-président du Conseil Formation et Vie Universitaire	M. le Professeur Philippe CHEVALIER
Vice-président de la Commission Recherche	M. Fabrice VALLÉE
Directrice Générale des Services	Mme Dominique MARCHAND

COMPOSANTES SANTE

Faculté de Médecine Lyon Est – Claude Bernard	Directeur : M. le Professeur G.RODE
Faculté de Médecine et de Maïeutique Lyon Sud – Charles Mérieux	Directeur : Mme la Professeure C. BURILLON
Faculté d'Odontologie	Directeur : M. le Professeur D. BOURGEOIS
Institut des Sciences Pharmaceutiques et Biologiques	Directeur : Mme la Professeure C. VINCIGUERRA
Institut des Sciences et Techniques de la Réadaptation	Directeur : M. X. PERROT
Département de formation et Centre de Recherche en Biologie Humaine	Directeur : Mme la Professeure A-M. SCHOTT

COMPOSANTES ET DEPARTEMENTS DE SCIENCES ET TECHNOLOGIE

Faculté des Sciences et Technologies	Directeur : M. F. DE MARCHI
Département Biologie	Directeur : M. le Professeur F. THEVENARD
Département Chimie Biochimie	Directeur : Mme C. FELIX
Département GEP	Directeur : M. Hassan HAMMOURI
Département Informatique	Directeur : M. le Professeur S. AKKOUCHE
Département Mathématiques	Directeur : M. le Professeur G. TOMANOV
Département Mécanique	Directeur : M. le Professeur H. BEN HADID
Département Physique	Directeur : M. le Professeur J-C PLENET
UFR Sciences et Techniques des Activités Physiques et Sportives	Directeur : M. Y.VANPOULLE
Observatoire des Sciences de l'Univers de Lyon	Directeur : M. B. GUIDERDONI
Polytech Lyon	Directeur : M. le Professeur E.PERRIN
Ecole Supérieure de Chimie Physique Electronique	Directeur : M. G. PIGNAULT
Institut Universitaire de Technologie de Lyon 1	Directeur : M. le Professeur C. VITON
Ecole Supérieure du Professorat et de l'Education	Directeur : M. le Professeur A. MOUGNIOTTE
Institut de Science Financière et d'Assurances	Directeur : M. N. LEBOISNE

Acknowledgements

This thesis would not have been possible without the assistance, guidance and encouragements of many individuals.

First of all, I would like to thank my supervisors, Associate Professor Jean-Patrick Gelas, Assistant Professor Gülfem Isiklar Alptekin and Professor Parisa Ghodous for their supports, advices and discussions.

I also would like to thank all the members of the SOC team, and my colleagues at University of Lyon1, for their talks and discussions.

I would like to thank my parents and my brothers and all my wonderful friends for their love and support.

Finally, my sincere thanks to my wife Fatma and two children Ibrahim and Muhammed for their limitless patience and support throughout my thesis years.

Contents

1 Introduction **15**
- 1.1 Motivations . . . 15
- 1.2 Research questions . . . 17
- 1.3 Contributions . . . 18
- 1.4 Outline . . . 18

2 Definitions of the Terms: Sustainable and Green **21**
- 2.1 Sustainable . . . 21
  - 2.1.1 ICT sustainable . . . 23
  - 2.1.2 Sustainable Software . . . 25
- 2.2 Green . . . 26
  - 2.2.1 Green Software . . . 28
  - 2.2.2 Green with/within Software . . . 29
- 2.3 Conclusion . . . 30

3 Related Works **33**
- 3.1 Hardware methodologies . . . 34
- 3.2 Software methodologies . . . 35
- 3.3 Hybrid methodologies . . . 50
- 3.4 Conclusion . . . 51

4 Sustainable and Green Software Engineering Process **53**
- 4.1 BUA Methodology . . . 53
- 4.2 Requirements . . . 55
- 4.3 Design and Implementation . . . 57
- 4.4 Tests . . . 59
- 4.5 Usage . . . 61
- 4.6 Maintenance . . . 63
- 4.7 Disposal . . . 64
- 4.8 Green analysis . . . 66
- 4.9 Conclusion . . . 66

5 A generic power consumption methodology: GMTEEC **67**
- 5.1 Business Layer . . . 67
  - 5.1.1 Hotpoint . . . 67
  - 5.1.2 Improvement . . . 68
- 5.2 Application Layer . . . 69
  - 5.2.1 Language . . . 69
  - 5.2.2 Library . . . 71
- 5.3 Interface Layer . . . 73
  - 5.3.1 Platform . . . 73
  - 5.3.2 Operating System . . . 75
- 5.4 Hardware Layer . . . 76

5.4.1 Component . . . 76
5.4.2 Simulation . . . 76
5.5 Conclusion . . . 76

6 GMTEEC methodology applied: TEEC **79**
6.1 CPU . . . 80
6.2 Memory . . . 82
6.2.1 Activate power . . . 84
6.2.2 Precharge power . . . 85
6.2.3 Read power . . . 85
6.2.4 Write power . . . 85
6.2.5 Total power . . . 85
6.3 Hard Disk . . . 86
6.3.1 Hard disk structure . . . 86
6.3.2 Power modeling . . . 87
6.4 Network . . . 89
6.5 Total power consumption of all components . . . 89
6.6 Conclusion . . . 89

7 Experiments and Validation **91**
7.1 Fibonacci sequence . . . 91
7.2 Source code adjustment . . . 92
7.2.1 Strength reduction . . . 92
7.2.2 Eliminate common subexpressions . . . 93
7.2.3 Code motion . . . 94
7.2.4 Unrolling loops . . . 96
7.3 Several function optimizations . . . 96
7.3.1 Tests Description . . . 97
7.3.2 Results . . . 99
7.3.3 Validation . . . 100
7.4 TEEC compared with three other tools . . . 101
7.4.1 Search an Integer . . . 102
7.4.2 Students mini-project . . . 104
7.5 Conclusion . . . 107

8 Conclusion **109**
8.1 Lessons learned . . . 109
8.2 Future directions . . . 110
8.3 Publications . . . 111

List of Figures

1.1 Electricity consumption (TWh) [1] 16

2.1 Software Sustainability Heart 23
2.2 Our Sustainable Software definition 26
2.3 Sustainable and Green Software 27
2.4 Our Green Software definition 29
2.5 Green with Software 31
2.6 Green within Software 31

3.1 Software power consumption methodologies classification 33
3.2 Systematic map in a bubble plot of research type and nature facets 36

4.1 BUA (BeforeUsageAfter) methodology 55
4.2 Requirements stage with criteria for Green Analysis 57
4.3 Design and implementation stage with criteria for Green Analysis 59
4.4 Tests stage with criteria for Green Analysis 61
4.5 Usage stage with criteria for Green Analysis 63
4.6 Maintenance stage with criteria for Green Analysis 64
4.7 Disposal stage with criteria for Green Analysis 66

5.1 GMTEEC 68
5.2 Global desktop operating system market share in May 2017 [2] 75
5.3 Global mobile and tablet operating system market share in May 2017 [3] 75
5.4 Distribution of peak power usage by hardware subsystem in one of Google's data centers [4] 76
5.5 Average power use per server distributed by hardware subsystem [5] 77
5.6 Computer power distribution by components [6] 77

6.1 TEEC 79
6.2 Code manipulations 80
6.3 One common gate in CPU 81
6.4 The functional bloc diagram of a DDR3 SDRAM 83
6.5 Hard disk structure. 87
6.6 LBAs. 88

7.1 Power consumption of Fibonacci sequence with TEEC. 92
7.2 Power consumption of Fibonacci sequence with Joulemeter. 92
7.3 Strength reduction unoptimized. 93
7.4 Strength reduction optimized. 93
7.5 Subexpression unoptimized. 94
7.6 Subexpression optimized. 94
7.7 Code motion unoptimized. 95
7.8 Code motion optimized. 95
7.9 Unrolling loops. 96

7.10 Unoptimized functions power consumption. 99
7.11 Unoptimized functions energy consumption. 99
7.12 Optimized functions power consumption. 100
7.13 Unoptimized functions energy consumption. 100
7.14 wattsup?PRO. 101
7.15 Unoptimized and optimized functions power consumption obtained with wattsup?PRO.101
7.16 WattsUp-Pro power consumption measured and used as reference. 102
7.17 TEEC power consumption estimation. 102
7.18 IPG power consumption estimation. 103
7.19 Joulemeter power consumption estimation. 103
7.20 WattsUp-Pro power consumption measured and used as reference. 105
7.21 TEEC power consumption estimation. 105
7.22 IPG power consumption estimation. 105
7.23 Joulemeter power consumption estimation. 106
7.24 Performance and energy results of each project. 106

List of Tables

2.1 Sustainable Software Definitions 26
2.2 Green Software Definitions 29

3.1 Comparison of Powermeters 34
3.2 Studies using hardware methodologies to measure the energy consumption of software 35
3.3 Research Type Facet 36
3.4 Tools using CPU component to estimate the software energy consumption 41
3.5 Tools using Memory component to estimate the software energy consumption . . 45
3.6 Tools using Hard Disk component to estimate the software energy consumption . 47
3.7 Tools using Network component to estimate the software energy consumption . . 50

5.1 Languages Types 71
5.2 Libraries 72
5.3 Virtual and Physical machines advantages and disadvantages 75

6.1 Possibilities to reduce power consumption of the CPU. 82
6.2 Data sheet specifications 84

7.1 Functions 99
7.2 Functions time execution. 100
7.3 Energy consumption comparison 103

Summary

The number of mobile devices (smartphone, tablet, laptop, etc.) and Internet users are continually increasing. Due to the accessibility provided by cloud computing, Internet and Internet of Things (IoT), users use more and more software applications which cause an increasing effect on gas emission. Thus, ICT (Information and Communication Technologies) is responsible of around 2% worldwide greenhouse gas emissions which is equivalent of that emitted by the airline industry. According to recent reports, the Intergovernmental Panel on Climate Change (IPCC), CO2 emissions due to ICT are increasing widely. Nevertheless, ICT, in allowing to solve complex problems in other sectors, can greatly and easily participate to reduce significant portion of the remaining 98% of global CO2 emissions.
The use of software implies hardware operations which are physically responsible of energy consumption. Consequently, software is indirectly involved in the energy consumption. Thus, we need to reduce software energy consumption while maintaining the same functionalities for the software in order to build sustainable and green software.
Firstly, in this thesis work, we define the terms sustainable and green in the area of software development. To build a software product, we need to follow a software engineering process. Hence, we define and describe sustainable and green criteria to be respected after each step of this process in order to establish a sustainable and green software engineering process.
Then, we focus on the software energy consumption estimation. Many research works tried to propose various tools to estimate the energy consumption due to software in order to reduce carbon footprint. Unfortunately, these studies, in the majority of cases, consider only the CPU and neglects all others components. Existing power consumption methodologies need to be improved by taking into account more components susceptible to consume energy during runtime of an application. Writing sustainable, power efficient and green software necessitates to understand the power consumption behavior of a computer program. One of the benefits is the fact that developers, by improving their source code implementations, will optimize software power consumption. Moreover, there is a lack of analyzing tool to dynamically monitor source code energy consumption of several components. Thus, we propose GMTEEC (Generic Methodology of a Tool to Estimate Energy Consumption) which is composed of four layers assisting developers to build a tool estimating the software power consumption. Hence, in our work, respecting the layers of GMTEEC, we develop TEEC (Tool to Estimate Energy Consumption) which is based on mathematical formula established for each component (CPU, memory, hard disk, network) in order to estimate the total software energy consumption. Moreover, we add in TEEC the capacity to locate dynamically the hotpoints which are the parts of source code consuming the greater amount of energy in order to help and guide developers to optimize their source code and build efficient, sustainable and green software.
We performed a variety of experiments to validate the accuracy and quality of the sustainable and green software engineering process and TEEC. The results demonstrate the possibility to save significant quantity of energy and time at limited costs with an important positive impact on environment.

Résumé

Le nombre de périphériques mobiles (smartphone, tablette, ordinateur portable, etc.) et les internautes augmentent continuellement. En raison de l'accessibilité du cloud computing, de l'Internet et de l'Internet des Objets (IdO), les utilisateurs utilisent de plus en plus d'applications logicielles qui provoquent un effet croissant sur les émissions de gaz à effet de serre. Ainsi, les TIC (Technologies de l'Information et de la Communication) sont responsables d'environ 2% des émissions mondiales de gaz à effet de serre qui sont équivalentes à celles émises par l'industrie aérienne. Selon des rapports récents, le Groupe d'experts Intergouvernemental sur l'Evolution du Climat (GIEC), les émissions de CO2 dus aux TIC augmentent rapidement. Néanmoins, les TIC, en permettant de résoudre des problèmes complexes dans d'autres secteurs, peuvent grandement et facilement participer pour réduire une partie importante des 98% restants des émissions mondiales de CO2.
L'utilisation du logiciel implique des opérations matérielles qui sont physiquement responsables de la consommation d'énergie. Par conséquent, le logiciel est indirectement impliqué dans la consommation d'énergie. Ainsi, nous devons réduire la consommation d'énergie du logiciel tout en conservant les mêmes fonctionnalités pour le logiciel afin de créer des logiciels durables et verts.
Premièrement, dans ce travail de thèse, nous définissons les termes «durable et vert» dans le domaine du logiciel afin de créer des logiciels respectant les critères de ces termes. Pour créer un produit logiciel, nous devons suivre un processus d'ingénierie logicielle. Par conséquent, nous décrivons des critères durables et verts à respecter après chaque étape de ce processus afin d'établir un processus d'ingénierie logicielle durable et écologique.
En particulier, nous nous concentrons sur l'estimation de la consommation d'énergie du logiciel. De nombreux travaux ont essayé de proposer divers outils pour estimer la consommation d'énergie due aux logiciels afin de réduire l'empreinte carbone. Pendant longtemps, les solutions proposées se sont concentrées uniquement sur la conception du matériel, mais ces dernières années, les aspects logiciels sont également devenus importants. Malheureusement, ces études, dans la plupart des cas, ne considèrent que le CPU et négligent tous les autres composants. Les modèles de consommation d'énergie existants doivent être améliorés en tenant compte de plus de composants susceptibles de consommer de l'énergie pendant l'exécution d'une application. L'écriture d'un logiciel durable, performant et vert nécessite de comprendre le comportement de consommation d'énergie d'un programme informatique. L'un des avantages est que les développeurs, en améliorant leurs implémentations du code source, optimiseront la consommation d'énergie du logiciel. De plus, il existe un manque d'outil d'analyse pour surveiller dynamiquement la consommation d'énergie du code source de plusieurs composants. Ainsi, nous proposons GMTEEC (Méthodologie Générique d'Outil pour Estimer la Consommation Energétique) qui se compose de quatre couches aidant et guidant la construction d'un outil permettant d'estimer la consommation énergétique d'un logiciel. Ainsi, dans notre travail, en respectant les couches de GMTEEC, nous créons TEEC (Outil pour Estimer la Consommation Energétique) qui repose sur une formule mathématique établie pour chaque composant (CPU, mémoire, disque dur, réseau) susceptible de consommer de l'énergie afin d'estimer la consommation totale d'énergie du logiciel composée de la somme de chaque consommation d'énergie par composant. De plus, nous ajoutons à TEEC la capacité de localiser dynamiquement les points chauds qui sont les parties du code source consommant la plus grande quantité d'énergie afin d'aider et guider les développeurs à optimiser

leur code source et à créer des logiciels efficaces, durables et verts.
Nous avons réalisé une variété d'expériences pour valider la précision et la qualité du processus d'ingénierie logicielle durable et verte et la précision et la qualité de TEEC. Les résultats démontrent la possibilité de réduire la consommation d'énergie et d'améliorer les performances à des coûts limités avec un important impact positif sur l'environnement.

Chapter 1

Introduction

"People with passion can change the world for the better."

Steve Jobs

1.1 Motivations

Experts have commonly recognized that human activity is responsible for the increase of the earth atmospheric temperature mainly due to the greenhouse gases generated by energy production. Consequently, the United Nations Climate Change Conference, COP 21 (Conference of the Parties), has reasserted the aim of keeping the increase in temperature below 2 ° C by the end of the century [7].

In the last years Information and Communication Technologies (ICT) is increasing very fast with the goal to propose new solutions and facilitates people's life. In fact, the number of Internet users and websites, sent email, Google searches, written blog posts, sent tweets, viewed Youtube videos, photos uploaded on Instagram, Facebook, Google+, Twitter, Linkedin and Pinterest active users, Skype calls, websites hacked, Computers, Smartphones and Tablets sold, Internet traffic are growing each day in the world. Consequently, energy consumption, electricity used and greenhouse gas emissions (GHGE) due to ICT are also increasing dramatically. Hence, ICT is responsible about 2% of global CO2 emissions [8] which is the equivalent emitted by the airline industry. Moreover, with the considerable increase of mobile devices and the expansion of network, as it is represented in Figure 1.1, ICT sector electricity consumption will increase about 60% between 2007 and 2020 [1].

Nevertheless, ICT is contributing efficiently to reducing the global carbon footprint and to solving complex problems in several sectors of activity. Thus, ICT can greatly and easily participate to reduce significant portion of the remaining 98% of global CO2 emissions from other sectors. GeSI's SMARTer2020 [9] report shows that the increased use of ICT, such as cloud computing, video conferencing, etc. could cut the projected 2020 global greenhouse gas emissions by 16,5% [9], amounting to $1.9 trillion in gross energy and fuel savings and a reduction of 9.1 Gigatonnes carbon dioxide equivalent (GtCO2e) of greenhouse gases. This is equivalent to more than seven times the ICT sector's emissions in the same period. Hence, it is necessary to continue these improvements in order to keep the same functionalities of our ICT products and at the same time reduce the energy consumption due to these products to obtain sustainable and green products which will be eco-friendly with the environment. A lot of improvements have been made to save energy in several sectors.

Particularly, within ICT hardware, improvements made by manufacturers are considerable and limits of improvement are nearly reached. In fact, physical components in a device are consuming

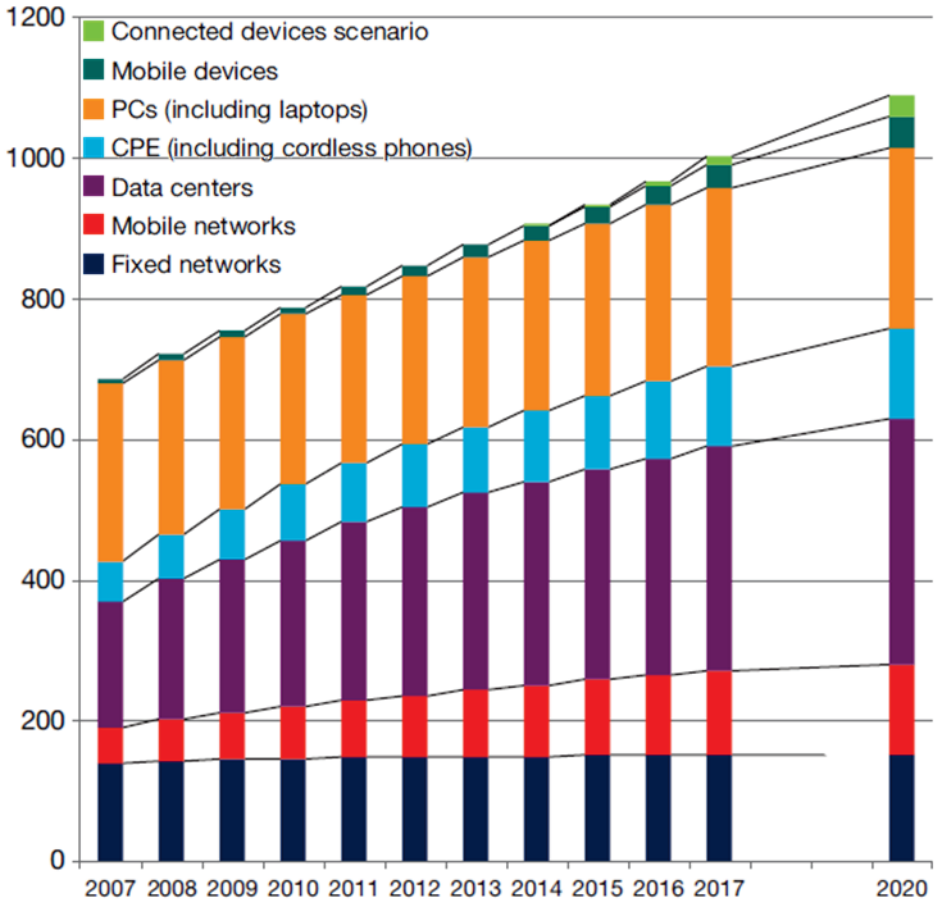

Figure 1.1: Electricity consumption (TWh) [1]

energy. However, software use involved hardware operations which imply energy consumption. Hence, software is also indirectly source of energy consumption. Thus, more and more researches are investigating to save energy concerning software use. Moreover, software is more complicated to develop, manage and sell and contributes to create more economic activity than hardware.

Several definitions of the terms sustainable and green have been proposed in order to act for the environment. Hence, according to the sector, the meanings of those terms are not similar and a meaning in a sector can not be applied in another sector to create a product linked at this sector and respecting the meanings of those terms. In ICT sector also there are a lot of definitions proposed concerning sustainable and green terms. However, an accurate approach is needed to define these terms in our research area concerning software products in order to create sustainable and green software.

Companies whose main activity is to develop and sell software use different development lifecycle to create software product. Firstly, the direct way is to begin the implementation of software without to carrying out any description of specification functional and techniques and also any design of development. With this approach, companies want to earn money quickly and the goal is to offer to the client a software product operating. By this way, it is too complicated to perform improvement of the software to have a new version. Each time, it is like a new software is developed. Moreover, it will be very difficult to realize the maintenance of this software product especially when the developer performing is different from the developer who developed the software. Another way is to follow a software engineering process described by the company in order to have a better organization in the tasks to create the software product. In this case, specifications are defined between the client and the company. Then, implementation and tests

are performed. Next, the software product is deployed for the use of the client. Maintenance is realized to solve the defects described by the client. During these steps different team could work separately to check the quality of the software product. Even if there is a better organization and management in this case, there is a lack of concern for sustainable and green approach. In fact, the company creates the software product to earn money and does not take into account the green aspects of the software product to reduce the green house gas emissions keeping the same functionalities. Hence, it is important to measure and/or estimate the energy consumption due to software in order to know if it is possible to limit negative impacts.

Several tools using different methodologies have been presented in order to reduce the energy consumption due to software. Firstly, studies were about hardware methodologies which consist to measure directly the energy consumption of components (CPU, memory, hard disk, network, etc.) with powermeters or printed circuits connected at the component. By this way, it is possible to obtain accurate results about the energy consumption of components. However, the main limits concerning this methodology are the fact that it is impossible to connect a powermeter on a Virtual Machine (VM) and measure the energy consumption of a particular process. Moreover, this approach causes additional energy consumption due to the use of hardware device to measure the software energy consumption and depending on the situations it could be expensive to buy a powermeter or build a special printed circuit. Then, researches have been oriented on the hybrid methodologies which consist to use a hardware device such as a powermeter to perform the energy measurement and a software tool to get automatically and manage the information obtained with the hardware device. Even if the management of energy consumption is easier with this approach, we observe the same limits of those hardware methodologies. Hence, new recent software methodologies are more and more studying to simplify and facilitate the software energy measurement. In this case, the limits observed in the both previous methodologies are solved because an estimation of software energy consumption is proposed. In fact, mathematical formula are established to estimate the energy consumption of main components such as CPU, memory, disk, network, etc. taking into account the features of each component. Then, the accuracy of this approach is checked with hardware devices such as powermeters. Hence, we choose to study, describe and develop this software methodology in order to reduce the energy consumption of software while keeping the same functionalities of software. Writing power efficient software requires understanding the software application power consumption behavior. Thus, developer could develop efficient, sustainable and green software.

Otherwise, several tools have been implemented to estimate the energy consumption of software. However, in the majority of cases, the CPU was considered the most energy consumer component and the other components have been neglected. In the other cases researches limited their study only at their specific domain of research and proposed tools which take into account only one particular component. Hence, it is necessary to have a tool able to estimate the energy consumption of several components (CPU, memory, disk, network, etc.) in the same time during the runtime of software. If we remain at this step, developers could know the energy consumption of their software. They did not know if their source code is efficient or not and even if they try to improve their source code they will lose enormous time to identify the parts of code to optimize. Consequently, it is important to help and guide developers to locate dynamically the hotpoints of their source code in order to improve their code to obtain efficient software.

1.2 Research questions

In this thesis, we will focus on the following research questions:

- What are the meanings of the terms sustainable and green related to software area?

- How is it possible to establishes sustainable and green software engineering process?

- How to build a tool allowing to estimate software energy consumption?

- How to help and guide developers to optimize their source code in order to build efficient, sustainable and green software?

1.3 Contributions

The main goal of our thesis is to provide a methodology allowing to create sustainable and green software product in order to reduce the greenhouse gas emissions. Hence, the contributions are summarized as follows:

- **Definition of the terms sustainable and green** for a software product. After studying, the proposed definitions of those terms in different sectors, we will take a look at the ICT sector and more particularly at the software product level. Understanding the meanings of those terms will allow to create efficient product limiting negative impacts on the environment.

- **Sustainable and green software engineering process** respecting the meanings of the terms defined. BUA (Before Usage After) methodology is proposed in order to describe all the steps to respect for the creation of a software product. The contribution of this methodology is to add a green analysis step after each classical step of a software engineering process. This green analysis stage consist to check if the sustainable and green criteria established for each stage are validated. When a step validates all the criteria, it is possible to pass to the next step. If not, a back at a previous step is performed.

- **GMTEEC** (Generic Methodology of a Tool to Estimate Energy Consumption) is proposed in order to develop a tool allowing to estimate software energy consumption. In fact, many tools have been presented and in each case, authors take into account their own way to create an energy estimating tool. With, GMTEEC, we propose an approach to respect layers to have a generic methodology which adapt at any situation.

- **TEEC** (Tool to Estimate Energy Consumption) is an application of GMTEEC. With this tool, it is possible to estimate the energy consumption of several components like CPU, memory, hard disk and network in order to obtain the global software energy consumption. Moreover, TEEC allow to locate parts of source code which are the most energy consumer. Hence, TEEC helps and guides developers to optimize their source code in order to obtain efficient, sustainable and green software.

1.4 Outline

The remaining of this thesis manuscript is organized as follows.

Chapter 2 proposes definitions of the terms sustainable and green. A generic meaning of those terms is given. Then, a study examining the definitions of those terms in ICT sector is carried out in order to better understand their meaning. Thus, a definition of sustainable and green software, green with software and green within software is proposed in order to take into account

during software engineering process.

Chapter 3 presents a summary of the previous work in this area of research. Hardware and Hybrid methodologies allowing to measure the software energy consumption are described and several tools using these methodologies are examined. Then, their limits are presented. A systematic review is performed concerning tools proposed in software methodologies. The lacks are detected in order to propose solutions.

Chapter 4 describes a sustainable and green software engineering process. Based on the definition established on Chapter 2, BUA (Before Usage After) methodology is proposed to follow well defined steps to create a sustainable and green software product. For this, in the software engineering process, after each step a green analysis step is added in order to check if sustainable and green criteria are respected.

Chapter 5 describes a generic power consumption methodology: GMTEEC (Generic Methodology of a Tool to Estimate Energy Consumption). According to their research area, researchers proposed tools allowing to estimate the software energy consumption. However, there is not a generic methodology describing the layers to develop a tool to estimate software energy consumption. Hence, GMTEEC helps and guides to create a tool in all situations.

Chapter 6 presents the tool we developed called TEEC (Tool to Estimate Energy Consumption). All the layers of the GMTEEC have been studied in order to create TEEC allowing to estimate the energy consumption of a variety of components like CPU, memory, hard disk and network. Hence, total software energy consumption is calculated. In addition, TEEC helps and guides developers to find and optimize the hotpoints in the source code.

Chapter 7 evaluates and validates our contributions. Several experiments are performed to validate the accuracy of the proposed tool TEEC and the effectiveness of the sustainable and green software engineering process.

Chapter 8 summarizes the thesis work and contributions presented. We propose also future research directions.

Chapter 2

Definitions of the Terms: Sustainable and Green

> "The right relation... is that the land should suffice for the maintenance of the inhabitants, and that there should be as many inhabitants as the land can maintain."
>
> Jean-Jacques Rousseau

Understanding the meanings of the terms will permit us to build products in respect to these criteria and conditions. Thus, we will present the definitions of terms "Sustainable" and "Green" used in other studies, and then we will introduce our own definitions. Our approach to define the terms is inspired by the following paper [10] where sustainability is separated in three levels which are Business Process Sustainability, Services Sustainability and Information Systems Sustainability. Then, in this last level, there are sub-levels which are ICT Sustainability including IT Sustainability containing Hardware and Software Sustainability. Next, in each level, Sustainability is described in detail and comparison is proposed between the different studies. Even if our approach seems similar, there are several differences because we do not consider Sustainability composed by three levels, but it is the heart of three dimensions which are Economic, Social and Environment. Moreover, we take into account only ICT and Software sub-levels in order to be implied only on our area of research. The same approach for the terms linked to Green is followed in order to be focused mainly on the software area. Moreover, we propose for each terms our own definitions.

2.1 Sustainable

Independently of any context, we will examine the general definition of the term "Sustainable" in order to better understand it meanings and the effects of its application on concrete projects. Sustainable is a largely used word. We list the following definitions taken from different sources:

- The Cambridge dictionary [11]: "causing little or no damage to the environment and therefore able to continue for a long time".

- Another definition proposed by Oxford dictionaries [12] is the fact of "conserving an ecological balance by avoiding depletion of natural resources".

- In [10], sustainable is “the capacity of something to last for a long time“.

- According to [13], “Sustainability can be also discussed with reference to a concrete system such as an ecological system, a human network, or a specific software system. Here, global sustainability implies the capacity for endurance given the functioning of all these systems in concert.“.

- Brundtland Commission: “sustainable development is the way of development that meets the needs of the present without compromising the ability of future generations to meet their own needs“ [14].

- In [15], five dimensions of sustainability are taken into account:

  - Individual sustainability: refers to the maintenance of the private good of individual human capital.

  - Social sustainability: meaning maintaining social capital and preserving the societal communities in their solidarity.

  - Economic sustainability: aims at maintaining assets which do not only include capital but also added value. This requires to define income as the amount one can consume during a period and still be as well off at the end of the period, as it devolves on consuming added value, rather than capital.

  - Environmental sustainability: seeks to improve human welfare by protecting natural resources.

  - Technical sustainability: the central objective of long-time usage of systems and their adequate evolution with changing surrounding conditions and respective requirements.

However, it is possible to include human and technical sustainability on the others in order to have only three main dimensions, as cited on the UN, which are environmental protection, economic growth and social equality. Respecting the three dimensions of this report, we can define the sustainability of each one as:

- Environmental protection: in sustainable environment restoration is faster than destruction.

- Economic growth: economy respects ecological criteria.

- Social equality: sustainable society consists of justice and/or decreasing poverty.

Sustainability has been defined several times, but the most frequently cited definition is from Our Common Future [14] which contains two main concepts. First idea is the “needs“ and the second concept is the “limitations“.
In the literature, sustainability is applied in several sectors. However, in our work, we focus on the ICT sustainability and more particularly on the software sustainability that is illustrated in Figure 2.1.

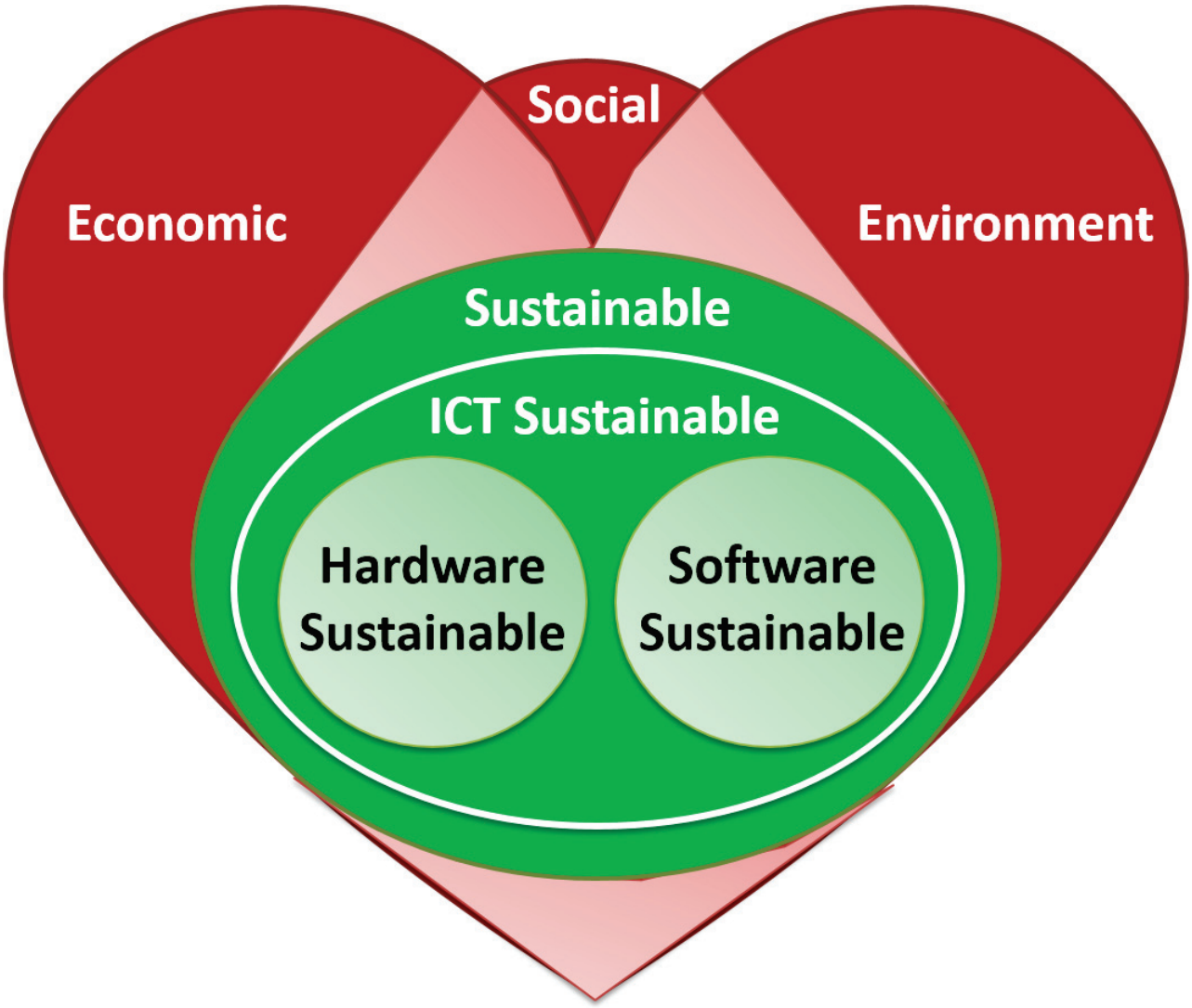

Figure 2.1: Software Sustainability Heart

2.1.1 ICT sustainable

ICT Sustainability becomes more and more important in research area with the creation of the conference ICT4S (ICT for Sustainability) [16] since 2013. This conference is composed of researchers with government and industry representatives, including also decision-makers. Researchers focusing on ICT effects on sustainability and developers of sustainable ICT systems. In the appendix of the proceedings [17], recommendations taken into account under the title "How to Improve the Contribution of ICT to Sustainability" are: "The transformational power of ICT can be used to make our patterns of production and consumption more sustainable. However, the history of technology has shown that increased energy efficiency does not automatically contribute to sustainable development. Only with targeted efforts on the part of politics, industry and consumers it will be possible to unleash the true potential of ICT to create a more sustainable society".

Sustainable ICT implies the respect of several criteria which can enumerated like following:

- **Increase the recycling:** reuse as much as possible existing resources. Each part of a device such as metals, plastic, glass, minerals should be recycled in order to be reused in another component.

- **Reduce the energy consumption:** ICT equipment needs to be manufactured with the minimum energy. Moreover, these equipments, during their usage, have to consume the minimum energy. Hence, it is possible to keep the sustainable aspects.

- **Reduce the waste:** During all the steps of its lifecycle (manufacture, transportation, storage, buy sell, usage, maintenance, disposal), an ICT equipment has to reduce the waste of all type.

- **Reduce the materials:** Use more and more small pieces of ICT equipment which are manufactured using less energy implying also less waste. In addition, usually a taller laptop is not more efficient than a smaller one.

ICT for sustainability can be grouped into:

- **Sustainability with ICT:** build, develop and extend products with ICT equipments which are respecting sustainability.

- **Sustainability within ICT:** Reduce the energy consumption, waste and materials during the whole life-cycle of an ICT product.

The European Commission is supporting research and development projects whose main goal is, through the use of ICT, to improve water management, energy efficiency and adaptation to climate change.

In 2009, it is adopted the Recommendation entitled "mobilising ICTs to facilitate the transition to an energy-efficient, low-carbon economy" in order to provide the policy framework, in this area, for its activities and to unlock, in several ways, the ICT sector energy efficiency potential.

The ICT for Energy Efficiency Forum has been set up in the goal that ICT industry develop a framework to set itself energy efficiency targets and measure its energy and environmental performance. This Forum has been encouraged to take into account also buildings and transport sectors where using ICT is possible to save energy significantly.

Software plays a major role, both as part of the problem and as part of the solution as noted in [18].

According to [19], ICT can develop solutions that offer benefits both internally and across the enterprise.

As defined in the SIGGreen Statement [20], the Information Systems discipline can have a central role in creating an ecologically sustainable society because of the fields five decades of experience in designing, building, deploying, evaluating, managing, and studying information systems to resolve complex problems.

The Ericsson Energy and Carbon report [21] describes how the increase use of ICT could help cut greenhouse gas emissions by more than 15%.

2.1.2 Sustainable Software

Sustainable software should be used in several areas such as data centers, Web services, software systems, software products, etc. Several studies are in process, here we are interested particularly at the software products.

Several researches show the importance of sustainable software products in order to solve complex problems and contribution of saving energy.

Sustainability needs to be considerate as an important condition to respect during a software development even if it not the habits for the developers. For this, the authors in [15] define the following issue: now, little guidance is presented for the systems under development on how the sustainability can be improved with the contribution of software engineering. To solve this problem, authors, for sustainability, describe a reference model where sustainability is composed of five dimensions which are: individual, environmental, economic, technical and social sustainability.

As noted in [22], improving power consumption allows to obtain sustainable software. Until now, hardware improvements have given interesting results concerning energy efficiency of components thanks to the great interest of industry about this aspect. However, there are several improvements to make in software side in order to build sustainable software.

In [23], sustainable software is defined as a software, whose indirect and direct negative impacts on society, economy, environment and human beings that result from the software development, deployment, and usage which have a positive effect on sustainable development and/or are minimal.

As remarked by [24], there are two ways to interpret the Sustainable Software term which are: the software code being sustainable, agnostic of purpose, or the software purpose being to support sustainability goals, i.e. improving the sustainability of humankind on our planet. In ideal, the two previous understanding coincide in a software system that contributes to more sustainable living. Hence, sustainable software minimizes the environmental impact of the processes it supports, has a positive impact on economic and/or social sustainability and is energy-efficient. These impacts can occur direct, indirect or as rebound effect. Software Engineering for Sustainability aims to make use of methods and tools in order to achieve this notion of sustainable software.

Another point of view about sustainable software is defined by [25]:

- direct and indirect consumption of natural resources, which arise out of deployment and utilization, are monitored, continuously measured, evaluated and optimized already in the development process,
- appropriation and utilization aftermath can be continuously evaluated and optimized,
- development and production processes cyclically evaluate and minimize their direct and indirect consumption of natural resources and energy.

Moreover, sustainable software has been defined several times. We summarize several of these in the following Table 2.1.

Reference	Definition
[24]	Energy-efficient, minimizes the environmental impact of the processes it supports, and has a positive impact on social and/or economic sustainability
[26]	Create reliable, long-lasting software that meets the needs of users while reducing negative impacts on economy, society, and environment
[23]	Software, whose impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software are minimal and/or which have a positive effect on sustainable development
[27]	Create reliable, long-lasting software that meets the needs of users while reducing environmental impacts
[28]	The usage processes of a software system with respect to social and environmental aspects

Table 2.1: Sustainable Software Definitions

Finally, in order to adapt at our particular research area concerning software, we propose our own definition of the sustainable software in the Figure 2.2.



Figure 2.2: Our Sustainable Software definition

Hence, we are able to understand clearly the meaning of the term sustainable software that we can apply, particularly, during the development process of our tool to estimate the energy consumption of software.

In the literature, it is possible to find definitions about the terms sustainable and/or green which are also usually used in the same goal. In our study, we consider both terms differently and we will follow the same approach that we made for sustainable software in order to offer a meaning of the term green software.

2.2 Green

After that we defined sustainable software, we need to understand another term “Green“ which is often linked and included in the sustainability. In our case, as we represent in the Figure 2.3, we consider green software at the inside of the sustainable software.

As we made previously with the term sustainable, firstly, we will give several definitions of the term “Green“ in ICT sector in order to better understand its meanings and impacts on the environment on different sectors to better adapt and give a definition in our study. Moreover, we

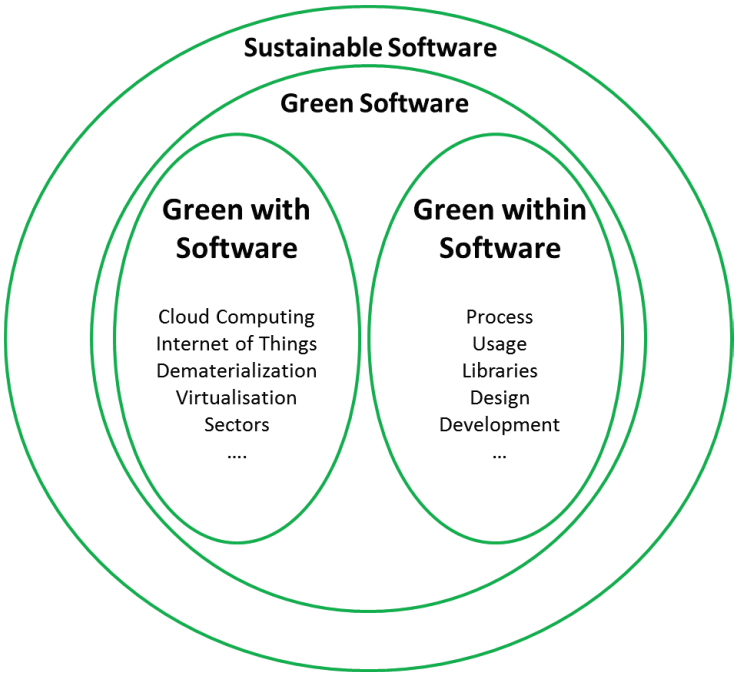

Figure 2.3: Sustainable and Green Software

need to take it into account in ICT in order to reduce the software carbon footprint. Then, we will establish a list of several definitions concerning "Green Software", and we will focus more particularly on the terms "Green with Software" and "Green within Software".

General definitions corresponding to the term "Green":

- In the category greener energy solutions, Merriam-Webster [29] defines the word "Green": is tending to preserve environmental quality (as by being recyclable, biodegradable, or non-polluting).

- According to [30], green information systems which are inclusive of green information technologies meanings: an integrated and cooperating set of people, processes, software, and information technologies to support individual, organizational, or societal goals.

- In [31], Green IT corresponds to the study and practise of designing, manufacturing, and using several devices such as monitors, printers, storage devices, computers, servers and also networking and communications systems effectively and efficiently with minimal impact on the environment. It contains energy efficiency economics, sustainability of environment and the ownership total cost, which includes the disposal and recycling cost. In addition, it consists to create energy-efficient, environmentally sustainable business processes and practices. Information Technology can support, assist, and leverage other environmental initiatives and help in creating green awareness.

- As remarked by [32], in Green Information Technologies, the green corresponds to the environmentally sustainable application of IT. Hence, green is to be interpreted in relation to the environmental problem of climate change and emission of greenhouse gases. Green IT illustrates a situation where IT support greenhouse gas emissions reductions (directly, indirectly or in a systemic way).

- According to [33], the green IT adoption process is a set of four stages which are:
  - Plan.
  - Design.
  - Implement.
  - Measure the performance of the process.

  This process is considered cyclical in nature, characterized by a continuous improvement life-cycle.

- As noted in [34], Green IT is a systematic application of environmental sustainability criteria to the design, production, sourcing, use and disposal of the IT technical infrastructure as well as within the human and managerial components of the IT infrastructure in order to reduce IT, business process and supply chain related emissions and waste and improve energy efficiency.

2.2.1 Green Software

Energy efficient improvement in area of hardware advanced greatly and continue to progressing. Otherwise, those last few years a new trend is developing in the area of green software. Several meanings are proposed for this term and we will examine several of them and then we will propose our own definition related at this term in order to apply in our research efficiently.

As noted in [35], mobile platform manufacturers are seeking to increase the battery life for these platforms. Great improvement have been performed in battery technologies, new low power states have been integrated in processors and displays have performed improvement to reduce their power consumption. Moreover, new improvement are developing. Software can help easily to reduce the power consumed on mobile platforms and also to extend the battery life. Authors present green software features and the software design considerations and methodologies to improve energy efficiency of software.

As remarked by [10], green in software engineering is defined as those practices which apply engineering principles to software by taking into consideration environmental aspects. The development, the operation and the maintenance of software are therefore carried out in a green manner and produce a green software product.

In [36], green software is required to fulfill three abstract requirements:

- The required software engineering processes of software development, maintenance, and disposal must save resources and reduce waste.

- Software execution must save resources and reduce waste.

- Software must support sustainable development.

Reference	Definition
[37]	During the IT lifecycle, produce the minimum waste
[38]	For the optimal use of computing resources, a set of best practices
[39]	The study and practice of designing, manufacturing, and using computer hardware, software, and communication systems efficiently and effectively with no or minimal impact on the environment
[40]	Green IT initiatives can range from those that focus on reducing IT infrastructure's carbon footprint to those that transform a business. Green IT can be deployed to support a variety of sustainability initiatives, such as those to measure carbon footprints, monitor the environmental impact of business practices, reduce waste in business processes, lower resource consumption, or increase energy efficiency and reduce greenhouse gas emissions
[30]	An integrated and cooperating set of people, processes, software, and information technologies to support individual, organizational, or societal goals

Table 2.2: Green Software Definitions

With these most abstract requirements, authors deduce three abstract green factors:

- **Feasibility:** How resource efficient it is to develop, maintain, and discontinue software.
- **Efficiency:** How resource efficient it is to execute software.
- **Sustainability:** How software supports sustainable development.

We summarize several of green software definitions in the Table 2.2.
Based on these previous definitions of green software and taking into account our particular situation, we propose the meanings of the term green software illustrated in the Figure 2.4.



Figure 2.4: Our Green Software definition

2.2.2 Green with/within Software

Several works focused on the inside and outside aspects of green concerning software.

In [41], if we consider that sustainability is similar to green for the authors, we observe a definition of sustainability in/for software engineering based on two aspects which are:

- Development Process Aspect (**Sustainability for software engineering**): Sustainability during the initial software development process meanings development with responsible use of ecological, human, and financial resources. The function associated is: to perform software development with minimized environmental impact and a sufficient economic balance.

- System Production Aspect (**Sustainability in software engineering**): Sustainability of the software system as product with respect to its use of resources for production is achieved, for example, by using green IT principles, sustainability produced hardware components, and optimising the required logistics for assembly, etc. The function associated is: to produce (assemble) a system with minimized environmental impact and a sufficient economic balance.

  The function is needs satisfaction and the two other aspects are also considered about the maintenance and the use:

  - Maintenance Process Aspect: Sustainability of the software system during its maintenance period until replacement by a new system includes continuous monitoring of quality and knowledge management. The function associated is: to maintain and evolve a software system with minimized environmental impact, a sufficient economic balance, and well-managed knowledge.

  - System Usage Aspect: Sustainability in the usage processes within the application domain triggered by the software system as product takes into account responsibility for the environmental impact and designing green business processes. The function associated is: to maintain and evolve a software system with minimized environmental impact, a sufficient economic balance, and social responsibility.

In [42], Green IT 1.0 (Green for IT), study having the main goal to reduce the IT environmental impact of IT and Green IT 2.0 (IT for Green), works trying to reduce the environmental impact of operations using IT, notions are defined.

As remarked by [10], green by software covers software developed for domains that work in the preservation of the environment, as well as software that helps to manage energy-intensive applications. Nevertheless, green in software is related to how to make software in a more sustainable way resulting in a more sustainable product.

As noted, several works have given a similar approach considering the improvement made thanks to software in other fields and the evolution inside the software. Taking into account these works and focusing specifically in our field of research, we want to give a new approach about the terms “Green with software“ (See Figure 2.5) and “Green within software“ (See Figure 2.6). This idea is depicted in Figure 2.3.

2.3 Conclusion

After establishing brief general definitions about the terms related to sustainability and green, we focused more accurately in our concerning area software. We looked at the meanings of



Figure 2.5: Green with Software

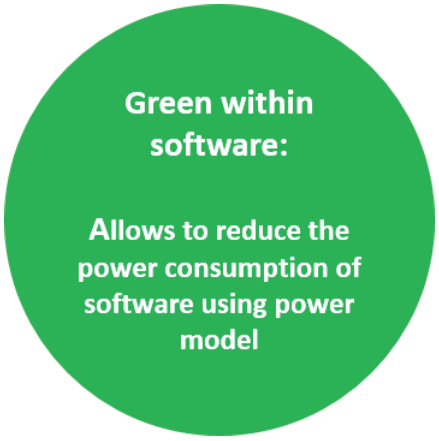

Figure 2.6: Green within Software

several terms to have an idea about the works performed in the ICT area. Then, we extended these proposed works in order to focus more accurately on software area. Hence, we proposed our own definitions of the terms “Sustainable Software“, “Green Software“, “Green with Software“ and “Green within Software“. Thus, the meanings of these terms will allow to build efficient, sustainable and green software product using software engineering process in order to reduce the green house gas emissions due to ICT in keeping the same functionalities of software.

Chapter 3

Related Works

"Software is getting slower more rapidly than hardware becomes faster."

Wirth's law

Several methodologies using different techniques have been developed to measure and estimate software power consumption in academic literature.

Several systems, depending on users needs, have several power supply calculators [43, 44] which enable to select most appropriate components (Motherboard, CPU, Memory, Video Cards, etc.). According to manufacturers information, the peak power consumption of components is taken into account to make assumptions and predictions to calculate the total power consumption. Those calculators are mainly used by gamers to build a modern gaming PC. These online tools give a vague and global estimation of the power consumption of different components. It has been noticed that to build an accurate tool to estimate the power consumption of a computer program is required. Thus, studies have oriented their research directions in this way to develop different methodologies. The most studied and common solutions were based on hardware, since it was possible to obtain accurate results. Then, researchers tried to combine simplicity, scalability and accuracy by developing hybrid methodologies. Finally, the recent research has focused on software methodologies only, in order to be adapted to various environments. Figure 3.1 illustrates the evolution of the proposed power consumption/measurement/estimation models of software.

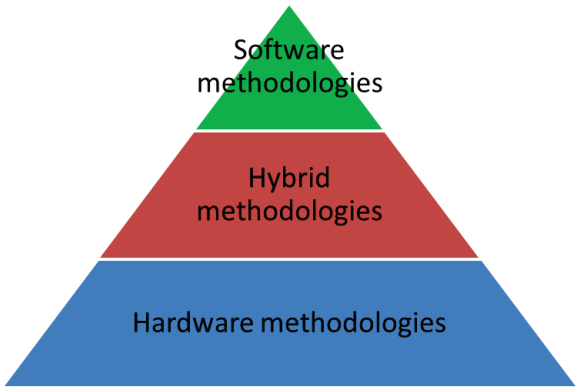

Figure 3.1: Software power consumption methodologies classification

Name	**Accuracy (%)**	**Sampling Speed (/s)**	**Maximum Current (A)**	**Maximum Voltage (V)**	**Average Price (€)**
Kill A Watt	0.2	1 sample	15	125	20
LMG450	0.1	1000 sample	16	600	8000
Watts Up Pro	1.5	1 sample	15	120	130
Brand Electronics 20-1850 CI	1.5	1 sample	15	120	300
CHROMA 66200	0.1	240 sample	15	130	1000
WT1800E	0.05	2000 sample	50	1100	25000
PA3000	0.04	1000 sample	30	600	7000

Table 3.1: Comparison of Powermeters

3.1 Hardware methodologies

In this kind of methodology, a hardware device is used to measure the energy consumption of a specific component or the global system or sensors able to measure the power consumption are connected directly to the monitored component.

Research works which measure the component-based power consumption on hardware methodologies can be categorized in two group.

In the first category of studies, voltages and current powering devices are measured directly with a power meter to obtain the instantaneous power consumption (expressed in Watts) [45, 46]. Powermeter allow to measure and log several information such as power usage, energy consumption, current, voltage, etc. Then, log data can be stored in the memory of the powermeter or send directly at a computer interface with a user set interval in order to use the data for different needs (display or analysis). This approach has given rise to the accuracy of software methodologies. The performance of our proposed power model is also demonstrated with this accurate power measurement technique.

In [47], the authors use a powermeter (Brand Electronics 20-1850 CI) to analyze the power profiles of different hardware components in the context of database operations. They try to examine the power consumption due to these operations to demonstrate the energy saving potentials. For this, they design a set of micro-benchmarks to exercise the hardware components of a database server using typical database-centric operations.

In [48], a WattsUp-Pro power meter providing a 1.5% measurement error rate is used to propose a framework PET in order to reduce database energy consumption in optimization query. Moreover, we can group others power measurement hardware in the following Table 3.1 to compare the accuracy, the sampling speed, the maximum current and voltage and the average price.

In the second category of studies [49, 50], the components that are required to be measured are identified. Then, adapted and customized power sensors are directly connected to these components. Usually, high performance servers use this approach [49].

Name	Description
etop [53]	ASIC (Application Specific Integrated Circuit) is used to measure the power consumption of components. Thus, it is possible to obtain quick and accurate results with this method. However, the hardware component need to be build with the dedicated ASIC and when an upgrade is necessary hence the whole hardware have to be replaced. Moreover, the use of this method will be expensive.
SCHNEIDER ELECTRIC [54]	Active metering is provided by APC metered rack Power Distribution Units (PDUs) in order to allow circuit protection, energy optimization and power utilization data to enable managers of data center to inform about decisions on load balancing and right sizing IT environments in order to reduce at the minimum the total ownership cost.
SynapSense Power Suite^{TM} [55]	It measures power from the level of equipment to the circuit, cabinet or PDU and across the floor of data center to display a power usage efficiency (PUE) view in real time. Taking into account, in a data center, metering, monitoring, trending and tracking power, SynapSense tools for energy management optimizes and manages effectively energy usage of data center. Hence, there are several benefits for the clients which are: reduce of the stranded power, real time capacity identification and optimization of loads.
INA231 [56]	It is a current-shunt and power monitor proposed by Texas Instruments. Voltage of bus supply and shunt voltage drops are monitored by INA231. Conversion times, programmable calibration value and averaging associated with an internal multiplier allow direct power (in watts) and current (in amperes) readouts.

Table 3.2: Studies using hardware methodologies to measure the energy consumption of software

IBM Power Executive [51], including hardware and firmware components, enables IBM customers to manage the power and thermal needs of BladeCenter systems in the datacenter. Server management is improved by Intel API Intelligent Platform Management Interface (IPMI) v2.0 [52] that allows also to minimize costs. In fact, it enables to decrease overall server management costs by allowing clients to maximize IT resources and save time.

It is possible to list more studies using powermeters to measure the energy consumption due to software, we propose several of them in the following Table 3.2 with a bit description of each work.

In general, hardware methodologies are considered to be more accurate than others. However, for a specific program and in virtual machines cases, it is not possible to accurately measure the power consumption. Furthermore, physical measurement results are not sufficient to describe the observed power behavior. On the other hand, these power monitoring circuits consume power themselves. Therefore, we need new approaches to measure the power consumption of devices.

3.2 Software methodologies

Contrary to the other methodologies, software methodology involves, mathematical formulae which is given for each component, with respect to its characteristics. Accordingly, the total power consumption is estimated. Usually, simplifications and assumptions are adopted depend-

ing on the area of study. Moreover, in some cases, several components consumption may be neglected, in the majority of cases CPU power consumption is taken into account, whereas the other components (memory, disk, network, etc.) power consumption are not considered. Therefore, the lack of information and accuracy can cause unsatisfactory and incorrect results. In a power measurement tool, accuracy and completeness are fundamental.

Systematic reviews in related literature have been analyzed [57, 58]. The general research approaches are summarized respecting research type facet in Table 3.3.

Name	Description
Validation Research	Investigated techniques are novel and have not yet been implemented in practice.
Evaluation Research	Techniques are implemented in practice and an evaluation is conducted.
Solution Proposal	For a problem, a solution is proposed. It is possible that the solution is either an important extension of an existing technique or a new approach. A good line of argumentation or a small example describe the applicability of the solution and potential benefits.

Table 3.3: Research Type Facet

The CPU, memory, disk and network observed in Figure 3.2 respectively consists of the studies, where CPU, memory, disk and network are considered to establish a power estimation model of software.

The two bubble plots in Figure 3.2 are drawn in respect to Tables 3.2, 3.2, 3.2 and 3.2 which are representing the tools used to measure the software power consumption. Research type facet, description and limits of each tool is performed.

Therefore, we observe that the majority of studies for computing power consumption takes into account only one component and neglects others. Moreover, the most remarkable research type facet is the solution proposal.

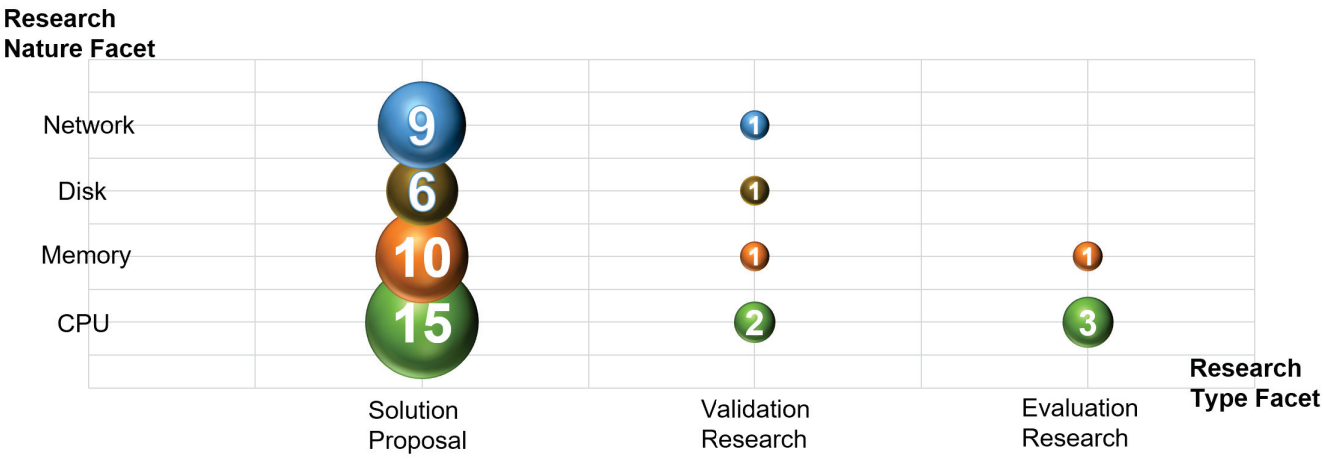

Figure 3.2: Systematic map in a bubble plot of research type and nature facets

Study	Research Type Facet	Description	Limits
CAMP [59]	Solution Proposal	Common Activity-based Model for Power (CAMP) is a technique to estimate activity factors and power for micro-architectural structures.	A tool which estimates only the power consumed by the CPU and does not take account the energy consumption due to other components. Moreover, source code level of energy consumption is not managed.
Framework proposed by Gupta and Singh [60]	Solution Proposal	When the user logs into the system, the framework creates a power profile that starts working. Operating system repository perform the main task of keeping the record of all installed software and maintains the record of power consumed by running software.	Contrary to other tools, this framework does not take account frequency and voltage in the calculation of CPU power consumption and consider CPU Usage, KernelTime, UserTime, UpdateDelay, RawUsage, and RawPower. Hence, a global power consumption of CPU is obtained.
Power model by Bertran [61]	Solution Proposal	The model inputs are defined as the power component activity ratios. Then, the training data are defined which is generated using micro-benchmarks. The required data are collected to train and validate the model respectively. Then, method is used to build the model.	Power consumption is calculated considering micro-benchmark features, activity ratios and power consumption ratios obtained by the manufacturer. Hence, data about component susceptible to consume energy can not be obtained dynamically. Moreover, in the case where the manufacturer does not give these data, it will not be possible to calculate the power consumption.
CiPE [62]	Solution Proposal	Cross-core interference Profiling Environment (CiPE) methodology and framework composed of a lightweight runtime environment on which a host application runs. It manipulates the co-running contention synthesis engine, while monitoring and analyzing the resulting dynamic impact on the host application.	To cross-core interference sensitivity (CIS), a program have to be run only once on CiPE framework which is difficult to manage. CIS is the normalized difference between an application's IPC (instruction per cycle) in the presence of contention and its IPC when it is running alone. Power consumption formulae for component are not established and used.

Study	**Research Type Facet**	**Description**	**Limits**
ECM model [63]	Solution Proposal	the ECM (Execution-Cache-Memory) model is applied to show that the scaling properties of bandwidth-bound codes on a multi-core chip better than a simple bottleneck analysis. Energy behavior to solution is qualitatively explained by the model with respect to the number of active cores.	The dependence of achievable memory bandwidth on the core clock frequency. Power information is hardly accessible.
Sankaran model [64]	Solution Proposal	A statistical approach that models the impact of the cores on overall power consumed by Chip Multiprocessors is developed using Performance Counters.	Only data for CPU cycles and corresponding power consumption are considered. Hence, power consumption for other components is neglected. The impact of source code on energy consumption is not also considered.
Atalar model [65]	Solution Proposal	Models proposed for predicting the throughput and power behavior of lock-free concurrent queues under steady state usage.	Model is not generalized for all data types. Simple tests are executed which is show the lack of accuracy.
NIPD [66]	Solution Proposal	A zero cost, simply a solution based on software. A NIPD (Non Intrusive Power Disaggregation) technique that develops power mapping functions (PMFs) between the servers states and their power consumption, and infer each server power consumption with the aggregated power of the entire data center.	Solution provides power estimation at the rack and server level with a relative error that could be important. The estimation is based on component states, the number of servers and virtual homogeneous clusters, at a given time power consumption, state vector, component state of server and number of server and several coefficients. Hence, only a global power consumption is obtained and power consumption of a particular process is not calculated.

Study	Research Type Facet	Description	Limits
Chen [67]	Solution Proposal	On multi core computer systems, a state-based energy/performance model for a given parallel application. Given the application properties of parallel degree and computation intensity, and the system energy characteristics, the optimal number of cores and the optimal frequencies are derived for the application in order to reach the minimum energy consumption.	Only the energy consumption due to CPU is considered and it depends on several factors which are number of cores, frequency scaling, parallel degree, memory intensity and the performance requirement of the application. The execution time is obtained by estimating the speedup of the first task executed on multi core computers. Hence, the difficulty is to obtain accurate results.
Tudor model [68]	Solution Proposal	A trace driven analytical model for understanding the energy performance of server workloads. The key of this methodology is the modeling of the CPU core degrees and estimating the cores number and clock frequency that optimizes energy performance without compromising execution time.	It estimates the execution time of an application. Time of CPU which accounts for both memory and cores response time. General and system parameters are number of cores, clock frequency, idle, work cycles and stall cycles CPU power consumption. Hence, a global power consumption of the servers is calculated and the power consumption of a given process can not be estimated which limit the accuracy of the model.
CPT [69]	Solution Proposal	For multi core computer systems, CPT (Concurrency, Power and executive Time) is a general energy efficiency model. It is a unified model that allows to decide the configuration of a system in terms of energy efficiency to execute a given workload. The main question is: to perform the task, how much energy is consumed?	Energy efficiency of this model is defined by active idle power of the system and average power dissipation of each thread, the total workload size that is assigned to a system, the concurrency level of the workload and the total time considered to complete the workload. Another principle to reduce dynamic power is reducing frequency and/or voltage of a CPU. Thus, a global model to reduce power consumption.

Study	Research Type Facet	Description	Limits
Anole [70]	Solution Proposal	Anole allows to support energy adaptation. It is a framework developed for designers of operating system and mobile application. When certain energy events happen, the framework supplies several APIs to trigger system and applications to change to different power states. To support the energy aware APIs, a module of energy profiler is developed containing three parts: state collector, energy models and energy estimator.	Estimation of the applications energy consumption used a method which is based on resource utilization. Moreover, User behaviors are not considered to design the default energy aware policies. To support energy adaptation, more energy aware policies should be designed. It is a need to design an energy aware scheduling algorithm with the objectives to control and save the energy consumption.
Safari [71]	Solution Proposal	Safari is a software power profiling tool. Major points have been considered: profiling overhead must be minimized, use power models to profile power using system information and performance monitoring counters.	Information collected are based on the CPU (CPU utilization and frequency, context switch rate, cache miss rate and instruction per cycle). Hence, others components power consumption are neglected which could limit the accuracy of this tool.
Oi study [72]	Solution Proposal	It is a case study of performance power analysis of JVM implementations. The multi threading effects, slower clock speed and the correlation of cache reference parameters to the power consumption are also described.	A global power consumption based only CPU is given and again the other components power consumption are not considered.
David [73]	Solution Proposal	It is a system allowing to run large benchmarks using small memory capacities or storage amount readily available on most computers. David creates a compressed version of the original file system image by omitting all file data and laying out meta data more efficiently.	It is designed to be used for benchmarking. To measure the CPU overhead of the storage model alone, it is runned in the model only mode where remapping, block classification and data squashing are turned off. Only the CPU usage is calculated and the energy consumption is not taken into account.

Study	Research Type Facet	Description	Limits
SPAN [74]	Validation Research	It is a two level power model that estimates per core power dissipation on CMP (Chip Multi Processor) on the fly by using only one PMC and frequency information from CPUs. SPAN is a software power analyzer which identify behavior of power associated with software source code manually. Given an application, SPAN is able to determine its power dissipation rate at the function-block level	Only the energy consumed by CPU is considered and the other components energy consumption is not taken into account. In this case, programmers need to use specific function manually (span_create(), span_open(), span_start(), span_stop(), etc.) to call the SPAN APIs. This task could be not too easy for the developers. Simplicity and adaptability aspects are more emphasized than accuracy, whereas this aspect is more important than the others in order to validate a power monitoring tool.
PowerAPI [75]	Validation Research	PowerAPI is a system monitoring library. It estimates, in real time, the running processes energy consumption based on raw data collected from CPU through the operating system.	The GNU/Linux operating system is used and data are taken from procfs and sysfs file systems. Thus, the tool could not be easily adapted to another operating system in order to get information about CPU. The energy consumed by source code is not taken into account.
Greenspector [76]	Evaluation Research	Greenspector is a software solution for developers with a unique set of green rules for energy-oriented code analysis, a cross-platform measurement capacity and a test bench.	It is oriented for mobile devices and mainly based on power consumed by CPU.
Joulemeter [77]	Evaluation Research	A solution for Virtual Machine (VM) power metering. Power models are built to infer power consumption from resource usage at runtime and identify the challenges that arise when applying such models for VM power metering.	Globally, the energy consumption due to CPU, memory and disk is calculated. However, for a given process, only the energy consumed by CPU is stored. The code source level energy consumption is not analyzed.
Intel Power Gadget [78]	Evaluation Research	Intel Power Gadget is a software-based power usage monitoring tool enabled for Intel Core processors. Monitor and estimate real-time processor package power information in watts using the energy counters in the processor.	Another tool which estimate the power consumption due to CPU and neglect other components power consumption. Moreover, it is not possible to estimate the power consumption of a given process.

Table 3.4: Tools using CPU component to estimate the software energy consumption

Study	Research Type Facet	Description	Limits
Vogelsang [79]	Solution Proposal	A DRAM power model which uses a description of DRAM architecture, technology and operation to calculate power usage and verifies it against datasheet values. Assumptions about the DRAM roadmap are used to extrapolate DRAM energy consumption to future DRAM generations.	DRAM description parameters grouped in six categories: Physical floorplan, Signaling floorplan, Specification, Basic electrical information, Logic block description and Technology are used as power model input. Hence, only the DRAM power consumption is considered. Moreover, the results are validated using datasheet values. Thus, the limit of the model is accuracy.
Gulur [80]	Solution Proposal	An analytical performance model of the DRAM Cache. The model estimates average missed penalty and bandwidth.	The model take into account only key parameters linked to DRAM such as cache block size, tag cache/predictor hit rate, DRAM Cache timing values, off-chip memory timing values and salient workload characteristics.
Chen [81]	Solution Proposal	This study identifies the common micro-components used by internal memory commands and pre-calibrates the power consumption pattern of each micro-component. It decomposes target design architectures into these micro-components to derive power estimation. It considers also the data variation effect by leveraging the fact that memory circuit is mainly doing data passing.	Identify the activated DRAM units of each command and the micro-components. This approach evaluates only the DRAM power consumption and does not consider other components susceptible to consume power.
SMLA [82]	Solution Proposal	Simultaneous Multi-Layer Access (SMLA) is a 3D-stacked DRAM architecture which increases the internal DRAM bandwidth by accessing multiple DRAM layers concurrently, thus making much greater use of the bandwidth that the TSVs offer. To avoid channel contention, the DRAM layers must coordinate with each other when simultaneously transferring data.	How each component within a DRAM scales with clock frequency is determined, and then these observations are used to estimate the energy consumption of proposed designs. The reduction in the frequency is used to reduce the power consumption due to DRAM. Hence, techniques allowing to reduce the power consumption of DRAM are presented.

Study	Research Type Facet	Description	Limits
DRAF [83]	Solution Proposal	DRAM-based Reconfigurable Acceleration Fabric (DRAF) is a substrate for bit level reconfigurable logic that trades off performance of several FPGA devices for major improvements in area efficiency and power consumption.	Estimation for an FPGA and a DRAF device with 75 mm2 die size of the maximum power consumption. DRAM optimizations are performed to reduce access latency and energy consumption. Hence, it is a global approach concerning DRAM power consumption.
DReAM [84]	Solution Proposal	Authors propose a method to distribute the energy consumed in DRAM memories to concurrent running tasks and an efficient implementation of such method. This approach relies on tracking both, the activity incurred by running tasks and the memory bank states they induce. Then, energy is attributed to tasks based on their memory behavior.	Model considers that a device can be in three different states: Power Down (PD), Standby (S), and Active (A). In each state, there is a power consumption. Hence, static and dynamic power consumption of DRAM are calculated together and it is not possible to know the power consumption due to a process.
EFGR [85]	Solution Proposal	This study proposes modifications to the peripheral circuitry of a DDR4 DRAM device so as to expose the non-refreshing banks to the memory controller. In parallel with an ongoing refresh, it accesses a few non-refreshing banks to service memory requests. In the rank, banks must be in pre-charge state to execute a refresh command.	Refresh command is power hungry, any refresh design needs to address during refresh the peak power consumption of device. Only DRAM power consumption is estimated based on the feature of it. Other components are not analyzed.
NUAT [86]	Solution Proposal	To every memory access request, a score is given by NUAT in order to attribute a priority at the request with the highest score. Partitioned Bank Rotation (PBR) and PBR Page Mode (PPM) are used to perform a scoring. PBR is a mechanism that draws access speed data from refresh position and timing. PPM selects a better mode of page between open and close page modes based on the data from PBR.	DRAM carries out three operations which are: write, read, and refresh. To complete each operation, a series of DRAM commands should be issued with pre-defined time intervals. The main goal is to reduce latency which could indirectly reduce the power consumption.

Study	**Research Type Facet**	**Description**	**Limits**
Memristive [87]	Solution Proposal	A model used for solving hard optimization and learning problems. It demands massively parallel computation at a very fine granularity. Control techniques and configurable interconnects eliminate unnecessary latency, bandwidth, and energy overheads associated with streaming the data out of the memory arrays during the computation process.	The model exhibits potential for improving the performance and energy efficiency of large scale combinatorial optimization and deep learning tasks. It is again a global calculation of the power consumption which study only the RAM.
Valero [88]	Solution Proposal	Authors propose a hybrid L2 cache that mingles eDRAM and SRAM technologies to provide by design leakage energy and area savings with respect to typical SRAM caches. To address both performance and energy, the cache controller is designed. Two main design choices have been studied to further increase energy savings : avoid unnecessary destructive reads of eDRAM and estimate the optimal percentage of low-leakage eDRAM banks.	Leakage and dynamic consumption of 512KB and 1MB caches have been analyzed. Energy of leakage includes the consumption of the data array, the tag array and the controller logic. The tag array dynamic energy which is looked up on every cache access and the energy of both data array and controller logic have been analyzed separately. Three steps are considered in the energy consumption: a write access to that bank, a read access to the target eDRAM bank and another write access to the target SRAM bank. It is only an energy consumption analyze of RAM energy consumption.
CACTI-D [89]	Validation Research	It adds support for main memory DRAM chip organization and support for modeling of commodity DRAM technology. It allows modeling of the complete memory hierarchy with consistent models all the way from SRAM based L1 caches through main memory DRAMs on DIMMs.	Using benchmarks with large data sets architectural simulation is performed. The execution time results, power breakdown in the memory hierarchy and system energy delay product for the different system configurations are presented. It is a tool limited to the modelization of the DRAM which does not consider other components.

Study	Research Type Facet	Description	Limits
Micron [90]	Evaluation Research	Over previous DDR and DDR2 SDRAM, DDR3 SDRAM provides additional bandwidth. In addition to the premium performance, DDR3 has a lower operating voltage range. The result can be a higher bandwidth performing system while consuming less or equal power of system. A description of how DDR3 SDRAM consumes power.	The DRAM master operation is controlled by clock enable (CKE) which have to be taken into account HIGH to enable the DRAM to receive PRE, ACT, READ and WRITE commands. When CKE goes HIGH, commands start propagating through the DRAM command decoders and the activity increases the power consumption. DRAM power consumption is calculated considering all parameters given in the data-sheet depending on the state of the DRAM.

Table 3.5: Tools using Memory component to estimate the software energy consumption

Study	Research Type Facet	Description	Limits
Parsons [91]	Solution Proposal	A mathematical hard disk timing model designed for use in an execution driven full system simulator. This model depends far less on the disk structure details or the disk physical layout, making it less complex and easier to configure.	The main goal is to simulate performance of a data-intensive workload. A comparison concerning different disk type have been made to measure the operation per second. Hence, for a particular process, it is not possible to know its associated performance.
Tempo [92]	Solution Proposal	It presents a method to create performance and power model of a disk. Using a trace of all requests issued to the disk with a real time streaming workload, the proposed method is used to estimate the power consumption of the disk.	Disk power consumption is done using a disk request model assuming that data transfer and seek component of a disk request are independent and can be treated separately. The power consumption is estimated during read, write and seek operations.

Study	Research Type Facet	Description	Limits
STAMP [93]	Solution Proposal	STorAgeModeling for Power (STAMP) developed a scalable power modeling method that estimates the storage workloads power consumption. The modeling concept is based on identifying the major workload contributors to the power consumed by the disk arrays.	The methodology is based on three components which are workload translation (translates the front-end workload into the back-end activities), power tables (for each disk activity, the power table is a set of pairs) and interpolation (computes the estimated power consumed for each primitive back-end activity). Thus, an approximate power consumption of the disk is obtained.
TRADE [94]	Solution Proposal	It is a simple methodology for creating hard drive runtime energy models through the use of obtainable data derived from published specifications and performance measurements. Generated models are used to create a TRADE estimator (TRAce-Driven disk drive Energy estimator).	Bandwidth variation and seek time are dependent on several factors (rotation speed and recording density). These characteristics are exploited as the basis for fingerprinting hard drive energy consumption. To calculate energy consumption active and seek power provided by the drive manufacturer are used. Thus, it is a vague energy consumption estimation.
Llopis [95]	Solution Proposal	This work analyzes the power costs of carrying out intra-node data movement and I/O operations. To instrument the platform in order to perform the I/O analysis, Pyprocstat and PMLib frameworks are used.	This approaches combines the software and hardware instrumentation and data analysis techniques to gain insights into how different I/O patterns make use of system resources. Several parameters are considered which reduce the accuracy of this approach.
PCAP [96]	Solution Proposal	PCAP is an agile performance aware power capping system for the disk drive. It dynamically resizes the queues of disk to cap power. It operates in two performance aware modes, throughput and tail latency, making it viable for cloud systems with service level differentiation.	The active read/write mode of the HDD is of a primary interest for power capping. This is because the HDD draws most of the power in the active mode. Power capping may transition the HDD between the active mode and one or more of its low power modes to reduce the average power drawn in a period of time. It is a model trying to reduce the active mode power consumption.

Study	Research Type Facet	Description	Limits
MIND [97]	Validation Research	It is a black box power model for RAID arrays. It is designed to quantitatively measure the power consumption of redundant disk arrays running different workloads in several execution modes. Five modes (idle, standby and several types of access) and four actions have been defined to precisely characterize power states and changes of RAID arrays.	Energy consumption by four types of actions and power consumption in five types of modes are modelized depending on controller and disks. Only disk arrays power consumption are estimated.

Table 3.6: Tools using Hard Disk component to estimate the software energy consumption

Study	Research Type Facet	Description	Limits
WNIC [98]	Solution Proposal	This study explore the WNIC energy consumption implications of popular multimedia streaming formats. The energy consumption under varying stream bandwidth and network loss rates is investigated. History based on client side strategies is explored to reduce the energy consumed by transitioning the WNICs to a lower power consuming sleep state.	Various components of the system setup have been used to analyze the energy consumption characteristics of popular streaming formats such as traffic shaper, browser stations, wireless access point, multimedia server and monitoring station. According to the stream format, the energy consumed by WNIC to receive multimedia streams are compared. However, an analysis studying energy consumption function of the process is not established.
ON and PSP models [99]	Solution Proposal	It is an approach which presents to power modeling and runtime power estimation for wireless network interface cards. Run time power estimates is obtained by putting together four kinds of information: the working conditions, the nominal behavior of the card, its inherent power performance properties and the workload.	Focused only on the UDP traffic reception, two main operating modes are presented: ON (always on) and PSP (power save protocol). The two operating modes can be considered as macro states of a top level state diagram where state transitions are triggered by user commands. The strategy is based on the automatic analysis of current waveforms collected during the execution of synthetic benchmarks.

Study	Research Type Facet	Description	Limits
Sohan [100]	Solution Proposal	It measures and characterizes the idle and active power consumption for a number of production 10 Gbps Network Interface Cards (NICs) of varying makes, models, architectures and utilizing different physical media. The energy efficiency is compared to different configurations.	It measures energy consumption by intercepting the 3.3 and 12v power lines of the PCI-Express connector to measure the current power in the circuit. According to a NIC, idle and active power consumption is measured and it is not possible to distinguish the power used by an application.
Per-Frame Energy Consumption in 802.11 Devices [101]	Solution Proposal	The authors offers an in-depth description of the per frame energy consumption behavior in 802.11 Wireless LAN devices. Extensive measurements are carried out for 7 different type of devices and for both UDP and TCP traffic.	Total power consumed by transmissions is measured. The device total power consumption is articulated into three main components: the platform specific baseline power consumption, the power depending on the airtime percentage and the slope and the power depending on the frame generation rate.
SOLOR [102]	Solution Proposal	SOLOR (Self-Optimizing, Legacy-Compatible Opportunistic Relaying) is a framework which jointly optimizes the network topology which are the nodes associated to each relay capable node and the relay schedules, how the relays split time between the downstream nodes they relay for and the upstream flow to access points.	The 802.11 node power consumption can be modeled after the fraction of time it spends in receive, transmit, idle and sleep modes. Powers consumed by a non-relay node and a relay node are computed in order to calculate the total power consumption. Only the network representation is considered on the estimation of the power consumption.
Budzisz [103]	Solution Proposal	The most important proposals, considered as the two most common wireless access technologies, namely cellular and WLAN are studied.	The power consumption is expressed depending on the number of antennas, the maximum power out of the PA, the power consumed when the RF output is null and the slope of the emission-dependent consumption.

Study	Research Type Facet	Description	Limits
Vageesh [104]	Solution Proposal	This study proposes an RSU (Road Side Unit) placement strategy on a one dimensional road to minimize the total energy consumed. Packet transmission is through either a direct link with the RSU when the vehicle is within its transmission range.	A typical road scenario is characterized by variations in the traffic patterns over time. Each slot captures a snapshot of the particular road scenario at different time intervals. Hence, the energy consumed by the radio unit in the RSU is calculated. It is composed of energy consumed during packet reception and idle power. The global energy consumption is estimated.
Ganji [105]	Solution Proposal	The main goal is to verify if modification patterns in the energy level of a WLAN channel can indicate the user communication attempts. Thus, a detector is proposed whose task is to detect a user attempting to connect to the WLAN by sending a train of probe request frames with a given probability of detection and within a given delay of detection.	Using a WLAN adapter, as well as additional measurement equipment, experiments are conducted to quantitatively examine the performance of the proposed detector in terms of successful detection of the transmission of a train of PRB-REQs (Probe Request frames) in the low SNR regime. It is a special study limited at the area of network.
Chiaraviglio [106]	Solution Proposal	Formulate a theoretical model based on random graph theory that allows to estimate the potential gains achievable by adopting sleep modes in networks where energy proportional devices are deployed. It is not simple to foresee which are the scenarios that make sleep mode or energy proportionality more convenient.	The authors are interested in the energy saving that can be achieved in the transport portion of the network. They model it by an undirected graph depending on the set of transport nodes and links with cardinality. The link and node power consumption are calculated. Thus, the capacity to reduce energy consumption is showed for several cases. The study is limited to network.

Study	Research Type Facet	Description	Limits
Orion [107]	Validation Research	Orion is a power performance interconnection network simulator that is able to provide power features, in addition to performance features, to allow rapid power performance trade-offs at the architectural level.	Dynamic power, in CMOS circuits, the primary source of power consumed is formulated depending on the switching activity, the clock frequency, the switch capacitance and the supply voltage. Thus, only a total network power consumption is estimated, taking into account characteristics of each component in a network.

Table 3.7: Tools using Network component to estimate the software energy consumption

Based on the description and limits established in the Tables 3.2, 3.2, 3.2 and 3.2, we could say that depending on the study case, only several components energy consumption are taken into account, whereas the others are neglected. Moreover, there is a lack of a tool proposing the estimation of software power consumption taking into account several component susceptible to consume energy such as CPU, memory, disk and network. Consequently, we integrate this capacity in our methodology even if the energy consumed by a component can be neglected in several situations. In addition, tools proposing to examine the impacts of source code in the energy consumption of software are very little. Even those which are in this case offer only a manual approach which consisting in integrating several functions to the source code. However, it could be complicated, difficult, boring and expensive for developers to analyze and find the parts of source code that consume the most energy and maybe experts could be needed. Hence, a tool allowing to locate the hotpoints dynamically during runtime of software is needed in order to simplify the tasks of developers.

3.3 Hybrid methodologies

Hybrid methodologies would like to offer both accuracy of hardware methodologies and the simplicity of software methodologies [108]. Even if, several tools have been presented, it is complicated to obtain easily accurate results because it needs many efforts. In several cases, a hardware device such as a power meter or printed circuit is needed to measure power consumption of several components and in addition software tools to manage the data collected by these hardware devices in order to provide accurate results. However, adding several hardware could involve additional energy cost which could be more important usually than the energy saved with the optimization of the source code.

PUPiL (Performance Under Power Limits) [109] is a hybrid software/hardware power capping system where the goal is to combine the timeliness of hardware approach and the efficiency of the software approach . In order to assure a power cap, it navigates through nodes in a decision framework. Each node corresponds to a choice about how to use a particular resource. After establishing a decision, it measures power and performance and uses that feedback to drive the decision at the next node. To evaluate PUPIL, the authors use several benchmarks in order to compare the timeliness and efficiency with other hardware and/or software techniques. Hence, we

observe that the total power is compared to validate the accuracy of PUPIL. However, the capacity to estimate/measure the power consumption of an application depending on components is not implemented. Moreover, the dynamic and static power consumption parts are not separated.

PowerPack [110] framework is a combination of both hardware (circuits, meters, sensors, etc.) components that allow direct power measurement and instrumentation and software components (instrumentation APIs, drivers, etc.) that control power profiling and code synchronization. The goal of these components is to allow component level power measurement and automatic synchronization between power profiles and application code segments. To obtain isolated component power, the authors tap a precision sensing resistor into each individual DC power line and then, using a digital meter, measure the voltage difference at two ends of the resistor. All system DC power lines are measured and used to derive component power depending on a derived mapping between lines and components. To obtain total system power including AC/DC conversion, AC power is measured via an in-line sensor device between the system power cable and the wall. The software contained in PowerPack serves two purposes which are on-line data recording and postmortem data analysis. Hence, the use of this framework seems complicated and expensive because a lot of special devices and experts who are capable to control and manipulate these devices are needed. Moreover, the software part in PowerPack is used more to record and analysis than estimate or measure the power consumption.

GreenHPC [111] is a model that uses a hall effect sensor, on HPC applications, to precisely capture current with an arbitrary time-slice. The proposed framework is adapted to measure power consumption in ARM based processors. GreenHPC tries to calculate the instantaneous power which is the throughput of energy delivered on a specific instant. To compute instantaneous power voltage and current are used. Instantaneous power over time estimates the overall energy consumption of a system, corresponding to the amount of energy used to achieve the solution. Hence, power and energy are estimated. To collect, store and visualize power consumption, the framework use three main components which are: a sensor board in charge of noise filtering and current sensing, a data acquisition board which collects the sensor board data and the voltage from the power source and a Virtual Instrument (VI) in charge of the data processing, visualization and distributed clock synchronization. Thus, the total power consumption is estimated without taking into account the separation between the dynamic and static power and the capacity to consume power by different components.

Finally, this way of measurement methodology is more difficult to establish and it induces an additional power consumption. This may result in a cost increase. Moreover, this methodology could need several special hardware device like printed circuits which could be expensive to adapt at any situation. In addition, to combine efficiently the relation between hardware and software tools, experts should necessarily well implement, run experiments and analyze the results obtained which would be contrary to the simplicity fixed by this methodology. The measured power consumption is the global power consumption without establishing a distinction between static and dynamic power consumption and consideration that each component could consume energy.

3.4 Conclusion

We established a study concerning research proposing solution to measure or estimate software power consumption. We noticed that the studies using hardware methodologies were very numerous and advanced. In fact, in this case it is possible to obtain accurate results. It lead, initially, searches to use powermeters or printed circuits to measure accurately software power consumption. However, the limits concerning the fact that it is not possible to connect a hardware on a virtual machine or to measure a particular process power consumption have led searches to find another methodologies. Hence, works using software methodologies have been proposed to esti-

mate software power consumption more easily and quickly. In this case, mathematical formulae are established depending on the features of components to estimate their power consumption during the execution of software. As we chose to improve this area of research, we performed a systematic literature review on the proposed software methodologies in order to group them in terms of used components to estimate the power consumption. We noticed that often, to estimate software power consumption, only one component is used whereas others are neglected. In addition, dynamic source code analysis is not carried out for several component. Implemented tools based on these methodologies are using, in majority, powermeter to check the accuracy of their results. Otherwise, hybrid methodologies closer to hardware methodologies have also been proposed to measure software power consumption. Hence, software power consumption methodologies have similar limitations to hardware methodologies.

Chapter 4

Sustainable and Green Software Engineering Process

> "Software innovation, like almost every other kind of innovation, requires the ability to collaborate and share ideas with other people, and to sit down and talk with customers and get their feedback and understand their needs."
>
> Bill Gates

The increase in the demand of complex software applications causes additional power consumption for the ICT sector. To limit these greenhouse gas emissions, many efforts have been considered about the effects of hardware components on the environment. However, more and more efforts are needed to perform for more sustainable software products by examining all the steps of a software engineering process with a fine grain approach. In fact, hardware components consume energy, but software, which operates and manages hardware components, has also an indirect energy consumption. Thus, obtaining an efficient software will indirectly reduce the negative impact on the environment. For this, we need to take into account all the a software engineering process steps in order to obtain sustainable and green software products which help to minimize the negative impacts on the environment. Hence, it is possible to help and guide project managers, engineers, developers and users to better collaborate to produce efficient software applications.

In the literature, green software engineering process models have been introduced. In [112], software life cycle is analyzed step by step and the carbon footprint of each step is estimated with some formulae established making several assumptions used for each step. A similar approach is applied in [113], where two levels are defined: First, a software engineering process is defined with each stage. Second, tools are proposed to measure the power consumption of each step. Another proposed methodology is GREENSOFT [114] which gathers the aspects of a software product in four parts: life cycle of software products, sustainability criteria and metrics, procedure models and recommendations and tools. However, this methodology does not study the role of software in order to maintain and optimize the power usage in ICT.

4.1 BUA Methodology

Even if several green software engineering process methodologies have been presented, there is a lack of a generic methodology taking into account all the steps of a software engineering process. Hence, we propose BUA (BeforeUsageAfter) methodology, as a solution taking all the stages of a software engineering process into account like showed in Figure 4.1. We present three levels of

a software engineering process:

- **Before Usage**: This level includes the following steps: requirements, design, implementation and tests. This level will help to establish a first analysis about the software product described by the customer and developed by the manufacturer. By this way, it will be possible to understand if the requirements have been respected in executing all the tests. Then, it will be possible to reanalyze if something is wrong or to continue to the next level.

- **Usage**: This level represents the way to use the software product. The customer is using the software product for his business. The criteria needed for this step are described further.

- **After Usage**: This level consists of maintenance and disposal steps. In this level, it is possible to observe several errors that were not obtained during the Before Usage level and then offer corrections. Moreover, the customer could ask for an extension of the features of the software product to adapt at his new business situation. The customer could also demand a new product based on the previous product in order to have better performance and results for his business.

We include a green analysis step after each step to obtain a green and sustainable engineering process. The green analysis evaluates if the considered step respects all green criteria. If the green analysis does not validate a criteria of a stage, then we return to the previous step, until the requirements stage.

In each step of the software engineering process, there are energy consumed in different shapes depending on the sector for what the software product is following this engineering process. For example: the lights of a meeting room, the time taken to establish the requirements, the calls and mails exchanged between the different participants at the project, the hardware devices used such as computer, tablet, mobile phone, printer, paper, fax, etc., the transports used, upload and/or download software to servers, etc. Thus, each step of a software engineering process is responsible for energy consumption and we propose the following formula (4.1) to calculate the total energy consumed by a software engineering process (SEP):

$$E_{sep} = E_{Requirements}+E_{Design}+E_{Implementation}+E_{Tests}+E_{Usage}+E_{Maintenance}+E_{Disposal}+E_{GreenAnalysis} \quad (4.1)$$

where $E_{Requirements}$, E_{Design}, $E_{Implementation}$, E_{Tests}, E_{Usage}, $E_{Maintenance}$, $E_{Disposal}$ and $E_{GreenAnalysis}$ represents receptively the energy consumed during the requirements, design, implementation, tests, usage, maintenance, disposal and green analysis steps in a software engineering process.

Contrary to [112], we did not propose for each step an energy consumption estimation of the software engineering process, however, we define a generic methodology allowing to reduce the energy consumption of each step in respecting several criteria that we define and describe. Moreover, we validate these criteria by a green analysis step after each step in order to obtain sustainable and green software engineering process. In addition, we focus on the tests stage to propose a generic methodology to estimate the energy consumption of software at runtime.
We define and describe in detail all stage presented in the Figure 4.1 and their criteria. In all situations, these criteria could be applied to obtain efficient results, reduce the energy consumption and improve the performance of a software engineering process.

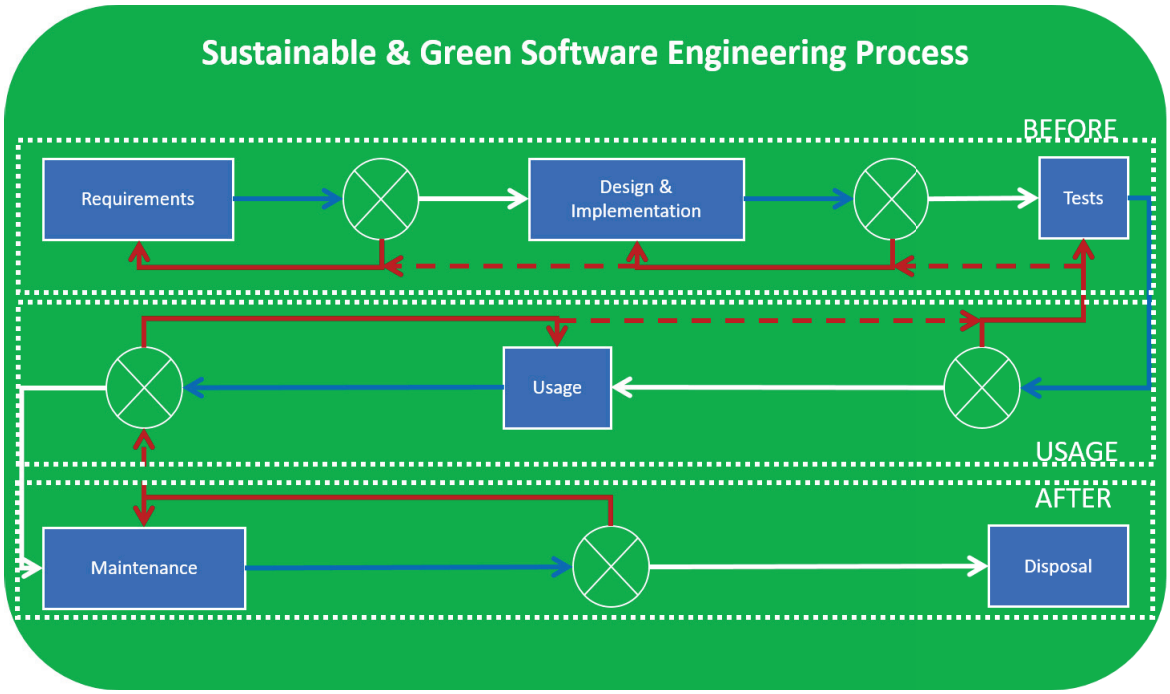

Figure 4.1: BUA (BeforeUsageAfter) methodology

4.2 Requirements

It is the first stage of software development process which consists to document, analyze, trace, prioritize, agree, control change and communicate to all stakeholders on functional and technical specifications. Requirements are established taking into account the propositions of the several source like the customer, the project manager, the developer, the user, etc. To describe requirements many companies are still using document whereas others use software tools. In addition, the deadline of each step of the process should be defined. The aim of a successful software product is to verify and meet the customer needs and expectations. In this step, a continuous communication among all the stakeholders is critical. Its the most important stage of the software engineering process because all needs of the customer have to be very well understood and noticed by the requirements team. Depending on the step of the process, if a mistake or a misunderstanding is observed, then several changes must be performed. Moreover, many efforts will be lost which implies time and money loss and several negative impacts will be generated in the progress of the process. Hence, this step effects all the following steps.

Criteria for Green Analysis:

- **Investigation**: Begin to write the functional specifications when responses are found at the following questions: tools to use to build and manage the requirements, the information to add in the requirements, the goals, the current tools, the constraints, etc. Then, the requirements established need to be approved by all members of the requirements team. Hence, these requirements will be critical to prevent unnecessary changes. In fact, usually clients forget to describe a functionality of the software when the process is in advanced step like tests stage, then a return back to the requirements stage is necessary to take into account this new aspect and change also the design and the implementation stages. In several cases, similarly, the development team forgets to describe a functionality given by the client. Hence, a waste of time and energy is implied.

- **Feasibility**: After to have described generally the software, establish a description about

the software concerning the business requirement of the company. Then, consider the following three aspects of feasibility:

  - **Economic**: Describe in detail the costs implied by the development and running of the software. Then, describe also in detail the benefits of this software product. The necessary budget has to be allocated for this project taking into account the costs due to at each step of the process.

  - **Technical**: The software proposed need to respect specifications. For this, it is important to satisfy constraints of the software. Each technical constraint need to be examined in order to demonstrate that required skills are available to perform each constraint. Moreover, the product needs to be pertinent and useful. For this, a developer team is needed in order to develop the software product. The skills of each developer have to be at a level (expert, junior, beginner, etc.) allowing to develop the product efficiently. Moreover, the hardware equipment must be appropriated to operate the software.

  - **Social**: Consider the effects of the software on the users who will be affected by this new software. Users need skills to efficiently use this new product. For this, schedule training sessions are needed for the users in order to describe all the functionalities of the software. Should we consider the redundancies and / or hiring of people with the appropriate skills? Then, we need to examine the reaction of people about this situation.

- **Tests**: Add at the requirements document a description of all the tests to be performed in order to ensure that the product has been entirely understood. For each requirement, it is necessary to have one or more tests defined to validate the functionality of each requirement. Then, the tests should be executed during tests stage by another person who is not member of the requirements stage team or automatized in a script to save time and energy. Hence, correct, accurate and efficient results should be obtained in order to lead to the next or previous step. At this stage, it should be necessary to also test the software power consumption with a powermeter or a software tool allowing to estimate software power consumption. Depending on the power consumption results obtained, software should be optimized in improving the quality of source code thanks to senior developers' analyses or best development practices techniques.

- **Traceability**: Verify that the company and stakeholders interests are accomplished concerning compliance, completeness, and consistency. Efficiently manage the changes performed about the requirements. For this, the origin of each requirement and change should be noted clearly with the date and the name of the user who proposed it in order to be easily access by anyone. In fact, like there are several people implied in the requirement elaboration, in several cases a functionality asked by the client is not present in the software product because it was deleted by a requirement team member. Consequently, a time and energy waste is provoked.

The Figure 4.2 represents the requirements stage respecting the criteria for green analysis. Once the requirements are defined in a document or with a software tool, it is possible to proceed to the next step in order to design and implement with the better way the requirements established.

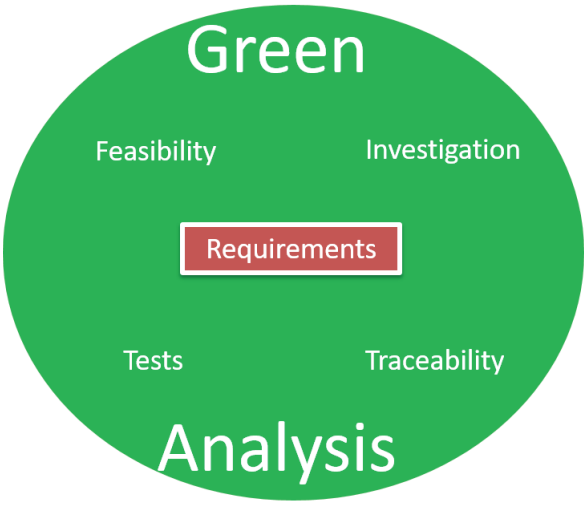

Figure 4.2: Requirements stage with criteria for Green Analysis

4.3 Design and Implementation

Design: Functional and technical specifications defined in the requirements stages are taken into account in order to conceptualize the system architecture and the algorithm. For the same requirement, it is possible to produce several different conceptions. Hence, it is important to choose the efficient design to avoid further problems. Moreover, the concepts and the relationships between them need to be established. For this, a modeling language like Fundamental Modeling Concepts (FMC), Jackson Structured Programming (JSP), Unified Modeling Language (UML), EXPRESS, Systems Modeling Language (SysML), Architecture Description Language (ADL), Service-Oriented Modeling Framework (SOMF), etc. should be used. Design patterns which consist of patterns describing a solution to a problem need to be reused if it is possible during the design step to accelerate the process. The designer has to describe correctly, accurately and efficiently the aspects of software product to build each detail in respecting the criteria below. If the design stage is established by a team, then firstly an uniformity concerning the rules of style and format will be necessary. To reduce conceptual errors, when the design is created, it could be interesting to check the design quality with experts.

Implementation: The programmers write the source code in one or more programming languages respecting the performed design. Expertize is needed by developers concerning algorithms and knowledge about the domain where the software will be used.

Criteria for Green Analysis:

- **Modularity**: Software products should be divided into a number of independent components (modules) that can be mixed in several configurations in order to allow a better maintainability. Each component could be implemented and tested separately and then integrated to ensure a functional software and speed up the process. Hence, modules necessitating little skills in programming language should be developed by beginner developers to improve their knowledge in programming and more complex modules should be implemented by senior developers in order to obtain efficient source code.

- **Organization**: A well defined relationship respecting a hierarchy among the components. Distribution of the different tasks among the team members to obtain efficient outcome.

Schedule the development of each modules with a deadline and then group all modules to finish the software implementation. In several cases a developer can be implement a part of source code which is implemented by another developer. Hence, we need to avoid this waste of time and energy.

- **Extensibility**: New characteristics should be added without negative impacts on the existing software. When the design and implementation are finished and the users are using the software and want to add a new functionality to the software. In this case, the existing design has to be conserved and it should be possible to integrate this functionality without impacting the global design of the software. Moreover, other software functionalities should be protected to avoid defects and errors in the application.

- **Efficiency**: The most appropriate programming language allowing to build optimized algorithms should be chosen. Depending on the demands of the client and taking into account the developers' programming skills, one or more programming language need to be chosen. Several programming languages provide better performance than others, whereas several others are easier to design and implement. Hence, the choice of the language should allow to perform better maintainability of the software and improve the efficiency of the software.

- **Compatibility**: The software product should operate with other products like a previous version of the same software. In fact, when the client asks to add new functionalities to a software, it is obvious that these new functionalities will be added to the existing software. If not, all the software engineering steps should be performed again which would cause a lot of waste. Hence, when the new functionalities are added to the existing software, care should be given to conserve the existing functionalities and not impact them to keep a software operational.

- **Usability**: The user should be able to easily use and manage all functionalities of the software. Usually, in complex software, it is not easy to know all the functionalities of software. Hence, training sessions should be organized in order to describe all the functionalities of the software to the users. This helps to reduce energy consumption and save time because users would perform their tasks only on one click instead of to click in several functionalities which do not correspond to their goals.

- **Reusability**: It should be possible to several developed codes in another projects with minimal changes and also external libraries, as showed in [115], in order to save energy and time. For this, a well documentation about the modules is needed in order to locate the part of source code which could be used in another application or during a development of software evolution.

- **Performance**: The interface should be smooth and respond quickly to the requests of the users. When the software takes time to respond, then the users could perform more errors. Several manipulations will be executed whereas it was not necessary. Moreover, if the performance of the software is not good, then several algorithms calculation could have failed due to errors.

- **Portability**: The software product should be operating in a variety of situations and environments. It should be usable in several operating systems like Windows, Linux, Mac, etc. In several cases when a company decides to change devices (desktop computer, laptop,

etc.) and also change the operating system linked at these devices, it could be difficult to adapt the software to the new environment. However, if it is taken into account to use the same software in multiple operating systems, it will be possible to save time and energy easily.

- **Readability**: The main goal and the operation of the source code should be understood easily by any developer to avoid errors, duplicated code and inefficiencies. It is important to adopt a common style of development among the developers to improve the quality of source code. For instance, when an integer is declared, it is possible to add a prefix i and capitalize the following character at the name of this integer in order to know the type of a variable without a lookup to its declaration. Hence, it is important to define and respect the same rules of development to help new developers discover the source code to adapt easily and quickly.

The Figure 4.3 represents the design and implementation stage respecting the criteria for green analysis. Once the design and implementation are performed, it is possible to pass to the next step in order to test the design and implementation carried out.

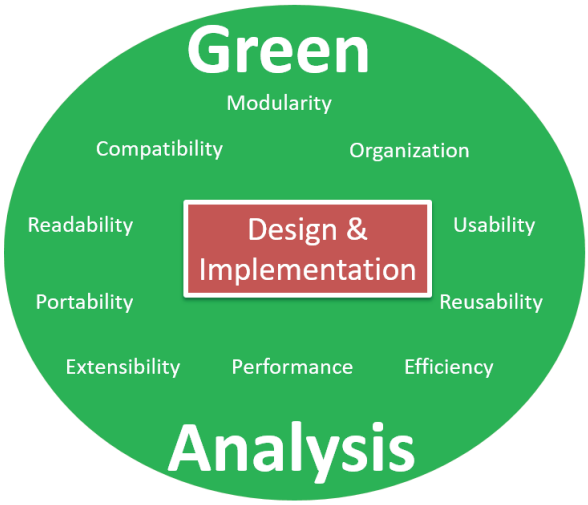

Figure 4.3: Design and implementation stage with criteria for Green Analysis

4.4 Tests

Execute the software with the intent to find errors and check if the software is usable. One or more properties of each software component are evaluated. It allows to validate if the software meets its specifications defined in the requirements stage, to discover faults or defects. Verify if the functions composing software are carried out during an acceptable time. Test results give information to the customer about the quality and efficiency of software. Tests are defined at the end of the requirements stage to show that the specifications have been understood. Use of different testers allows developers to see if the requirements are correct and consistent. There are several types of testing such as: unit, system, integration, black box, white box, path, automation, etc. Depending on the situation, it should be necessary to choose the most appropriate testing type.

Software energy consumption tests consist of measuring/estimating the energy consumption of the software in order to know whether the program can be improved. As presented in Related Works chapter, it is possible to use Hardware, Software or Hybrid methodologies to obtain software energy consumption. Further on we present a tool capable to estimate the software energy consumption and locate parts of source code which are heavy energy consumers.

Criteria for Green Analysis:

- **Planning**: There are many functionalities in a software and to verify the correct operation of each functionality, a schedule of the execution of each test is necessary. Several functionalities should be executed before others, because if a particular functionality does not operate, then it is not necessary to lose time in testing its sub-functionalities. Hence, the order of test execution should be automatized in order to validate each test quickly.

- **Execution**: It should be interesting to automatize tests using script which check each software functionality and send an answer that validate the functionality tested. Then, a report should list unvalidated tests in order to solve the problems associated with these tests. If the script result could give a description about the error, it will faster the problem solving. Otherwise, it is also possible to use testers to execute all the tests described in requirements stage. They can report all the errors in a document in order to correct by the appropriate team until the software release.

- **Analyze**: Concerning the defects found in the software, a meeting is needed between the client and the project team in order to know which errors should be corrected or reported in a next version. For this, the traceability could be used to determine if the situation involving an error was defined. It is important to describe the errors and their impact on the software. Depending on the level of the errors, it could be corrected later or before the deployment of the software. Maybe, the client have not described in detail a functionality involving this error or maybe the development team forgot to implement a functionality causing this error.

- **Functionality**: Software is composed of a set of functionalities to perform several tasks. Hence, all the tests described in the requirements step should be tested and validated in order to show the operation of the software. If not, the requirements could be redefined and then design and implementation could be adapted.

- **Measurement**: Depending on the quality of the source code software performing the same tasks could consume different amount of energy. For this, it should be interesting to measure or estimate the software energy consumption using methodologies defined in Chapter 3 like powermeter or software tool allowing to estimate software energy consumption. In our research, we will further propose a software tool allowing to estimate the software energy consumption and locate in the source code the part which are the most energy consumer in order to help and guide developers to improve their code to obtain efficient, sustainable and green software to reduce greenhouse gas emissions.

- **Validation**: Once all the tests have been executed and validated and the specified requirements are satisfied, then the software is ready to use. If several errors are persisting, then they should be corrected by identifying the causes of the errors and developing solutions to solve these errors. For this, it could be necessary to return back at a previous step to

carry out the adapted changes.

The Figure 4.4 represents the tests stage respecting the criteria for green analysis. When the tests are carried out, it is possible to proceed to the next step to use the software.

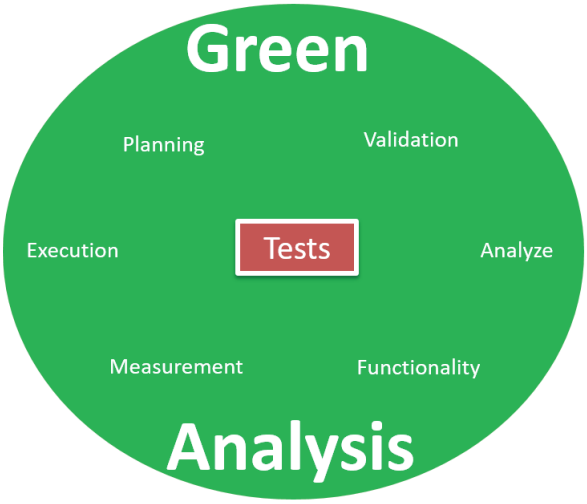

Figure 4.4: Tests stage with criteria for Green Analysis

4.5 Usage

The software needs to be installed and configured in the environment of the customer in order to be used for a business. Hence, the complete software product is released. During this step, the defects detected by the users have to be reported to the maintenance team in order to solve problems. For this, it should be better in term of efficiency, quality and time, if the developers solving the problem were the same who have developed the software because it will be easier to perform the adapted modification without involving any other errors as they know the specifications very well. Otherwise, new developers should understand the functional and technical specifications described in requirement stage to avoid other errors. In this case, it is likely that new defects could appear. Then, the users are notified concerning the fact that the software product is available again. A detailed description of the usage of the software product is needed for the user in order to respect green aspects. There is a dual responsibility for users and engineers to take into consideration. The user should be trained to use the software, because inappropriate manipulations can induce errors in the software.

Criteria for Green Analysis:

- **Ease**: The interface of the software product developed by the programmers have to be easy to be understood and to practice daily by any users. Moreover, a well detailed documentation of each properties of the software is necessary. The different links allowing to access a functionality need to be clear. For example, if one wants to perform a payment, it should be written “payment“ in the button of payment. Any confusion should be avoided.

- **Training**: The users of the software need to be trained at the use of the software by a person having the skills. All functionalities of the software should be presented at the users.

Hence, the users should be experts of this software in order to select the exact functionality to solve a particular need. However, if the users navigate in different menus to find the needed functionality, then time and energy will be lost.

- **Simplicity**: The interface of the software should be the most basic. Each item or link of the menu should be clear to understand the property presented by its. Moreover, each sub menu should be categorized in the similar functionalities and the navigation between the properties these functionalities should be simple.

- **Tests**: The users should execute all the capacities of the software in order to understand the functionalities of the software and also to verify if there are still defects in the software. For this, the software should be deployed in a test platform to perform experiments and train new users. This test platform needs to have all the features of the production platform with the same data. Hence, it is also possible to simulate the errors encountered during the real cases and solve client problems using this application in examining logs or debugging the application. This platform would allow to test the new functionalities of the software before passing to a production platform. Hence, it could prevent defects and errors.

- **Update**: After the defects are solved, replace a previous version of the software with a new release. In order to avoid the errors, it is important to update the software including the new functionalities to satisfy client demands related to these functionalities. During the update, it is important to keep in a file the descriptions related to this update. The new functionalities brought, the errors corrected, the name of the developers who performed the implementation, the date of the deployment. Hence, one should avoid to deploy two different versions at the same time which will imply a crushing of one version.

- **Tracking**: A version tracking system needs to be described in detail to know the developer who installs the new version of the software and the changes brought for this new version. Hence, it could be possible to come back at a previous version if the new version implies many defects. To avoid any error in production platform, it will be interesting to deploy a new version of the software, firstly, in a test platform, in order to validate the changes brought to the software. If the changes are not validated, with the tracking system it will be easy for developers to identify the changes performed in this version in order to locate quickly the errors and correct them.

- **Management**: The users should be aware of the use of the software in respecting green conditions. For this, the users of the software should switch to the most efficient mode (idle, standby, etc.) and/or close the software when it is necessary to reduce the energy consumption. Usually users take a break and let their device and software operating. Moreover, when users finish their work and go to their home, even in this case several users let their device and software operating whereas it will be easy to switch to a less energy consuming mode in order to minimize the consumption of energy. Hence, users need to be careful about the usage of their software to limit the negative impacts on the environment.

The Figure 4.5 represents the usage step respecting the criteria for green analysis. When the users are beginning to use the software, it is possible to report the defects to the maintenance team to have them solved.

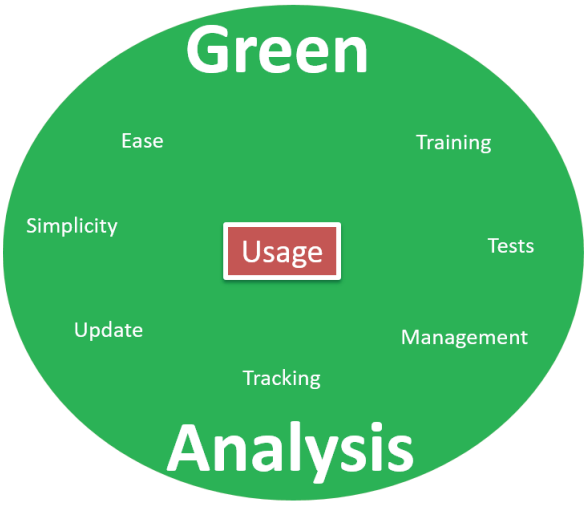

Figure 4.5: Usage stage with criteria for Green Analysis

4.6 Maintenance

This step is realized after deploying a software product, when it is necessary to fix the defects, or to introduce a new version or evolution in the program to improve performance. In this stage, the cost is proportional to the energy waste. Depending on the modifications that the customer needs, it is possible to return to the requirements stage in order to correctly understand the behavior of the program and then propose efficient modifications that do not impact any functionality of the application.

Criteria for Green Analysis:

- **Needs**: The customer needs for a new version or an optimization have to be well defined and described in order to realize a task without generating new errors in the software. Taking into account the previous modifications performed in this software, it should be easier to bring new changes in the software in order to answer the clients needs. Over the time, several functionalities might not used and become unnecessary. Hence, they could be also deleted in the new version of software in order to improve the performance of the software.

- **Analyze**: All the specifications need to be taken into account before bringing modifications. Often, the original developer and whoever brings modifications are not the same person or group. In this case, a particular attention should be given in order to conserve the sustainability and functionality of the software. In fact, it should be avoided to have any impact on the useful functionalities. For this, several tests should be executed to experiment with the new functionalities and also the existing functionalities linked to these new functionalities. Thus, it is possible to save time and energy in avoiding to repeat the same modifications several times.

- **Choice**: Often the importance of this stage is neglected and a basic new functionality to implement is given to a beginner developer. Thus, the error raw is important. So, to increase the speed of the process, decrease the costs and obtain an efficient maintenance, it is advised to choose the most appropriate developer depending on the situation. Usually,

for a complex new functionality, it is better to choose a senior developer to avoid errors or if the development is performed by a beginner then it should be interesting to check its modifications by a senior in order to help and guide the beginner developer to locate his errors and avoid them for the next time. In several cases, it should be interesting to choose a beginner developer for basic modifications in order to help him to improve these programming skills.

- **Tests**: After the modifications asked by the users have been established, then tests concerning these enhancements and the global software functionality should be performed to avoid errors. For this, each test needs to be described clearly. Then, it is preferable if the tests could be automatized in a script which allow to save time and energy. It should be important if the script could send a detailed description about the test with an error result. Thus, it would be easy to locate the error and to correct it quickly.

- **Ease**: The new functionalities of the software should be easy to use for the users. The access to these new properties should be simple in order to conserve the characteristics of the software. Moreover, the modifications of the software should not impact the interface of the software in order to avoid the confusions which could happen during the use of the software. The existing functionalities should be found in the same way and new functionalities depending on their features should be categorized with similar existing functionalities to help users in their usage.

The Figure 4.6 represents the maintenance step respecting the criteria for green analysis. The defects observed during the use of the software are corrected or the new capabilities are added in the software until this product become unusable, it is possible to pass to the next step of disposal.

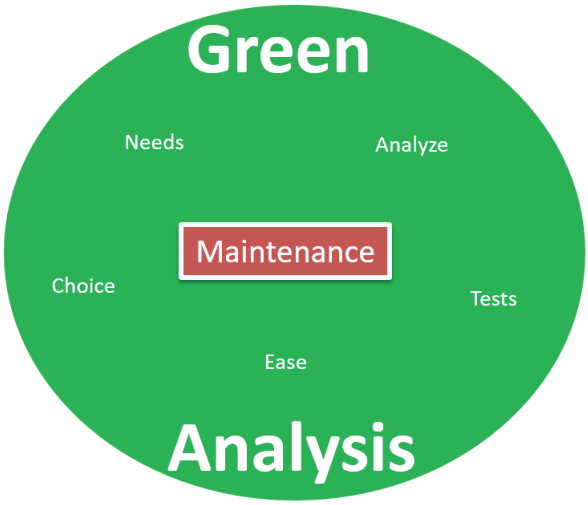

Figure 4.6: Maintenance stage with criteria for Green Analysis

4.7 Disposal

When the software product becomes unprofitable, unused and obsolete, then the software and hardware, running the code, should be replaced. Before this situation, it would be interesting to anticipate this situation in order to collect what is recoverable to reuse efficiently in another

product. It is always better if this step is never reached, or reached after a long period of time in order to benefit entirely of the software product. Moreover, compared to other stage, disposal stage has a lower importance and priority for the following reasons: there is no direct economical benefit associated with the disposal of software, an important investigation should be necessary which involves a loss of time and the necessity to analyze all the software engineering process to understand correctly the task of each part of source code. During this step, information obtained by the software need to be moved, archived or destroyed properly. Hence, a clean migration toward a new software product has to be performed.

Criteria for Green Analysis:

- **Control**: A detailed study concerning the materials used by software should be realized in order to know what material is always operating. In fact, the application is installed in a server and the access for the users is performed by a platform, like in cloud computing or the software is directly installed in a desktop computer. When the usage of the software is finished, depending on the situation if the desktop computer or if the server is very old, then maybe several components could be recovered to use in another device instead of being a waste. If not, the devices could be used to run another software or a new version of the unused software. Hence, it is possible to limit the negative impacts on the environment.

- **List**: A hardware and data information list used by software should be established in order to classify them in category to be recycled, reused or destroyed. When a company perform an activity of e-commerce on-line and this company used a subcontractor to manage the information about the customers. It should be for the interest of the company to keep in a database the data concerning these clients because if the company change the subcontractor, then the company risk to lose a lot of clients and will carry out several efforts to get back the information about its customers. Thus, to save time and energy consumption, it should be interesting to have a list of software data.

- **Recycling**: All the hardware using the software product should be converted into new materials. When a company stops the usage of a software to pass to another software or a new version of the old software, usually old hardware become waste. It would be better to recycle these devices in order to use its in another devices or to improve its performance to continue its usage. If not, it is also possible to give these devices to the associations which are interested to recycle and use these devices differently. Hence, it is possible to prevent the waste of the materials and reduce the energy consumption.

- **Reuse**: A disposal team composed of developers who developed the software could be performed in order to locate parts of code respecting green criteria which could be extracted. In fact, to design and implement sustainable and green source code could take a lot of time which implies a loss of time and consequently energy consumption. Hence, these parts of codes respecting green criteria could be used at best development practices to reuse in another software development project to keep a high quality source code. Hence, it is possible to conserve an eco-friendly software.

- **Responsible**: After all the precautions take in order to be respectful to the environment, if there are still several devices or components which have to be thrown away, it should be performed responsibly by respecting the environment. If the company does not have the means to treat these waste, it should be transfered to an expert company in this area in order to limit the pollution due to these wastes.

The Figure 4.7 represents the disposal step respecting the criteria for green analysis. When this step is accomplished the end of the software product is realized.

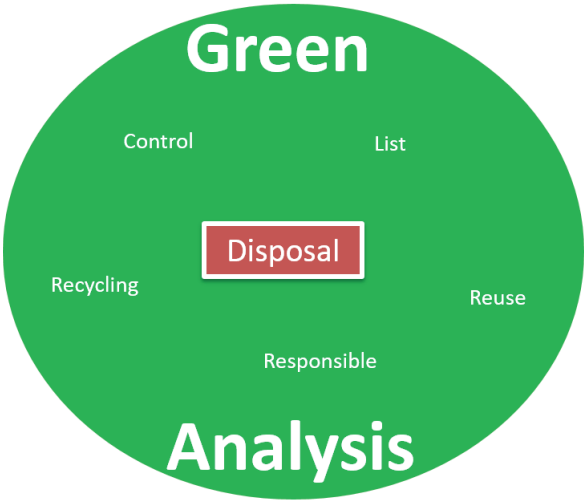

Figure 4.7: Disposal stage with criteria for Green Analysis

4.8 Green analysis

Contrary to other software engineering processes, we add this step, in our contribution, at the end of each step in order to check the sustainable and green criteria defined in each stage. Only when the sustainable and green criteria defined in each step are validated, it is possible to pass to the next step. However, if one or more criteria in a step are not validated, so a come back to a previous step is necessary in order to locate the problem and solve it. The validations of this step after each step will enable to obtain sustainable and green software respecting the definitions that we established in the chapter 2 for each terms.

Thanks to the sustainable and green criteria defined and described in order to perform a green analysis after each stage, we demonstrate the way to obtain a sustainable and green software engineering process respecting the meanings of the terms defined in the chapter 2. Moreover, we would like to focus our research study particularly on the estimation of the software energy consumption. For this, we need a generic methodology to establish a tool allowing to estimate the software energy consumption. Then, we must implement this tool in order to perform several experiments and validate the accuracy of this tool compared to other measurement tools. Hence, developers will improve the quality of their source code in order to build efficient, sustainable and green software product.

4.9 Conclusion

Based on the previous green software engineering process proposed in different works and taking into account the meanings of the terms related to sustainable and green that we defined in Chapter 2, we propose a new approach to manage sustainable and green software engineering process in order to build efficient, sustainable and green software product. In fact, in the classical steps of green software engineering process, we add at the end of each step a green analysis step composed of sustainable and green criteria to respect and validate in order to pass at a next step. If not, a come back to a previous step is performed.

Chapter 5

A generic power consumption methodology: GMTEEC

> "Every piece of system knowledge should have one authoritative, unambiguous representation. Every piece of knowledge in the development of something should have a single representation."
>
> Andy Hunt and Dave Thomas

In literature, various software engineering process methodologies have taken into account green and/or sustainable aspects. However, in related literature, a generic methodology allowing the estimation of the power consumption of given software is not introduced. In fact, the proposed estimation tools are based on a specific approach and have been with severe assumptions. For instance, several tools are limited to an Operating System (Linux, Windows, etc.), whereas others are limited to a programming languages (C, Java, etc.). Thus, in this thesis, a generic estimation methodology for power consumption of software is introduced as a contribution to the related literature. It is represented in Figure 5.1, called GMTEEC (Generic Methodology of a Tool to Estimate the Energy Consumption), which is composed of four layers: business, application, interface and hardware. A researcher needs to consider this power estimation tool during the design and development of software applications. The characteristics of GMTEEC are introduced for each layer, followed by detailed explanation.

5.1 Business Layer

In the Business Layer, we need to locate Hotpoint, i.e. the part of code that consumes the most energy because of the un-optimization or complexity of the source code at this part. Then, we deduce possible quality improvements for software source code in order to obtain optimized source code which consumes less energy and keeps the same functionality.

5.1.1 Hotpoint

Inside the source code of the software, there are objects, methods or part of codes that could consume more energy than others. Therefore, it is important to locate these hotpoints in the source code in order to guide developers to improve their code. Developers could take the opinion of senior developers to know if it is possible to modify the code.

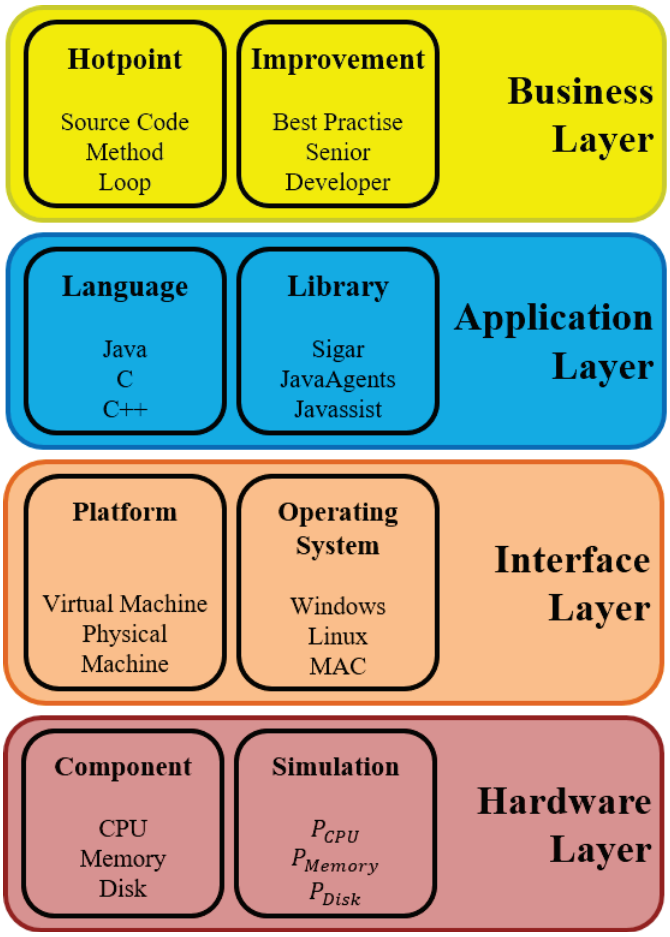

Figure 5.1: GMTEEC

5.1.2 Improvement

Usually, it is not possible to find available senior developers for taking advices. Hence, it will be interesting that the tool, based on best programming techniques identified in a knowledge database, to propose dynamic improvements for source code hotpoints. Moreover, it is better to begin the development of the software with the best practice techniques in order to develop a sustainable and green software. The aim is to solve maximum number of problems with minimum number of functionalities. Programmers having necessary skills and expertise can manage the development of a software in an easier manner. In applying best practices, it is possible to improve the maintainability, readability, performance and energy of code. We can list the following best practices to respect during a software implementation:

- **Description**: Start each file of your project with a description giving information and relation between the concepts used.

- **Packages**: Include only the library needed to run the program. If only several functions are used in a package, it is better to load only the specific associated package and note it in a comment.

- **Comments**: At the beginning of each method, loop or part of source code, a comment is needed describing the use of this code. Hence, another developer could better understand the goal of each code and when a maintenance is need, it will be easier and simpler to

perform changes. However, it is not necessary to overdone comments. In fact, when the code is obvious, it is better to avoid comments.

- **Limit**: Keep each function and/or loop in a respectable line size to read it easily. For this, if a function becomes too long, it is better to separate it in sub-functions.

- **Names**: Give an explicit simple name describing the classes, functions or variables. Like in indentation case, there are several styles to name an object. The most important is to be consistent. Usually underscore is used to separate words, such as '*code_java_name*' or except the first word, the first letter of each word is capitalized like 'codeJavaName'.

- **Unrepeat**: The same code may be repeated several times. In this case, it is better to automate the repetitive tasks.

- **Indentation**: There are variety of styles to indent a source code. The best way is to use the same style used in a project, or in a new project, use always the same style. Hence, the indentation style is needed to be consistent for a project.

- **Organization**: In order to have a simple way to access the information, we need to group the code in files and folders.

- **Separation**: Usually, the data of software are stored separately, e.g. in a database, and source code is used in another field. It is important to separate them to have a better organization.

- **Collaboration**: For the complex projects that are composed of several developers, it is better to understand the developments performed by other developers in order to validate or give advice to improve the quality of source code. In this case, it is better that the source code is validated by an expert.

5.2 Application Layer

In the Application Layer, after we choose an adapted language, we can consider useful libraries to get the information dynamically from hardware components.

5.2.1 Language

There are various programming languages, which have different characteristics depending on the case. We carry out a variety of comparison between them in order to guide developers to choose the most appropriate language according to their situation. Table 5.2.1 represents the language types.

Type	Description	Languages

Type	**Description**	**Languages**
Imperative	It is a paradigm of computer programming with a state and commands which modify the state. It is an abstraction of computers which are based on, with its registers and store, the Turing machine and the Von Neumann machine operating with the design of a modifiable store which is the object that is manipulated by the program. To provide structure to code and to manipulate the store, these languages provide several commands. In each language of this category, a particular hardware view is defined like Java machine or C machine. In these languages types a name used by a value can be used later by another value. The set of names and their values represents the state which is a logical model of storage. A states sequence is generated during an execution. The transition from one state to another is established by sequencing commands and assignment operations.	Ada, Assembly language, C, C++, C#, D, FORTRAN, COBOL, Go, Java, MATLAB, Pascal, Perl, PHP, Python, etc.
Procedural	It is a type of imperative language. It specifies a series of structured procedures and stages in its context to compose an application. In order to complete a computational application, it contains functions, commands and systematic order of statements. It separates an application in functions, variables, statements and operators. To carry out a task, procedures are implemented on the data. Those procedures can be invoked anywhere.	Ada, BASIC, C, C++, C#, COMAL, Java, Julia, Lua, Maple, Plus, R, S-Lang, etc.
Object-oriented	Object-oriented programming (OOP) language is a model organized around data rather than logic and objects rather than actions. Developers can create relationships between objects which can inherit features of other objects. Hence, object-oriented programs are easily changeable. The main advantage is to enable developers to create modules that are not changed when a new type of object is created.	Blue, C++, C#, COBOL, Cobra, D, Eiffel, Falcon, FORTRAN, Go, Groovy, JADE, Java, Objective-C, Perl, PHP, Python, R, S, etc.
Functional	It is a declarative programming paradigm based on expressions or declarations instead of statements. It is the process to build a program by using pure functions and avoiding side-effects and shared state. A pure function is a function where all of its inputs are declared as inputs and all its outputs are declared as outputs.	Aldor, BitC, C++, C#, Clean, Cobra, D, Java, Javascript, PHP, Python, etc.
Logic	It is a programming paradigm. In a formal logic system, program statements describe facts and rules concerning problems . Rules are defined such as logical clauses.	Alice, Ciao, Datalog, DLV, Janus, Prolog, Twelf, etc.

Type	Description	Languages
Scripting	It supports scripts which consist of applications implemented for a particular run-time environment which automate the execution of tasks. It uses a high-level construct to interpret and execute one command at a time. Generally, these languages are easier to learn and faster to code than more structured and compiled languages.	AppleScript, DCL, JCL, Lua, Perl, PHP, Python, Red, etc.
Service-oriented	SOP (Service-Oriented Programming) uses services to design and implement business applications. Contrary to OOP (Object-Oriented Programming), a service-oriented approach allows to developers an easy and simple way to integrate systems. They are less complex, faster, more easily maintained, and much easier to design.	C, C++, Groovy, Java, Scala, PHP, Python, etc.

Table 5.1: Languages Types

The ideal programming language (Java, C, C++, Python, PHP, etc.) depending on the need of the project and respecting several criteria (performance, usage, etc.) should be chosen to design and develop the tool.

5.2.2 Library

In order to obtain a sustainable and green tool, it is required to use libraries allowing to get several information and modify the source code, instead of developing them for several reasons (time, performance, optimization, etc.). Dynamically running libraries should be preferred. We group in Table 5.2.2 libraries depending on the language allowing to get components information dynamically and/or to manipulate the source code.

Library	Description	Languages
SIGAR [116]	SIGAR (System Information Gatherer and Reporter) is a cross platform, cross language library and command line tool for accessing operating system and hardware level activity. It was developed to overcome the lack of portable access to low level hardware and operating system metrics found in the Java platform. It supports multiple language bindings and operates on several OS/hardware combinations.	Java, Perl and .NET
Javassist [117]	Javassist (Java Programming Assistant) enables Java byte-code manipulation simple. It represents a class library for editing byte-codes in Java, it allows Java applications to define a new class at runtime and to modify a class file when the JVM loads it. Unlike other similar byte-code editors, it provides two levels of API which are source and byte-code level. If the source level API is used, then users can edit a class file without knowledge of the Java byte-code specifications.	Java

Library	Description	Languages
ASM [118]	ASM is a framework of all purpose Java bytecode manipulation and analysis. It enables to modify existing classes or dynamically generate classes, directly in binary form. Provided common transformations and analysis algorithms enable to easily assemble custom complex transformations and code analysis tools.	Java
BCEL [119]	The Byte Code Engineering Library (BCEL) goal is to give users a convenient way to analyze, create and manipulate Java class files. Such objects can be read from an existing file, be transformed by a program and written to a file again. An even more interesting application is the creation of classes from scratch at run-time.	Java
psutil [120]	psutil (process and system utilities) is a cross platform library for retrieving data on running processes and system utilization (memory, CPU, network, disks) in Python. It is useful mainly for system monitoring, profiling and limiting process resources and running processes management.	Python
phpSysInfo [121]	It is a open source PHP script that displays system information. It displays CPU, Memory, Uptime, SCSI, IDE, PCI, Ethernet and Video information.	PHP
PDH [122]	Performance Data Helper (PDH) contains a dynamic link library which simplifies collecting performance data from real time data sources. It provides a level of abstraction over the performance registry interface.	C#
/proc [123]	It is a process information file system which contains system information (cpuinfo, system memory, hardware configuration, etc.). Hence, it is considered like a control and information center for the kernel	C, C++
Linfo [124]	Very fast cross platform php script that describes the host server in details, giving information such as ram usage, hardware, raid arrays, disk space, kernel, network cards, OS, temps, samba/cups/truecrypt status, disks, etc.	PHP

Table 5.2: Libraries

According to the chosen language, libraries associated at this language should be used in order to get components data dynamically and then introduce them in the mathematical formulae to estimate the power consumption of each component. Moreover, it is needed to manipulate the source code to locate the hotpoints and guide developers to implement efficient programs.

5.3 Interface Layer

Interface Layer allows to choose a platform, where energy estimation tool can be executed and then adapted to the operating system.

5.3.1 Platform

Software and application are used more and more thanks to the fact that it is possible to execute them above a lot of platform like personal computer, mobile devices or cloud computing infrastructure based on servers and virtual machine, etc. Hence, we need to know the difference between physical and virtual machines in order to choose one of both depending on the case. For this, in Table 5.3.1, we show the advantages and disadvantages of each one when making decision.

Virtual machine		**Physical machine**	
Advantages	**Disadvantages**	**Advantages**	**Disadvantages**

Advantages	Disadvantages	Advantages	Disadvantages
• Easy to manage • Space saving • Easy to backup and restore • Easy to move and/or copy • Compatibility with programs • Virtual CPU and RAM can be easily reduce or extend • A variety of operating system are loaded • Hypervisor is used to create and run virtual machines • Hardware independence • Saving electrical energy • Availability and reliability: a software failure does not impact the other services • Cost reduction • Load balancing: the virtual machine is encapsulated. Hence, it is possible to modify the virtual machine platform and raise its performance • Reduce personnel costs, cooling and power by using less physical equipment	• Management: need to be instantiated, monitored, configured and saved • Direct access to hardware is difficult • An important RAM energy consumption because each virtual machine occupy a distinguish area of the same • Expensive • Compatibility of the applications • An important use of disk space because all the files for each operating system are installed on each virtual machine	• Architecture is simple because it contains its own processor, memory, etc. • Operating system is loaded to run other applications • Provide ease of access and reliability • Compatibility of the applications • Resources used for a dedicated task • Direct access to hardware is simple	• Difficult to move and/or copy • No load balancing technique • Snapshots are not used • Requires a human contact to upgrade hardware • Bound to a set of hardware component

Advantages	Disadvantages	Advantages	Disadvantages

Table 5.3: Virtual and Physical machines advantages and disadvantages

We observe that virtual machine has more advantages than disadvantages and physical machine advantages. However, it is important to take into account all aspects of virtual machine to avoid defects. Based on these criteria, it is possible to choose the platform and then associate it at an operating system.

5.3.2 Operating System

Depending on the goal, it is necessary to choose the most appropriate operating system (Windows, GNU/Linux, Mac OSX, etc.) where the tool will be able to estimate the power consumption of software in taking into account the other characteristics chosen. In Figure 5.2 [2], 5.3 [3], we observe respectively global desktop and mobile and tablet operating system market share obtained by NetMarketShare in May 2017. For the desktop market, Windows operating system, with its all versions, dominates the market, whereas for the mobile and tablet market, Android dominates all the other operating system.

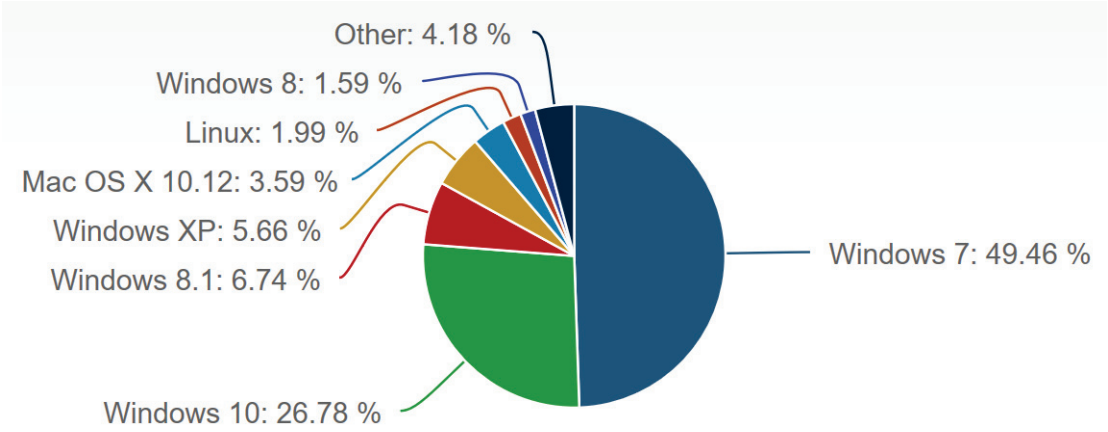

Figure 5.2: Global desktop operating system market share in May 2017 [2]

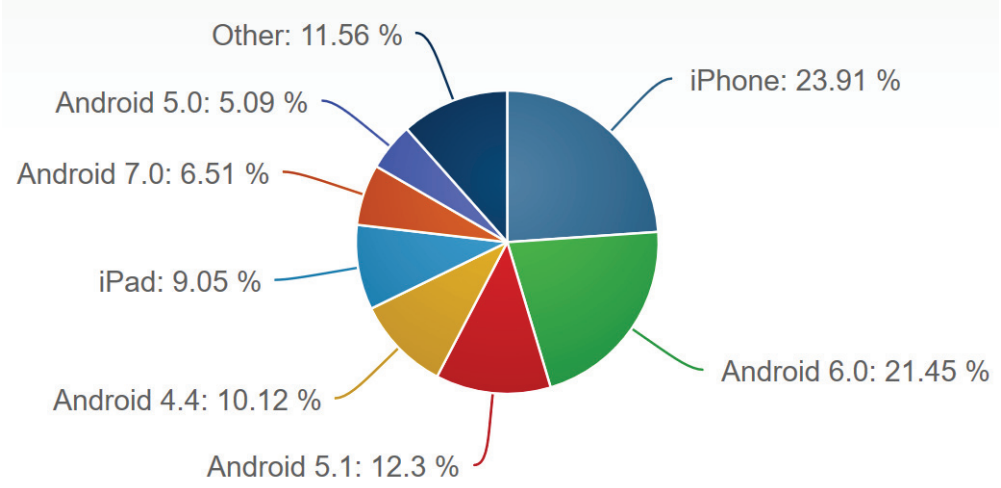

Figure 5.3: Global mobile and tablet operating system market share in May 2017 [3]

5.4 Hardware Layer

In the Hardware Layer, we identify the components that consume power and then we establish mathematical formulae to estimate the power consumption of each component.

5.4.1 Component

In each case, since different components (CPU, Memory, etc.) consume different levels of energy; a fine-grained study is required to identify and understand correctly the behavior of each component in order to obtain accurate results. In Figure 5.4 [4], we see a distribution of peak power usage by hardware subsystem in one of Google's data centers. In Figure 5.5 [5], we observe an average power usage per server distributed by hardware subsystem. Computer power distribution by components is represented in Figure 5.6 [6]. Hence, based on these figures, we remark that each component is susceptible to consume energy which is not necessarily negligible. Thus, it is necessary to take into account all components' energy consumptions in the estimation of software energy consumption.

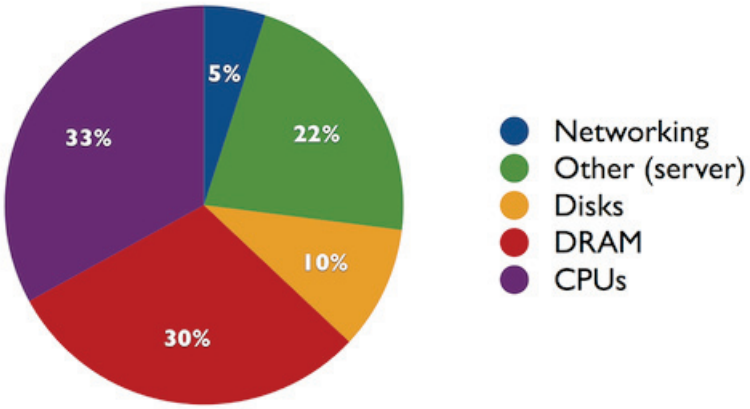

Figure 5.4: Distribution of peak power usage by hardware subsystem in one of Google's data centers [4]

5.4.2 Simulation

According to the device (server, computer, mobile, etc.), each component apt to consume energy during software execution should be taken into account during the software energy consumption estimation. For this, the behavior of each component should be studied very well in order to establish mathematical formulae depending on the characteristic of component which can be get automatically or easily. Then, an accurate global energy consumption of software could be deduced.

5.5 Conclusion

In software engineering literature, various tools have been proposed in order to estimate software power consumption. However, each tool has been built depending on the need of the authors research area. In the majority of cases, only one component has been taken into account to calculate power consumption. Several tools are operating only on Linux or Windows. In this chapter, we introduced four layers of GMTEEC, which are business, application, interface and hardware. We simplified the task of researchers to help and guide them to take into account in each layer several criteria in order to develop the most appropriate tool to estimate the software power consumption. Thus, in business layer, hotpoint needs to be locate to deduce improvement

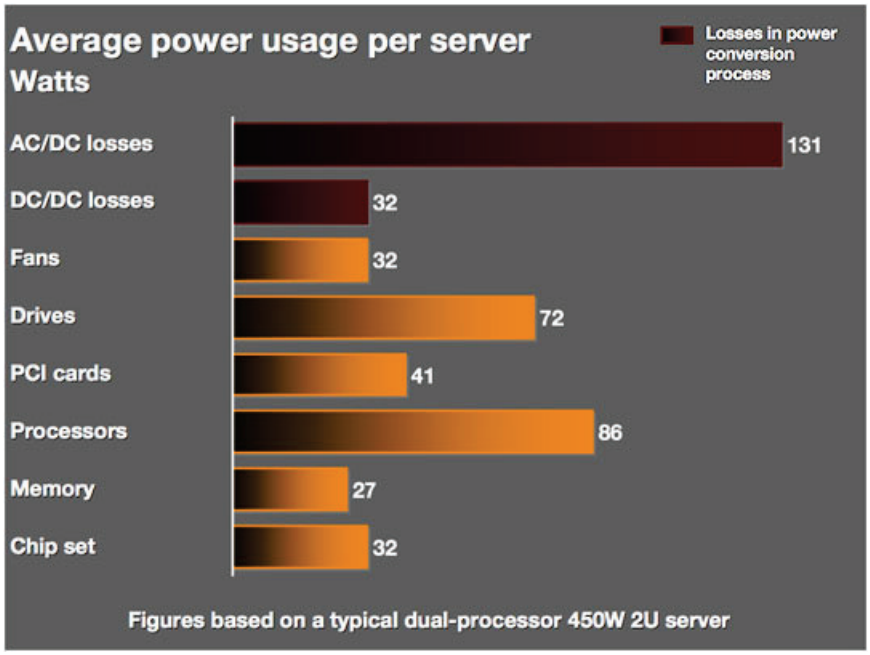

Figure 5.5: Average power use per server distributed by hardware subsystem [5]

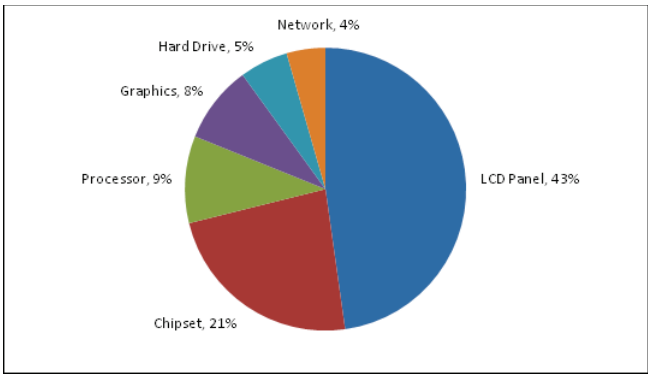

Figure 5.6: Computer power distribution by components [6]

to perform in order to improve the quality and efficiency of source code to obtain optimized source code consuming less energy and keeping the same functionalities. In application layer, the most appropriate programming language needs to be define in order to choose adapted libraries linked to this language to get data dynamically about components susceptible to consume energy. In interface layer, platform needs to be taken into account to choose the operating system adapted to this platform. In hardware layer, components apt to consume energy needs to be identified to define mathematical formulae to estimate the power consumption of each component. Even if it might be difficult, and in several cases impossible, the best way is to design and implement an energy estimation tool which will be able to take into account all conditions defined in each layer described previously. We apply this generic methodology in TEEC to propose the tool described in the following section.

Chapter 6

GMTEEC methodology applied: TEEC

"There is a constant need for new systems and new software."

Marc Andreessen

Respecting the previous generic methodology established in the previous section, we extend the previous work realized in [125] to design and implement a tool to estimate the energy consumption (see Figure 6.1) of software. For this, we developed TEEC (Tool to Estimate Energy Consumption) working on particular Windows operating system, but also on GNU/Linux. We established mathematical formulae, that we will explain in detail below, in order to estimate the energy consumption of CPU, Memory, Disk and Network. We chose Java because it is the most famous and used programming language [126] to get the information concerning the components. Sigar library [127] provides Java agents with the capability to load the content of classes at runtime in the JVM. Javassist [128] library enables to modify a class file when the JVM loads it and/or define a new class at runtime.

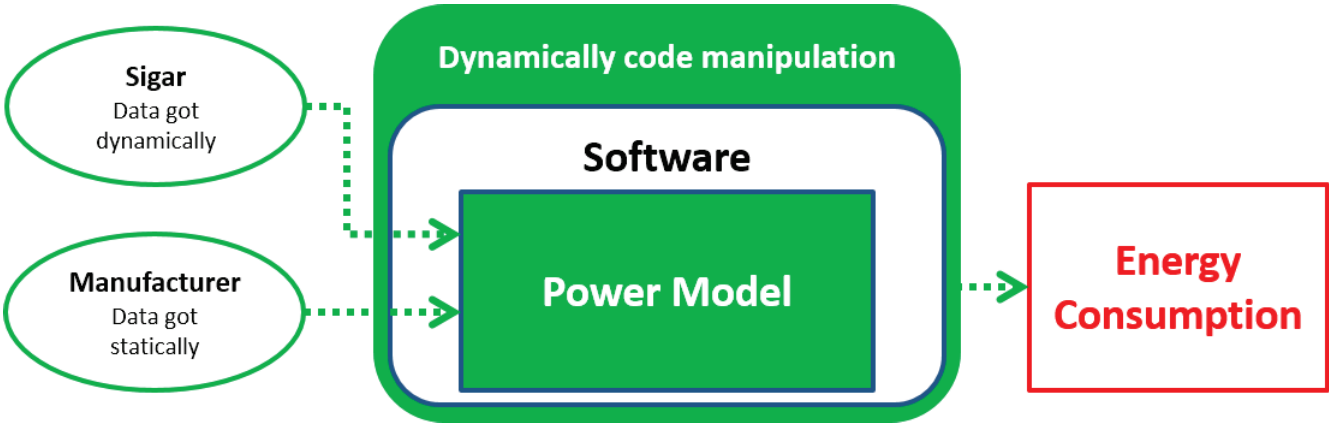

Figure 6.1: TEEC

As we can see in Figure 6.2, using insertBefore and/or insertAfter which are methods of Javassist, it is possible to add several lines of code at the beginning and/or at the end of a method (methodName) contained in a class (className) of a project without to change the original source code. In our case, we add a timer to calculate the execution time of the methodName. Consequently, hotpoints in the source code software are located using Javassist.
We establish power formulae taking into account the parameters that we can get dynamically thanks to the libraries defined previously and variable that are provided by the manufacturers in order to obtain accurate results. The power consumption of software can be separated in two parts:

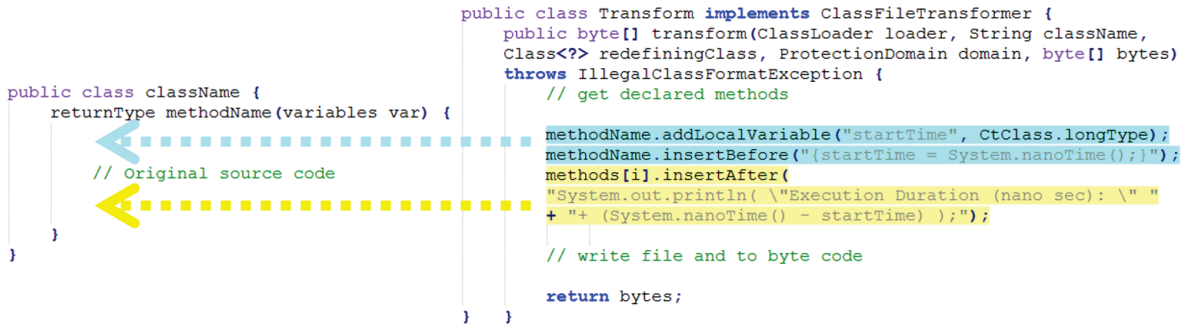

Figure 6.2: Code manipulations

- **Static power**: power to maintain a system running which depends on the quality of the components of devices produced by manufacturer.

- **Dynamic power**: power consumed during a task execution which depends on several factors like clock frequency, voltage and quality of source code.

Thus, we are interested only by dynamic power consumption because it is the only one that we can improve in our research context. Thus, we will establish only dynamic power estimation formulae for the following components.

6.1 CPU

For a long time the CPU was considered the largest energy consumer component [129] in a device. That is the reason why in most of the works, the modelization has taken into account only the CPU to estimate the energy consumed by a program. Several models and factors have been proposed for CPU power consumption, but in this study, we will consider the following equation (6.1):

$$P_{CPU} = P_{CPU,dynamic} + P_{CPU,static} \tag{6.1}$$

where $P_{CPU,dynamic}$ represents dynamic power consumption, while $P_{CPU,static}$ corresponds to static power consumption.

Only the manufacturer can reduce the static power consumption because it depends on the architectural characteristics of each component. In our case, we want to reduce the energy consumed by software. Therefore, we only consider the dynamic power consumption to get more accurate and efficient results.

The CPU, like many integrated circuit, is a set of gates. Hence, the main power consumption in CPU is due to capacitors charge and discharge during computations like respresented in Figure 6.3:

The energy is expressed (6.2) as follows:

$$E_{Vdd} = \int_0^{\infty} i_{Vdd}(t) \cdot V_{dd} \cdot dt \tag{6.2}$$

where i_{Vdd}(t) is the current and V_{dd} is the voltage.

Otherwise, the current is given with the following equation (6.3):

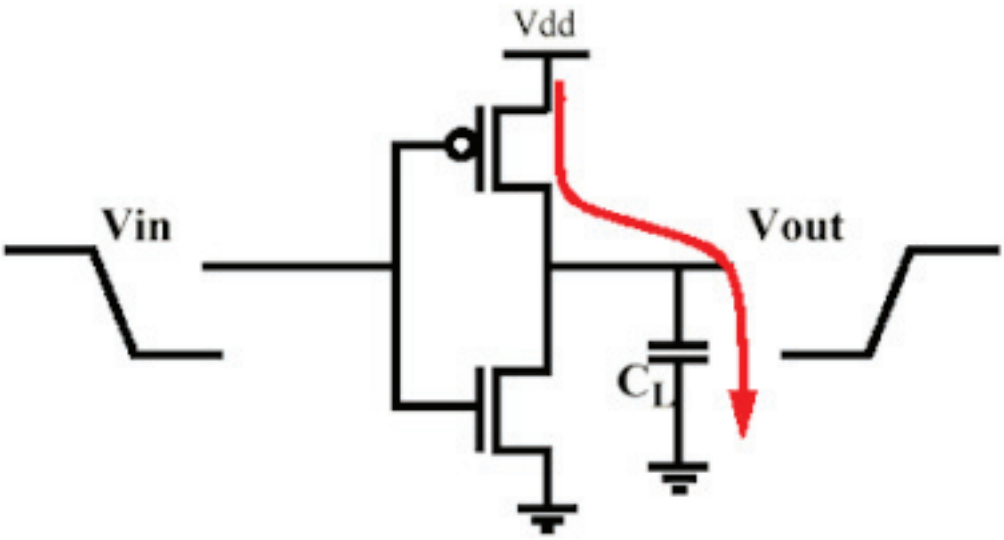

Figure 6.3: One common gate in CPU

$$i_{Vdd}(t) = C_L \cdot \frac{dv_{out}}{dt} \tag{6.3}$$

Thus, the equation (6.2) becomes (6.4):

$$E_{Vdd} = V_{dd}^2 \cdot C_L \tag{6.4}$$

We assume that in a switching cycle, there are low-to-high and high-to-low transitions. Therefore, we obtain the power equation (6.5) of this gate:

$$P = f \cdot V_{dd}^2 \cdot C_L \tag{6.5}$$

where f is the frequency imposed by the clock.

For N gates, the power need to be multiplied by N. Hence, we can define a parameter $\alpha < 1$ as the average fraction of gates that commute at each cycle. The equation of the power becomes (6.6):

$$P_{CPU} = \beta \cdot f \cdot V_{dd}^2 \tag{6.6}$$

Where f is the frequency, V_{dd} is the voltage and β is a constant ($\beta = C_L$.N.α).

In addition, we need the power consumed by the software given by the percentage of the process id (Nid). So, we obtain the equation (6.7):

$$P_{CPU,id} = P_{CPU} \cdot N_{id} \tag{6.7}$$

Thanks to these formulas, we can say that there are several ways to reduce the power consumption due to CPU grouped in the Table 6.1:
Dual voltage CPUs consist of uses a split-rail design to allow lower voltages to be used in the processor core while the external Input/Output (I/O) voltages remain unchanged.

Dynamic voltage scaling (DVS): the voltage used is increased (Overvolting) or decreased (Undervolting) depending upon circumstances.

Solutions	Technics
Voltage reduction	Dual voltage CPUs, Dynamic voltage scaling, Overvolting/Undervolting
Frequency reduction	Underclocking, Dynamic frequency scaling
Capacitance reduction	Integrated circuits

Table 6.1: Possibilities to reduce power consumption of the CPU.

Underclocking: modify timing settings to run at a lower clock rate than is specified.

Dynamic frequency scaling (DFS): the frequency of a microprocessor can be automatically adjusted for saving energy.

Integrated circuits: replace PCB (Printed Circuit Board) traces between two chips.

We defined mathematical formula allowing to estimate the power consumed by CPU. We noted also the different ways to save energy. Thus, we did not limit our work only on the power consumed by CPU, but we take into account also the impacts of other components on the software power consumption even if their power consumption could be neglected in several cases. Hence, we study the characteristics of memory.

6.2 Memory

According to [130], the power used on servers is increasing and two largest consumers of power are the processors and RAM chips. Especially because memory RAM size increase a lot this last decade. Several works [131], [131], [79] have the objective to optimize systems to reduce DRAM power consumption by:

- Using a detailed description of DRAM to calculate power usage.
- DRAM with through-Si-via technology.
- Using the memory controller, a comprehensive approach in order to manage DRAM power.

There are several memory system simulators, such as DRAMSim2 [132] and Cacti 5.1 [133].
In this work, we decide to study the DRAM in order to model its power consumption. Doing so, we use datasheet values from DRAM manufacturer to built the power consumption equation. Similar to CPU, we only interest in the dynamic power, since it is the only part for energy saving. Based on the description performed in [134] we noticed that the dynamic power consumption of DRAM is composed of:

- Activate power.
- Precharge power.

- Read power.

- Write power

In order to model these powers, we need to understand the functionality of a DDR3 SDRAM which is represented in Figure 6.4. The master operation is controlled by clock enable (CKE) that have to be high to enable the DRAM to receive activate, precharge, read and write commands. In this situation, commands begin to propagate across the DRAM command decoders, and this activity increases the power consumption.

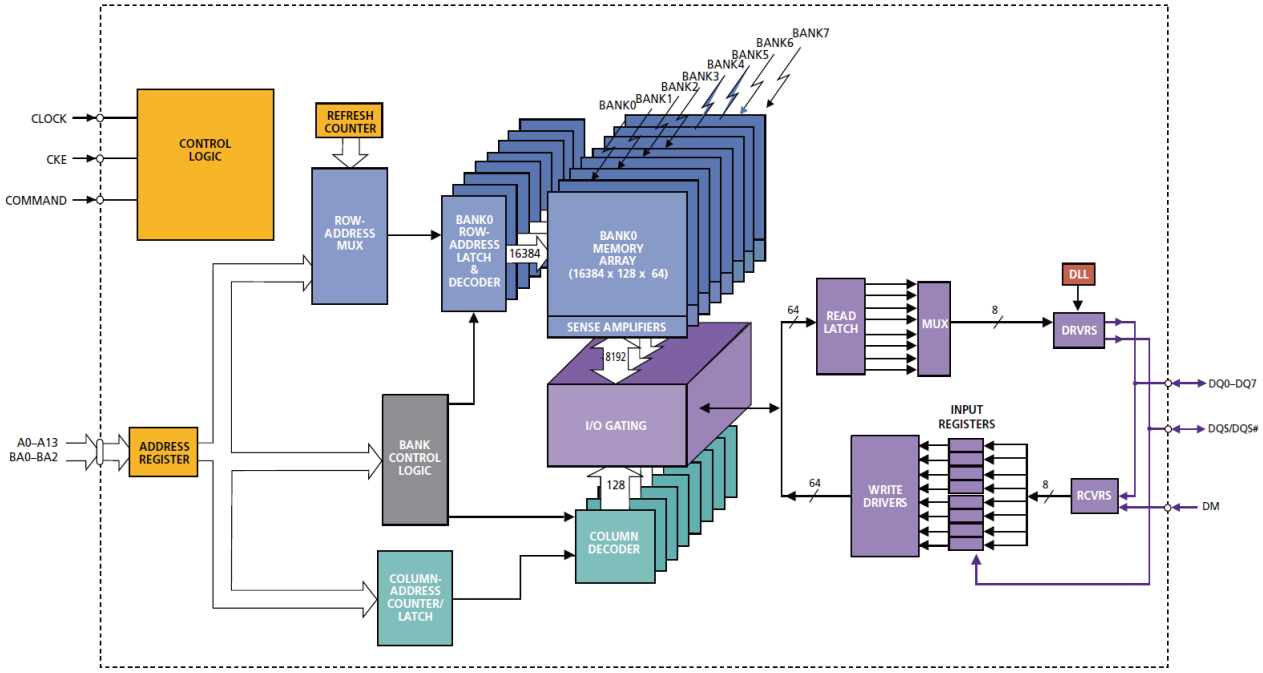

Figure 6.4: The functional bloc diagram of a DDR3 SDRAM

We regroup all the parameters that we will use to calculate the following powers in the Table 6.2.

Parameter	**Description**
Idd2P	Precharge power-down current
Vcc	Voltage
BNK_PRE	The percentage of time that all banks on the DRAM are in a precharged state
CKE_LO_PRE	Percentage of the all bank precharge time for which CKE is held LOW
Vdd	System VDD
IDD2N	Precharge standby current
$syst_ck_freq$	System CK frequency
Tck_used	Used for current measurements
IDD3P	Active power-down current
CKE_LO_ACT	Percentage of the at least one bank active time for which CKE is held LOW
IDD3N	Active standby current

Parameter	Description
IDD0	Operating current: One bank active-precharge
tRAS	Used for IDD0 calculation
tRC	Activate-to-activate timing
tRRDsch	The average time between ACT commands to this DRAM
IDD4W	Operating burst write current
Blength	Burst length
WRsch	The percentage of clock cycles which are inputting write data to the DRAM
IDD4R	Operating burst read current
RDsch	The percentage of clock cycles which are outputting read data from the DRAM

Table 6.2: Data sheet specifications

Accordingly, total power consumption of DRAM is give as (6.8):

$$P_{Memory} = P_{Activate} + P_{Precharge} + P_{Read} + P_{Write} \tag{6.8}$$

We detail each term of the previous equation 6.8.

6.2.1 Activate power

To the DRAM, the first command sent, during normal working, corresponds to a command of activate that chooses a bank and row address in order to enable a DDR3 SDRAM to write or read information. The data, that is stored in the chosen row cells, is then sent from the array into the sense amplifiers. Then, active state is assigned at the DRAM. The precharge command restores the data from the sense amplifiers into the memory array and resets the bank for the next activate command. This leaves the bank in its precharge condition.
Thus, the following expression (6.9) is used to estimate activate power:

$$P_{Activate} = Psys_{ACT_PDN} + Psys_{ACT_STBY} + Psys_{ACT} \tag{6.9}$$

Each term is expressed in the following equations (6.10, 6.11, 6.12):

$$Psys_{ACT_PDN} = IDD3P*Vcc*BNK_PRE*CKE_LO_ACT*(Vdd/Vcc)^2*syst_ck_freq/1000*Tck_used \tag{6.10}$$

$$Psys_{ACT_STBY} = IDD3N*Vcc*(1-BNK_PRE)*(1-CKE_LO_ACT)*(Vdd/Vcc)^2*syst_ck_freq/1000*Tck_used \tag{6.11}$$

$$Psys_{ACT} = (IDD0-(IDD3N*tRAS/tRC+IDD2N*(tRC-tRAS)/tRC))*Vcc*tRC/tRRDsch*(Vdd/Vcc)^2 \tag{6.12}$$

Hence, activate power depends of many factors. Each term of these equations are summarized in the Table 6.2.

6.2.2 Precharge power

Every activate command, that opens a row, have a precharge command, that closes the row, associated with it.
Precharge power is formulated in the equation (6.13):

$$P_{Precharge} = Psys_{PRE_PDN} + Psys_{PRE_sTBY} \tag{6.13}$$

Each term power consumption expression is given in (6.14, 6.15):

$$Psys_{PRE_PDN} = Idd2P * Vcc * BNK_PRE * CKE_LO_PRE * (Vdd/Vcc)^2 * 1 \tag{6.14}$$

$$Psys_{PRE_sTBY} = IDD2N*Vcc*BNK_PRE*(1-CKE_LO_PRE)*(Vdd/Vcc)^2*syst_ck_freq/1000*Tck_used \tag{6.15}$$

Precharge power depends also of several factors that are defined in the Table 6.2.

6.2.3 Read power

During active state, it is possible to read data from or write to DDR3 SDRAM. A read command decodes a specific column address associated with the data that is stored in the sense amplifiers. The data from this column is driven across the I/O, gating to the internal read latch. From there, it is multiplexed onto the output drivers.
Read power is expressed as follows (6.16):

$$P_{Read} = (IDD4R-IDD3N)*Vcc*8/Blength*RDsch*(Vdd/Vcc)^2*syst_ck_freq/1000*Tck_used \tag{6.16}$$

Each term of this equation is also described in the Table 6.2.

6.2.4 Write power

For a write data the power needed is like the read data except the data propagates in the opposite sense. Data from the DQ pins is latched into the data receivers/registers and is sent to the internal data drivers that transmit the data to the sense amplifiers across the I/O gating and into the decoded column address location.
Write power is defined with (6.17):

$$P_{Write} = (IDD4W-IDD3N)*Vcc*8/Blength*WRsch*(Vdd/Vcc)^2*syst_ck_freq/1000*Tck_used \tag{6.17}$$

Each parameter of this formula is also expressed in the Table 6.2.

6.2.5 Total power

Like in the CPU case we need to consider the usage percent of the process id (Mid). Thus, the formula of dynamic power consumption due to Memory for a given process is defined by the equation (6.18):

$$P_{Memory,id} = P_{Memory} \cdot M_{id} \tag{6.18}$$

Hence, we proposed mathematical formula allowing to estimate memory power consumption of software. To obtain an accurate and efficient tool, we need to study the behavior of other component like hard disk.

6.3 Hard Disk

Moreover, in the interest to extend our list of components to take into account, we integrate hard disk power consumption in TEEC.

According to the International Data Corporation (IDC), storage capacity increases each year with a rate of 60% [135]. This increase in storage capacity is likely to increase the proportion of total IT power that is consumed by hard drives.

According to [136], the size of transfers affects power consumption. In light of the observed results, we would like give advise to help hardware and software designers in optimizing data storage and access.

There are several tools proposed to test the performance of a hard drive. The most known are CristalDiskMark and HD Tune [137]. However, these programs allow only measuring read/write speeds. They do not give information about power consumption. That is the reason why researchers tried to take into account power consumption due to hard disk during the runtime. Several of the tools that are built for this purpose are:

- **DiskSim** [138] is a storage system simulator. It is like a reference of disk simulator and it used in many tools in order to obtain adapt several functions and provide power consumption information.

- **Dempsey** [139] has the objective establish power consumption model and performance characteristics of hard disks. In Dempsey, a detailed list of parameters from manufacture are not required, because it is obtained automatically.

- **Tempo** [92] assumes that component seeking and data transfer of a disk can be treated separately. It takes into account of both performance and power in writing, reading and seeking information.

- **SODA** [140], the optimal performance and power tradeoffs are functions of the desired performance and acceptable power consumption.

Taking into account of the ideas in these tools, in order to accurately describe the operation of the hard disk, we first analyze its structure.

6.3.1 Hard disk structure

Hard disk drive (HDD) is an electro-mechanical magnetic storage device. Buffers and digital controllers coordinate and check HDD activities. There are three main power dissipater in an HDD which are the spindle motor, the voice-coil motor and the on-board electronics. Four power management states are considered by hard drives which are: active, standby, idle and sleep.

In analyzing power consumption of the drive, it is important to have a detailed understanding of drives operation and their internal make up.

As in Figure 6.5, hard drives consist of:

- Platters: Data is registered and rotate constantly.
- Spindle motor: It turns the spindle attached to the platters.
- Read and write heads: They collect and send data to the platters.
- Read and write head actuator arms: They suspend the read and write heads above the platters.
- Actuator: It places the arms over the correct location on the platters.
- Printed circuit board electronic.

The platters stop only in standby mode, otherwise they spin constantly. The read/write heads are active while reading and writing,. The arm actuator is active when seeking and residing over locations on a platter. The printed circuit board electronics are always turned on.

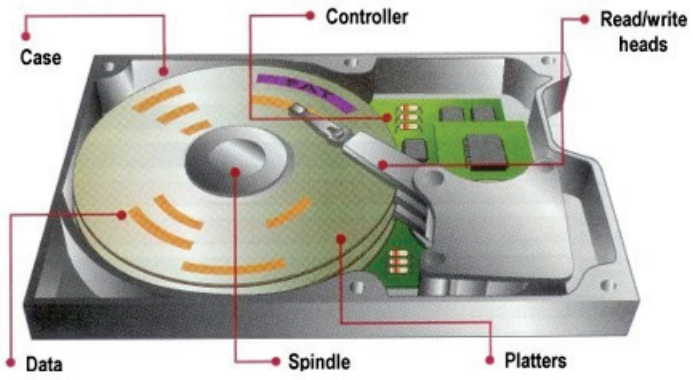

Figure 6.5: Hard disk structure.

The disks are filled with data in unit LBAs (logical block addresses). The operating system do not know the position of LBAs on the disk whereas these blocks are addressed by the operating system. The outer tracks circumference can hold more logical block than the inner tracks because it is larger, as represented in Figure 6.6. Each cylinder in a zone have the similar number of sectors per track. Also transfer speeds vary depending on which zone the heads are in. Data transfer in outer zone is faster than in inner zone.

Two phases represent a disk request process. First phase is the seek phase: moving of the disk head over the track containing the first sector of the request and the rotational latency for the sector to come under the head. Secondly, data transfer phase: the target sectors of the request are read or written. The disk is idle between two consecutive requests.

6.3.2 Power modeling

In active mode, the platters spin and the head seeks or actively reads or writes. In idle mode, a disk spins at its full speed but there is not any disk activity that takes place.

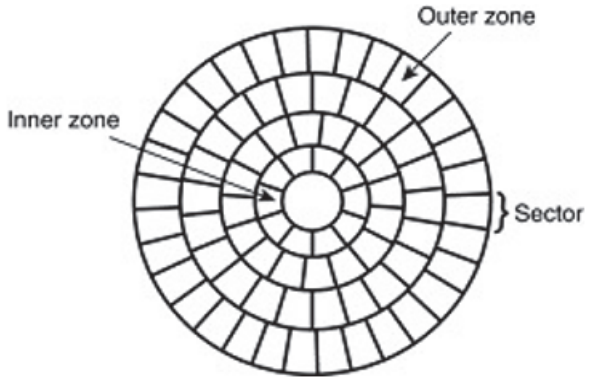

Figure 6.6: LBAs.

Using this model, the power consumption of a disk executing a sequence of requests is simply estimated as the following equation (6.19):

$$P_{Disk} = P_{Active} + P_{Idle} + P_{Standby} + P_{Sleep} \tag{6.19}$$

where P_{Active} corresponds to active mode power, P_{Idle} is the power at the idle state, $P_{Standby}$ represents standby mode power and P_{Sleep} is the sleep power.

Like previous components, hard disk is composed of dynamic (6.20) and static (6.21) power:

$$P_{Disk,dynamic} = P_{Active} \tag{6.20}$$

$$P_{Disk,static} = P_{Idle} + P_{Standby} + P_{Sleep} \tag{6.21}$$

Then, it is possible to assume (6.22):

$$P_{Active} = P_{Read} + P_{Write} \tag{6.22}$$

For a given process, Pread (6.23) and Pwrite (6.24) are defined as:

$$P_{Read} = \frac{P_{DiskRead}}{R_{DiskRead}} \cdot B_{Read} \tag{6.23}$$

$$P_{Write} = \frac{P_{DiskWrite}}{R_{DiskWrite}} \cdot B_{Write} \tag{6.24}$$

where $P_{DiskRead}$ and $P_{DiskWrite}$ are, respectively, the power consumption of disk read and write, $R_{DiskRead}$ and $R_{DiskWrite}$ are the rate of disk read and write, B_{Read} and B_{Write} represents the bytes read and write by the process at runtime.

We establish the mathematical formula to estimate hard disk power consumption of software. An accurate and efficient tool need more component power consumption estimation. Hence, we describe the network.

6.4 Network

We study the network in order to propose a mathematical formula allowing to estimate software power consumption. Like previous components, we are only interested by the dynamic power consumption.

We assume that the dynamic power consumed by the network is composed by the main activity of transmitting and receiving bytes. Thus, we establish the following equation 6.25:

$$P_{Network} = P_{Receive} + P_{Transmit} \tag{6.25}$$

For a given process, $P_{Receive}$ 6.26 and $P_{Transmit}$ 6.27 are:

$$P_{Receive} = P_{ReadBytes} \cdot NetR \tag{6.26}$$

$$P_{Transmit} = P_{TransmitBytes} \cdot NetT \tag{6.27}$$

Where NetR is Network Received bytes, $P_{ReadBytes}$ represents the maximum power received, NetT is Network Transmit bytes and $P_{TransmitBytes}$ corresponds to the maximum power transmit. $P_{ReadBytes}$ and $P_{TransmitBytes}$ are data obtained by the manufacturer specifications.

Hence, software power consumption due to network is estimated with the previous equation established. Then, we need to associate all the component formula to estimate the global software power consumption.

6.5 Total power consumption of all components

We define a global mathematical expression which represent the total dynamic power consumed by software. For this, we sum all the component power consumption formula to obtain the following equation 6.28:

$$P_{Software} = P_{CPU,id} + P_{Memory,id} + P_{Disk} + P_{Network} \tag{6.28}$$

We integrate all the parameters values that are given by the manufacturer and/or that we get dynamically in all these established formulae in order to obtain the global power consumption of software.

6.6 Conclusion

A lot of software power consumption estimation tools take into account only one component and neglect the others to estimate the power consumption of each component during software execution. Hence, we extend proposed previous works in taking into account several components like CPU, memory, hard disk, network depending on the features of each component in order to estimate total software power consumption. Thus, it is possible to have more accurate and efficient results that we can improve in performing needed optimization at the source code level. For this, TEEC allows to locate part of source code consuming the most energy in order to help and guide developers to improve the quality of their source code to build efficient, sustainable and green software product. To check the accuracy and efficiency of our tool TEEC, we conduct a large set of experiments described in the following section.

Chapter 7

Experiments and Validation

"The true method of knowledge is experiment."

William Blake

During the development of our tool TEEC, we performed a variety of different tests taking into account different situations in each case. We began, with the experiment consisting to estimate the power consumption of Fibonacci sequence only for the CPU to validate separately its power model accuracy. We compared the results obtained with Joulemeter tool, developed by Microsoft, allowing to estimate only the CPU power consumption for a given process. Then, we integrated the memory power model in order to see its impacts comparing to the CPU power consumption. For this, we carried out a set of experiments on well known functions. Next, we added in TEEC, the capacity to estimate the power consumption due to disk. We realized again several experiments on several functions in comparing optimize and unoptimize solutions. We validated the accuracy of results with wattsup ?PRO powermeter. We integrated also in TEEC the network power consumption and we realized two different experiments which consisted in searching an integer in a file and students mini-projects. This time we will compare the global power consumption due to CPU, memory, disk and network with two tools and a powermeter to check the accuracy of TEEC. In the following sections, we describe in detail each experiments.

7.1 Fibonacci sequence

First, the proposed tool is tested with a program that requires a lot of calculation, and therefore heavy use of CPU. As the proposed power tool, only measures the power consumed by the CPU, the measurement is more precise and accurate. The Fibonacci sequence is implemented which corresponds to a sequence of integers in which each term is the sum of the two preceding terms. Furthermore, the task manager is seen before and after the execution of the program to demonstrate that only the CPU is impacted. The usage of CPU is observed to increase from a few percent to thirty percent, and it stays around these levels until the end of program execution and returns back to a few percent.

With the proposed power model tool TEEC, the power consumption of Fibonacci sequence using recursive method and iterative method are estimated. The generated test calculates the first 45 values of the Fibonacci sequence with recursive method. For the iterative method, the calculations for the first 5000 values are performed.

The results are represented in Figure 7.1.

The results are compared to the results of Joulemeter application for a particular process with its Id and name (see Figure 7.2).

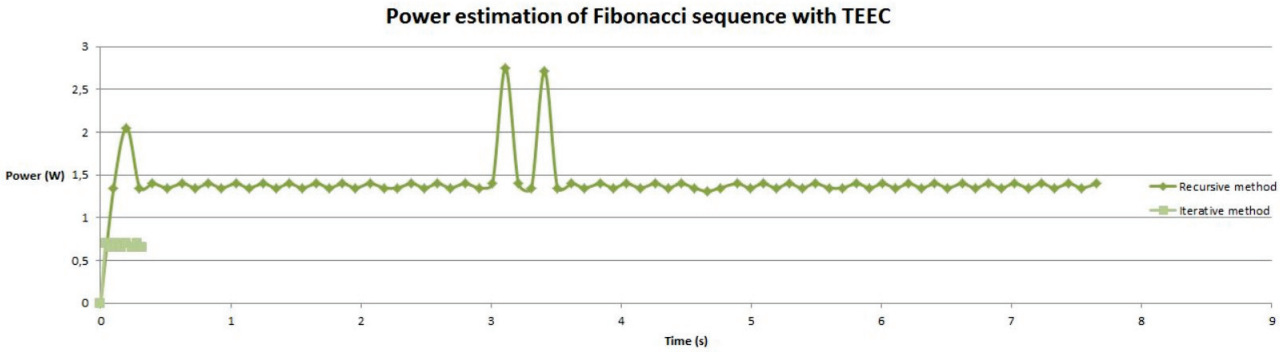

Figure 7.1: Power consumption of Fibonacci sequence with TEEC.

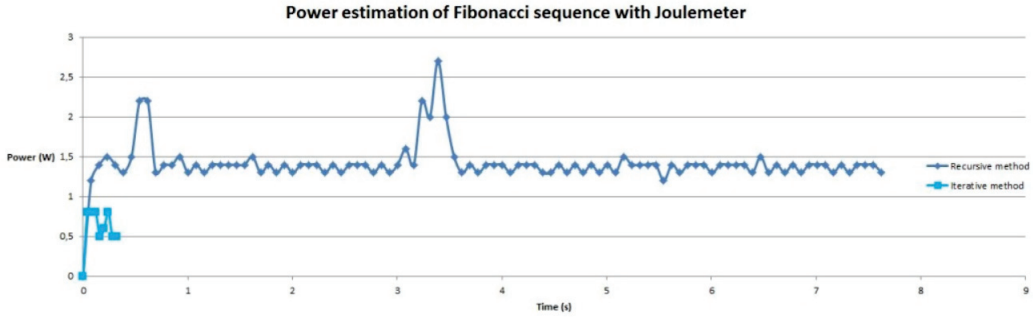

Figure 7.2: Power consumption of Fibonacci sequence with Joulemeter.

First, it is observable that quite similar results are obtained for the running application. It shows the effectiveness of the proposed tool and computational model. Moreover, the results reveal that the iterative method is quicker and consumes less power than the recursive method.

The measures need to be validated on other applications to demonstrate the precision and accuracy of the proposed model. Moreover, we have to take into other component's power consumption estimation.

7.2 Source code adjustment

We realize the following tests in order to see the impacts of source code on CPU and memory power consumption.

7.2.1 Strength reduction

Strength reduction consists of replacing an operation by a similar operation. The most common example of strength reduction is using the shift operator to multiply and divide. For instance, a » 2 can be used in place of a / 4, and a « 1 replaces a * 2.

In our case in order to see the impact of this replacement, we execute the same operation several time (here: 50000 repetitions). The results are represented in Figures 7.3 and 7.4.

For this test, we observe that the DRAM power consumption remains constant in the two cases and the time elapsed is similar. The CPU power consumption is less important after the strength reduction. This shows the impact of the code source on the CPU power consumption. Moreover, in the two cases, we observe that the CPU power varies and in several cases these values are

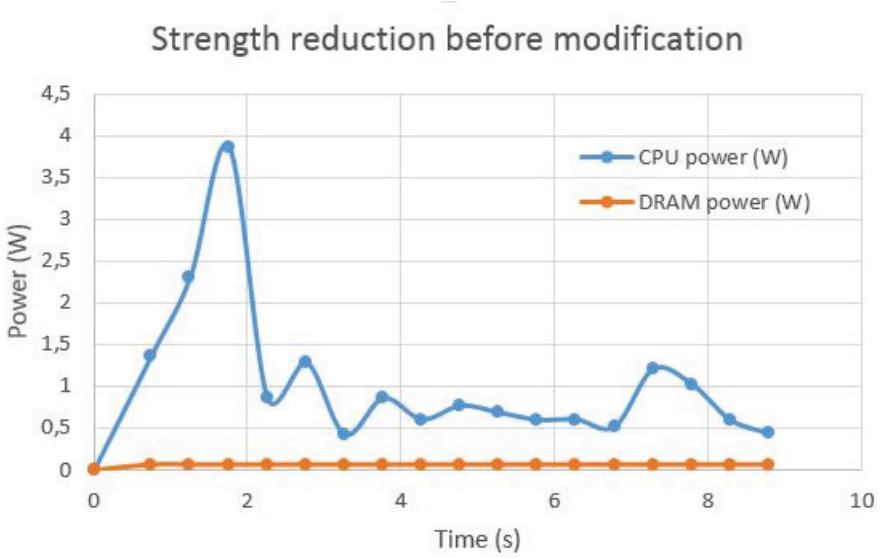

Figure 7.3: Strength reduction unoptimized.

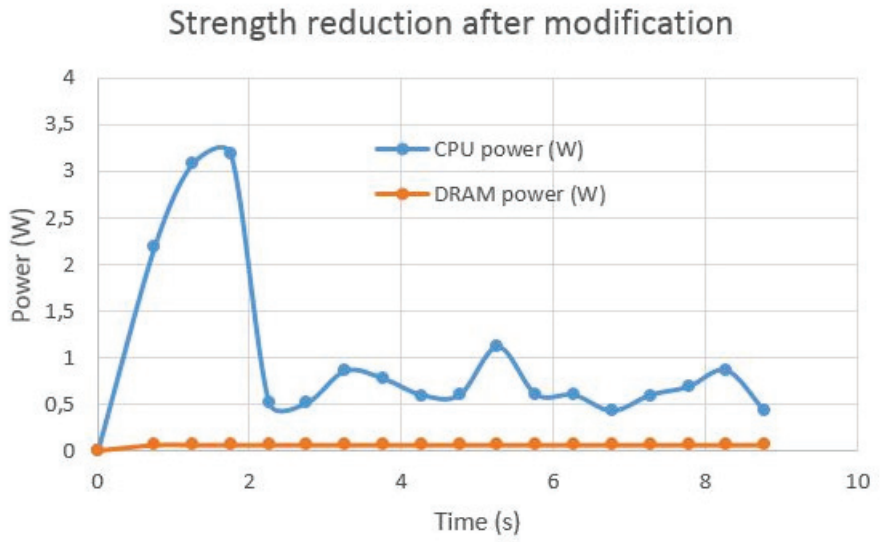

Figure 7.4: Strength reduction optimized.

closer to DRAM values. Thus, we can say that the DRAM power consumption is not always neglected in front of CPU power consumption.

7.2.2 Eliminate common subexpressions

To remove redundant calculation, we eliminate common subexpressions. This part of code:

```
double a = c * (d / e) * f;
double b = c * (d / e) * g;
```

can be rewritten as:

```
double h = c * (d / e);
double a = h * f;
double b = h * g;
```

We run test in a loop of 50000 repetitions to observe the variation of power. The results are showed in Figure 7.5 and 7.6.

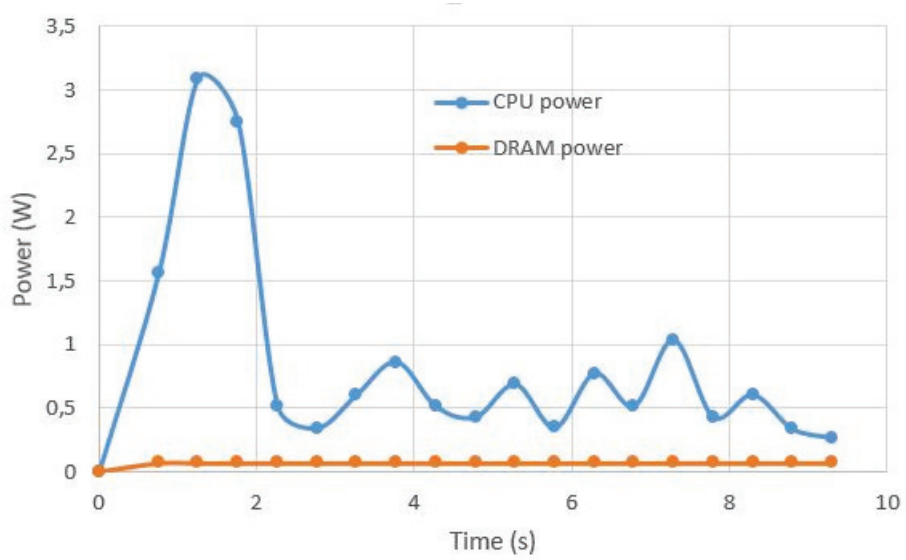

Figure 7.5: Subexpression unoptimized.

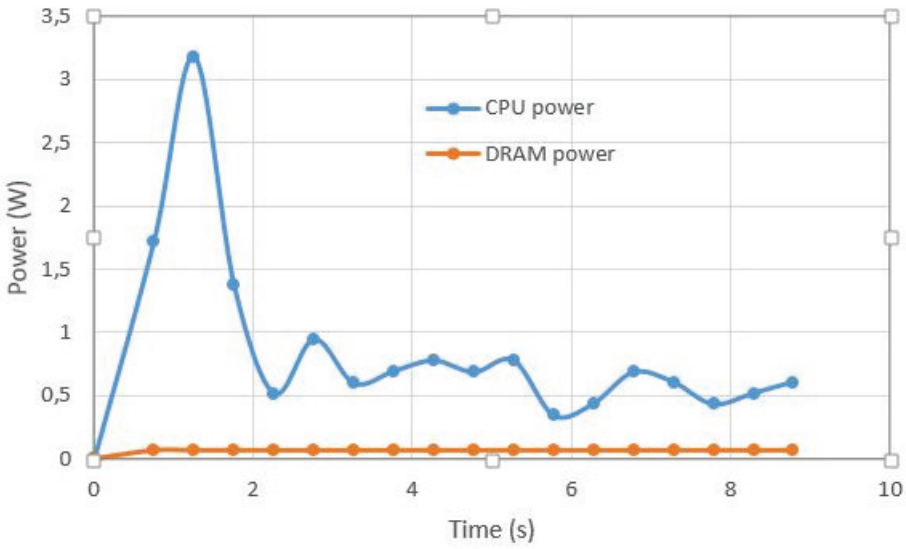

Figure 7.6: Subexpression optimized.

In this test, the results show that the CPU and the DRAM power consumption and the elapsed time in the two cases are quite similar. However, we note that the CPU power consumption vary and several times is more close to DRAM power consumption.

7.2.3 Code motion

Code motion moves code that calculates an expression whose result does not change. This is most common with loops, but it can also involve code repeated on each invocation of a method. For example:

```
for (int i = 0; i < a.length; ++i)
a[i] *= Math.PI * Math.cos(b);
```

becomes:

```
double pico = Math.PI* Math.cos(b);
for (int i = 0; i < a.length; i++)
a[i] *= pico;
```

The results of this test is represented on the Figures 7.7 and 7.8.

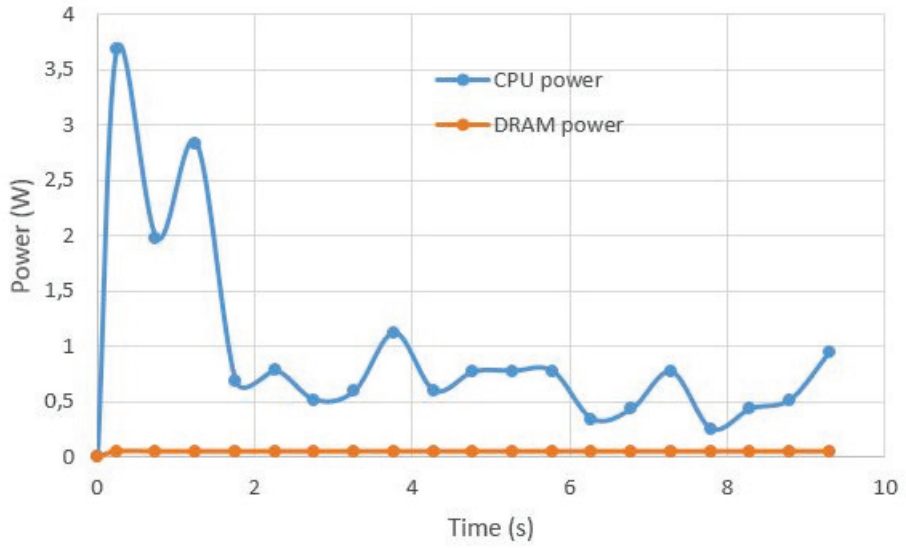

Figure 7.7: Code motion unoptimized.

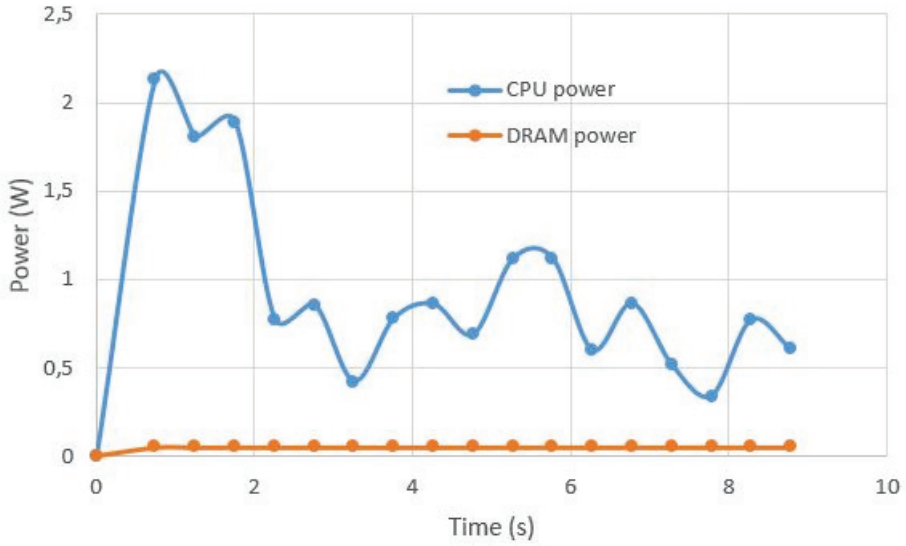

Figure 7.8: Code motion optimized.

This test show that in the unoptimized code motion, the time elapsed is slightly greater than optimized code. CPU and DRAM power consumption are quite similar in the two cases. In several cases CPU power consumption curve approaches DRAM power consumption curve.

7.2.4 Unrolling loops

Unrolling loops reduces the number of loop control code by performing more than one operation each time in the loop, and consequently running fewer iterations. With the previous example, if the length of the table a is always a multiple of two, the loop can be rewrite like:

```
double pico = Math.PI* Math.cos(b);
for (int i = 0; i < a.length; i += 2)
a[i] *= pico;
a[i+1] *= pico;
```

Figure 7.9 shows the power consumption of CPU and DRAM depending on the time.

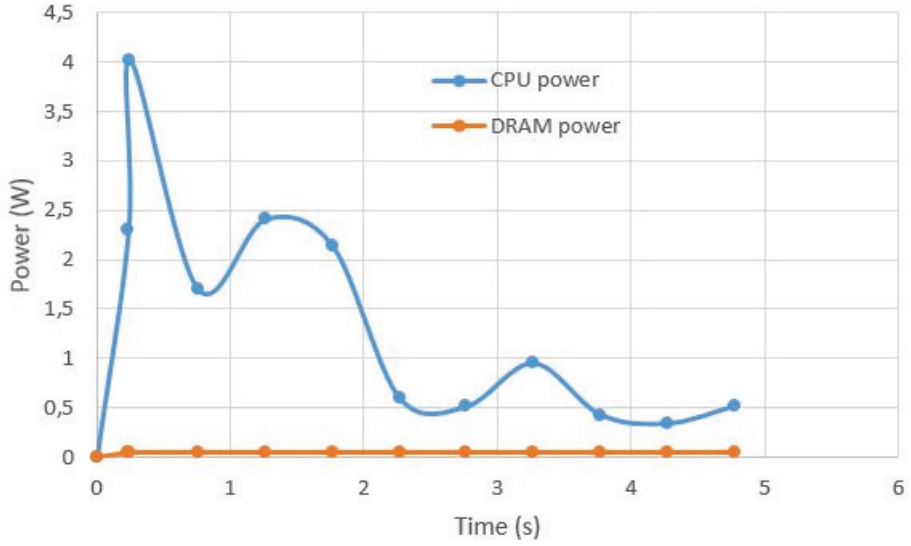

Figure 7.9: Unrolling loops.

Compare to the Figure 7.8, in this case, we observe that at the beginning of the curve, the CPU consumes more power than code motion during several time and then becomes similar. However, in unrolling loops case, the total execution elapsed time is the half of the code motion case. And at the end of the curve in Figure 7.9, the CPU power is less important than the curve in code motion (Figure 7.8). Moreover, in this test, the difference between CPU and DRAM power consumption is less important than code motion case.

Thus, the results reveal that the unrolling loops method is quicker and consumes less CPU power than the code motion method.

According to the previous test, we conclude that the power consumption due to memory is not always neglected compared to CPU power consumption. Hence, the need to take into account the memory power consumption in order to have an efficient and accurate tool to estimate the software power consumption. To extend our study, we need to perform other test taking into account other component power consumption.

7.3 Several function optimizations

Using TEEC, different tests have been executed with unoptimized and optimized methods in order to observe the variation of the power consumption due to the CPU, the memory and the

disk and compare them.

7.3.1 Tests Description

Loops have an important effect on the performance of a program and provide efficient way for repeating a piece of code as many times as required. Java has three types of loop control structures which are: while, do-while and for. If we do not know the number of required iterations, then while loop can be used. The do-while loop is always executed at least once and then the condition is checked at the end of the loop. If we know how many iterations are required, then we use for loop.

Therefore, it is interesting to study several methods that are used during a development of a program in order to analyze possible improvement.

Table 7.3.1 shows the difference between optimized and unoptimized source code.

Array copy

It is better to use an int data type than byte or short data types for a loop index variable, because of its efficiency. The fact to use byte or short data type as the loop index variable involves implicit type cast to int data type.
It is always efficient to copy arrays using System.arraycopy() than using a loop.

Locality of Reference

Elements close to each other in memory are faster to access. We can observe this principle with the programs described in Table 7.3.1. Locality of reference in an array is used.
In the unoptimized version, the loop reads the values of 100 elements in an array. In the optimized version, the loop loads 100 elements, but they are spaced 100 elements apart from each other.

Array and Array List

Arrays are harder to use than ArrayLists, but they have a speed advantage, even on simple element accesses. In Table 7.3.1, we represent a sum of two 100-element collections: an array and an ArrayList.

Integer List Loop

There are several ways to iterate elements of an integer list. In Table 7.3.1, we compare two different ways.

Char Array and StringBuilder

We can replace a StringBuilder with a char array in several programs.

Binary Search

As showed in Table 7.3.1, the BinarySearch method searches an integer in a sorted array of integers. This is more practical to use compare to a for loop.

Unoptimized	Optimized
Array copy	

Unoptimized	Optimized
`for (int j = 0; j < a.length; j++)` `b[j] = a[j];`	`System.arraycopy(a, 0, b, 0, b.length);`
Locality of reference	

`for(int i=0;i<1000000;i++){`
`int sum = 0;`
`for(int x=0;x<50000;x+=100){`
`sum += values[x];`
`}}`
	`for(int i=0;i<1000000;i++){`
	`int sum = 0;`
	`for(int x=0;x<500;x++){`
	`sum += values[x];`
	`}}`

Compare array to array list	

`for(int i=0;i<1000000;i++){`
`int sum = 0;`
`for(int v=0;v<list.size();`
`v++)`
`sum += list.get(v);`
`}`
	`for(int i=0;i<1000000;i++){`
	`int sum = 0;`
	`for(int v=0;v<array.length;`
	`v++)`
	`sum += array[v];`
	`}`

Compare integer list loop	

`for(Integer i:list) count++;`	`int size=list.size();`
	`for(int i=0;i<size;i++)`
	`count++;`

Char array StringBuilder	

`for(int i=0;i<1000000;i++){`
`StringBuilder builder=`
`new StringBuilder();`
`for(int v=0;v<1000;v++)`
`builder.append('?');`
`String result=`
`builder.toString();`
`}`
	`for(int i=0;i<1000000;i++){`
	`char[] array=new char[1000];`
	`for(int v=0;v<1000;v++)`
	`array[v] = '?';`
	`String result=`
	`new String(array);`
	`}`

Binary search	

`for(int i=0;i<10000000;i++){`
`int index = -1;`
`for(int j=0;j<values.length;`
`j++)`
`if (values[j] == 80)`
`index = j;`
`break;`
`}}}`
	`for(int i=0;i<10000000;i++)`
	`int index =`
	`Arrays.binarySearch(values,`
	`80);`

Unoptimized	Optimized

Table 7.1: Functions

7.3.2 Results

We develop two JAVA projects in order to regroup all the optimized and unoptimized methods previously defined. We obtain the following power and energy related relationships (Figure 7.10, 7.11, 7.12 and 7.13).

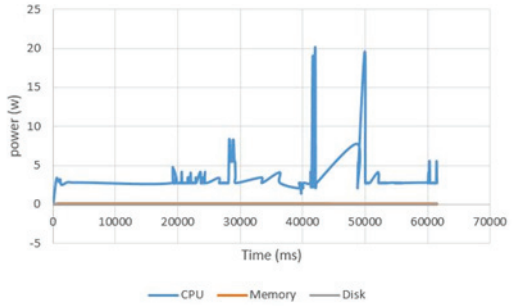

Figure 7.10: Unoptimized functions power consumption.

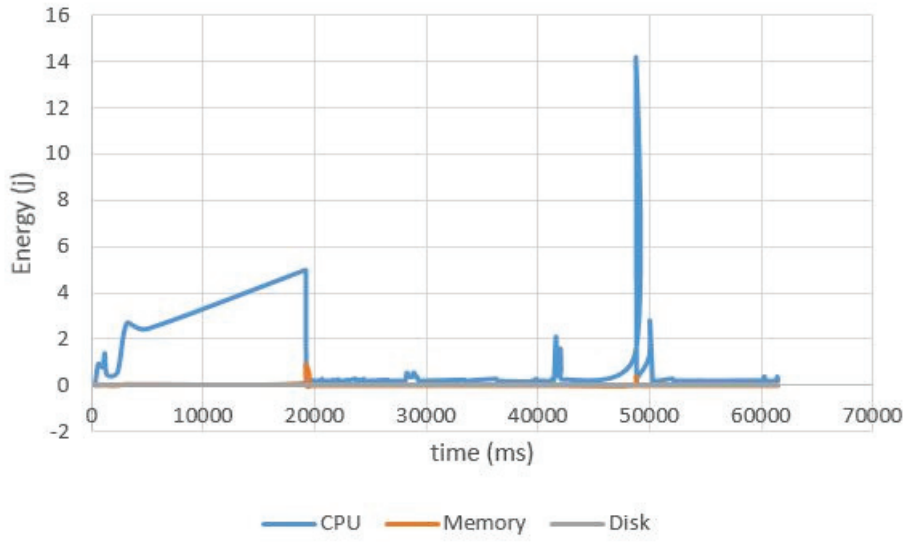

Figure 7.11: Unoptimized functions energy consumption.

Therefore, we observe that globally the power consumption of CPU dominates memory or disk consumption. If we examine the results obtained each 50 ms, we can note that the power consumption of disk can be neglected for these cases, but in several cases power consumption of memory must be taken into account. In addition, we can note that the power consumption of the unoptimized code is higher than the one of the optimized code and the total execution time of optimized code is less than the one of the unoptimized code. Consequently, it is a great interest to develop optimized parts of code in order to obtain green, sustainable and efficient software.

Hence, going more in details, for each method code, we measure the time elapsed during the execution of the tests and results are represented in Table 7.2.

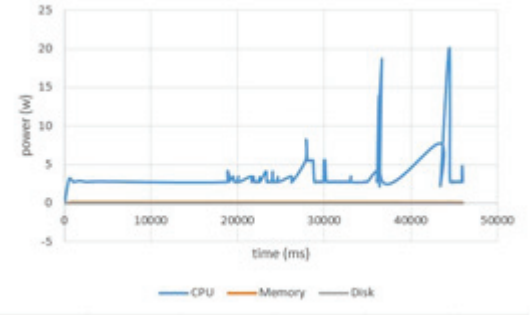

Figure 7.12: Optimized functions power consumption.

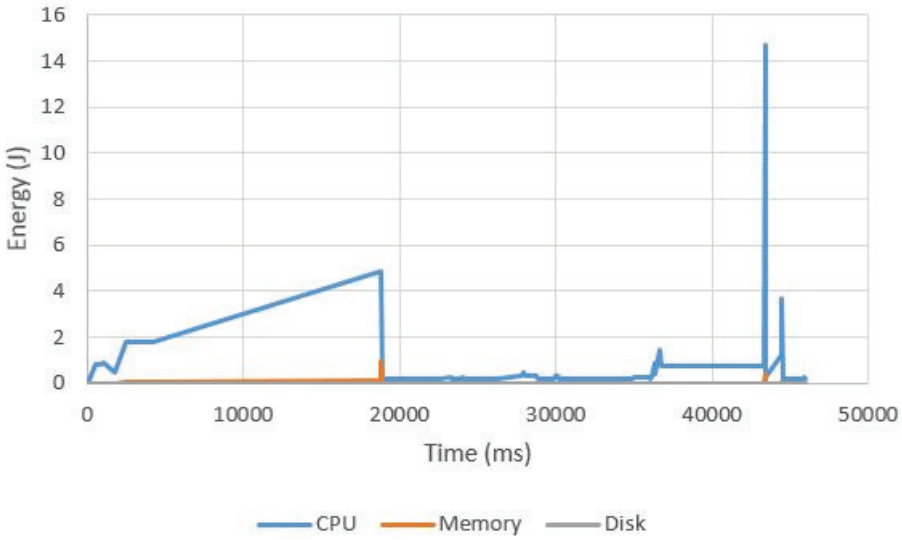

Figure 7.13: Unoptimized functions energy consumption.

Functions	**Unoptimized**	**Optimized**
	Time (ms)	
Array copy	359	312
Locality of reference	18140	17219
Compare array to array list	22047	17297
Compare integer list loop	7734	7391
Char array StringBuilder	11235	2421
Binary search	2250	438

Table 7.2: Functions time execution.

Hence, optimized codes are found faster than unoptimized codes. Particularly, we remark a faster execution of the following optimized methods: “Locality of reference”, “Compare array to array list”, “Char array StringBuilder” and “Binary search”.

7.3.3 Validation

To validate our experiments, we use a powermeter ‘wattsup ?PRO’ as shown in Figure 7.14. We connect this powermeter to the notebook via USB port. This device saves in his memory the power consumed by all process in runtime. So, we connect WattsUp to the notebook and then we wait until the power reach a stationary state. Then, we execute the unoptimized code, followed by the optimized code. We then transfer the results using the application WattsUpUSB and the

results are depicted in Figure 7.15.



Figure 7.14: wattsup?PRO.

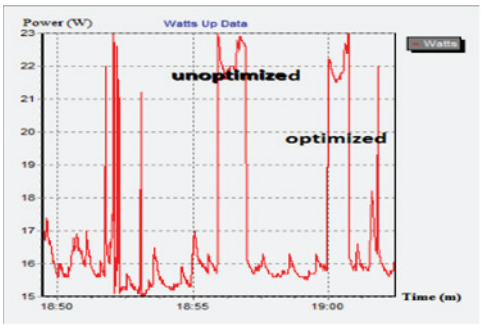

Figure 7.15: Unoptimized and optimized functions power consumption obtained with wattsup?PRO.

Comparing to the results obtain with TEEC, even if we make a measurement in each second, we can say that in all of the cases, optimized code test is faster and reveals less power than unoptimized code test. Each optimized and unoptimized curves present several increase of power as we observed with TEEC.

7.4 TEEC compared with three other tools

We perform two kinds of experiments. First, we test the optimization and the potential to save energy using a Java method which amounts to finding an integer in a list of integers contained in a file. Second, we measure the power consumption of the identical course project done by students. Their programming skills range from absolute beginners to highly experienced developers. Then, we execute their projects on the same system (an ASUS N751JK-T7238H laptop having quad-core i7-4710HQ, 2.5 GHz) and use four tools to measure the power consumption:

- **Intel Power Gadget (IPG) 3.0**: estimates the power consumption of all CPU intensive processes.
- **Joulemeter**: for a given process, estimates only the power consumption of the CPU.
- **TEEC**: our tool introduced in the previous section.

- **WattsUp-Pro**: a power meter hardware with a ±1.5% measurement error rate. We used it as a reference in order to validate measures obtained by the three other tools. Notice that we calibrate Joulemeter using WattsUp-Pro.

7.4.1 Search an Integer

We defined a file with 100 000 unique and randomly ordered integers. The goal of the program is to open this file and find a predefined value as quickly as possible. For this, we found about 40 programs using different programming techniques (iterative, recursive, linear, binary, dichotomy, etc.) on the Internet to solve this issue. Using these programs, we obtained the results that are depicted in Figure 7.16, 7.17, 7.18 and 7.19.

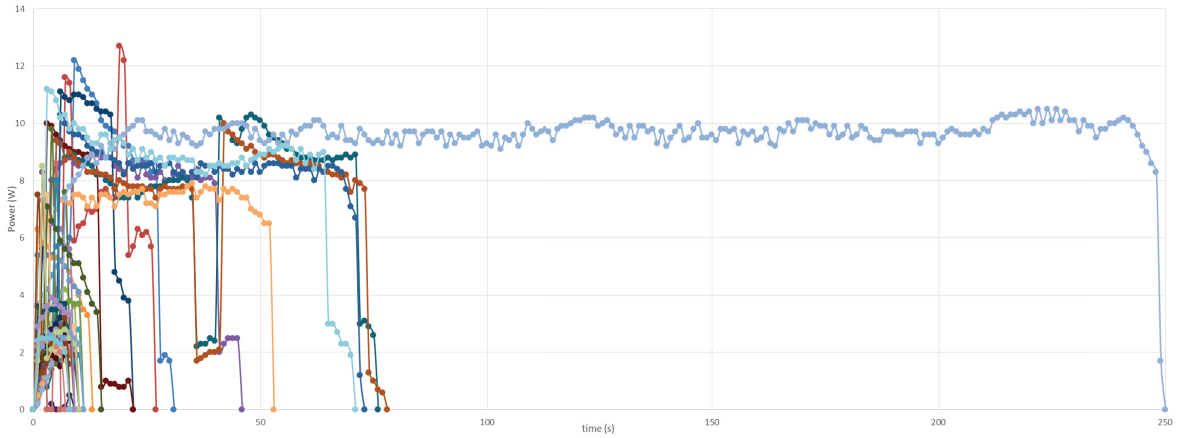

Figure 7.16: WattsUp-Pro power consumption measured and used as reference.

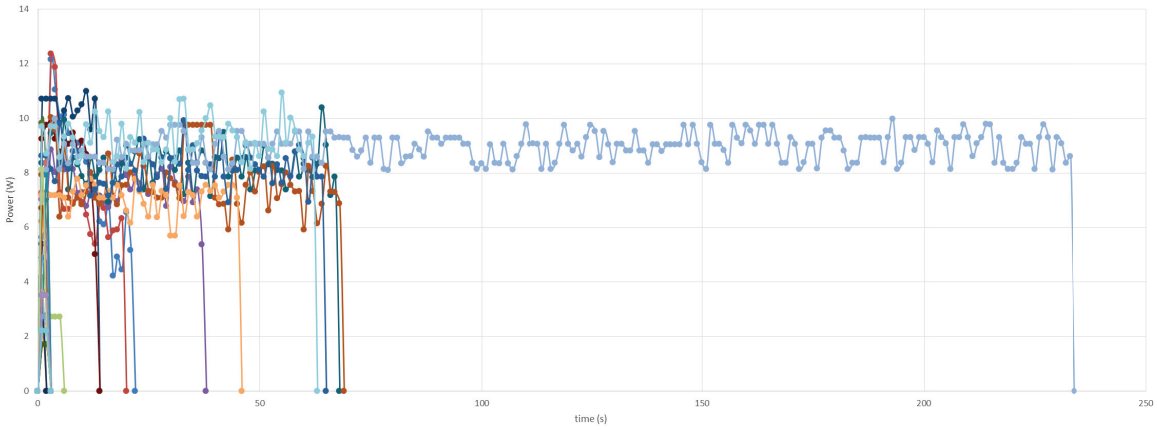

Figure 7.17: TEEC power consumption estimation.

In overall, the appearance of the curves, obtained with the four tools, seems similar. In each case, the slowest function takes about 250 seconds, while the fastest takes less than a second. However, when we take a closer look, we notice several important differences.

First, curves obtained with Joulemeter show a large deviation of power amplitudes. In fact, compared to the powermeter, we see a general deviation of about 7W. These results are not realistic although we used the WattsUp-Pro power meter to calibrate Joulemeter. In addition, the amplitude of the fastest methods exceeds 8W, whereas in the case of IPG (which also estimates CPU power consumption only) it remains below 6W. Secondly, even if it is not as

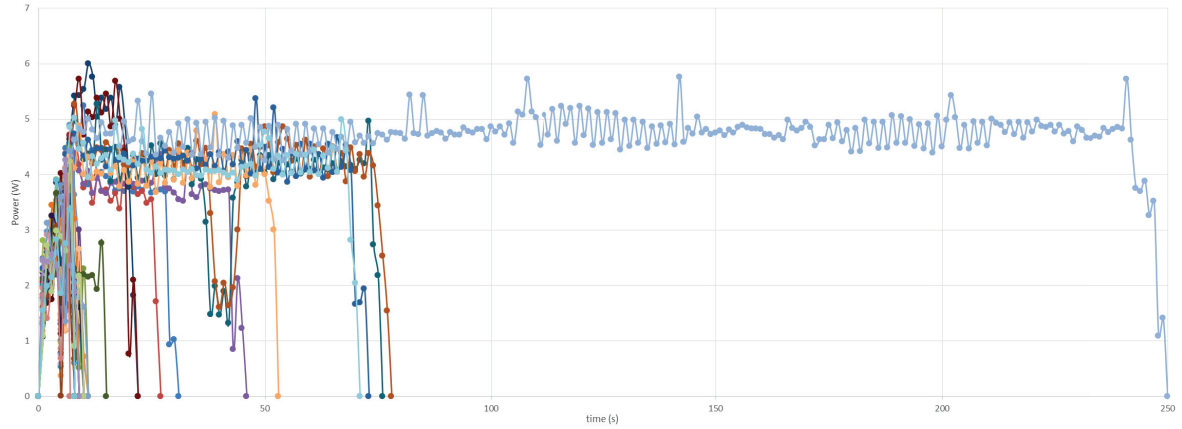

Figure 7.18: IPG power consumption estimation.

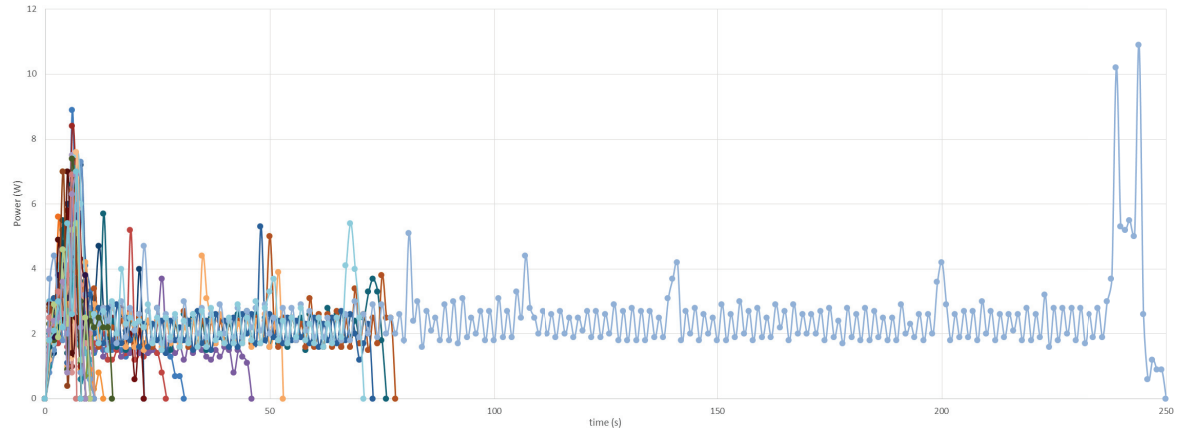

Figure 7.19: Joulemeter power consumption estimation.

remarkable as Joulemeter, the results obtained with IPG still represents a significant deviation of 4.5W compared to the power meter values. Moreover, the peaks obtained (about 6W) are half that of the powermeter (about 12W). Unlike the other tools, TEEC presents results very close to the power meter values. Indeed, we generally observe a deviation of about 1W maximum.

In order to compare the estimated (or measured) energy consumption of all methods by the four tools, we present in Table 7.3 the methods which consume the minimum and maximum energy, the average energy consumption of all methods and the percentage error relative to the powermeter of these average consumptions. To calculate the energy, we multiply the power by time.

Tools	**Minimum energy consumption (J)**	**Maximum energy consumption (J)**	**Average energy consumption (J)**	**Error rate (%)**
IPG	12,55	1161,92	89,76	45.63
Joulemeter	21,4	623,6	62,94	61,88
TEEC	2,32	2087,47	137,51	16,71
WattsUp-PRO	6,6	2343,5	165,09	-

Table 7.3: Energy consumption comparison

We find that Joulemeter estimates the biggest values of the method consuming the minimum energy, the smallest value of the method consuming the maximum energy and the average of all methods. Hence, Joulemeter is the tool with the highest error rate. Furthermore, in all cases, TEEC is the tool with the results closest to the power meter. Therefore, TEEC has the lowest error rate.

We can explain this by the fact that our tool takes into account not only the CPU, but also Memory, Disk and Network power consumption, whereas Intel Power Gadget or Joulemeter estimate only CPU power consumption. In fact, with our tool we notice that the power consumption due to Memory can reach up to 2.5 W which can not be neglected. In addition, TEEC also presents a small difference of power consumption which are explainable by the fact that other components that we do not take into consideration also consume power (e.g. chipset, PSU inefficiency, etc.). Thus, it is important to take into account all components in consideration during power estimation. Otherwise, TEEC becomes more accurate compared to the others in the execution time of each method. This is because it begins to estimate only when the method is executed whereas the others give a global view. Evaluating these 41 methods that carry out the same goal, we also optimize the source code of an application to reach a sustainable and green method and then application software. In fact, this simple experiment shows that it was possible to save about 240s execution time and 2336.9J. Also, we observed that several methods consume more power but are executed more quickly.

7.4.2 Students mini-project

Contrary to previous experiments, in this section, we want to show the impact on the energy consumption of software to respect the Green software engineering process that we represented in Figure 4.1 and described in the section IV. For this, we gave the same mini-project to 34 students which worked in pairs. The goal is to develop a program which allows generating an OWL (Web Ontology Language) file from data recovered from a database. We gave the tables representing the characteristics of a product that is a pen. The students have to manage the connection at the Oracle database, get the information and put them in the owlapi methods to generate an owl file. Firstly, students tried to understand the requirements, then they went to the design and implementation stages. They realized several tests which allowed them to validate the requirements. They continued to the usage step by opening the owl file in the software "protégé" and worked on it. Some students succeed in going to the maintenance step, optimizing their code and bringing several evolutions. Moreover, several students have reflected at the disposal step by identifying the parts of source code that could be reusable. For example, if the data are presented in an express file instead of a database, so they need to develop a parser to recover the data and reuse the way to generate the owl file with the owlapi methods.

Then, we execute the mini-projects and show the power consumption values obtained with each power measurement tool in Figure 7.20, 7.21, 7.22 and 7.23. In addition, thanks to our TEEC tool, we modify dynamically the source code to measure the execution time of each method in each project and count the number of method called. Consequently, we can easily locate the part of code that we can optimize in order to obtain sustainable and green software.

As in the previous test, globally, the shape of the curves obtained by the four tools seems identical. For all tests, the fastest method lasts about 80 seconds and the shortest last about less than a second. However, we also observe significant differences.

Again, the curves obtained with Joulemeter presents a large deviation from the power meter. We only observe a few peaks of about 7.5W, whereas for the power meter we observe several peaks that can reach 18W.

We find that the results obtained with IPG are less satisfactory than in the previous test. Indeed,

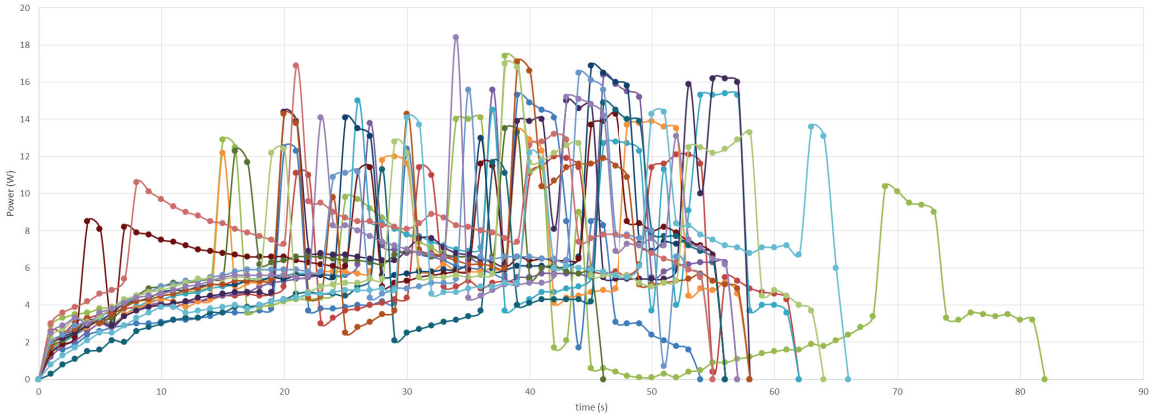

Figure 7.20: WattsUp-Pro power consumption measured and used as reference.

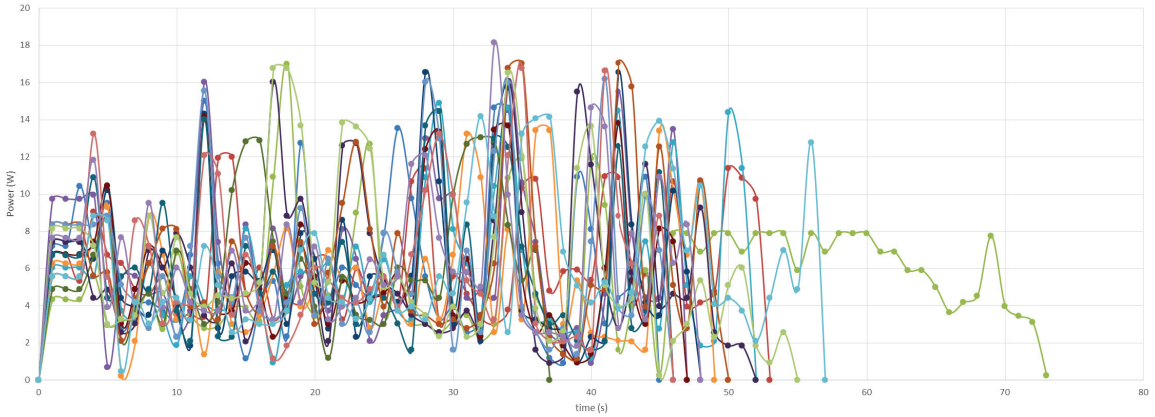

Figure 7.21: TEEC power consumption estimation.

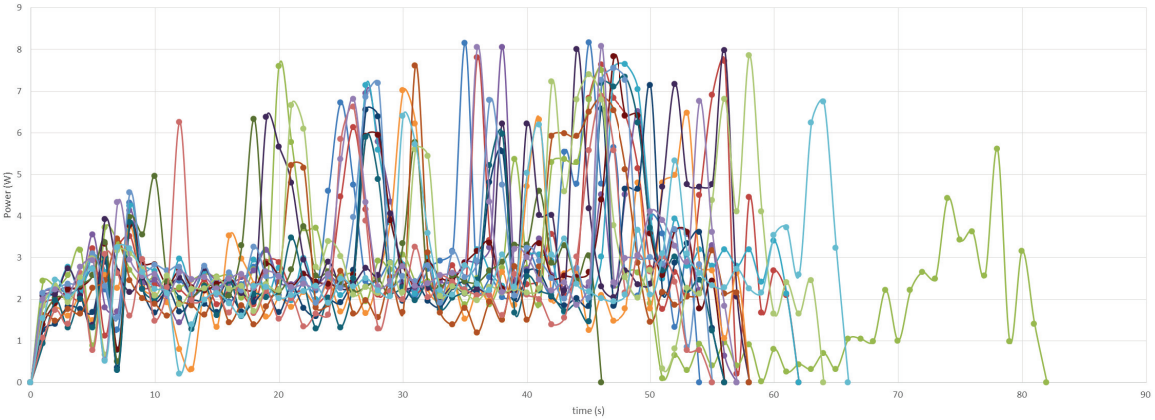

Figure 7.22: IPG power consumption estimation.

in general, we see a difference of 7W compared to the power meter. However, we observe many more peaks than Joulemeter which remain in the vicinity of 8W.

In TEEC case, we observe more peaks which are more close to powermeter than other two tools.

Moreover, we observe that the projects respecting sustainable and green software engineering process obtain better results.

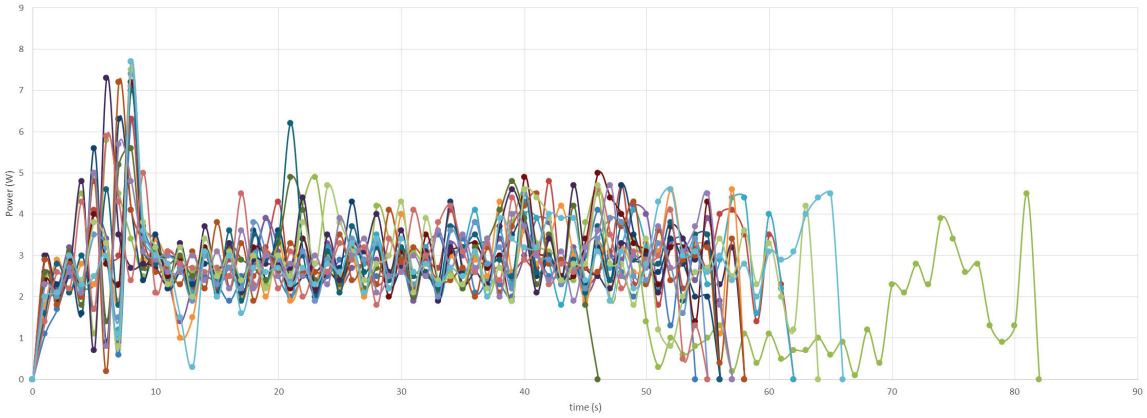

Figure 7.23: Joulemeter power consumption estimation.

We observe also that several methods consume more power but are executed more quickly. In Figure 7.24, we represented, for each project, the number of method that the project executes, the total, average, minimum and maximum time execution of each method of each project. The total energy consumption for each project classified by the tool used. Then, the minimum, maximum and average energy consumption among the projects and the percentage error relative to the powermeter of these average consumptions depending on four tools. To calculate the energy, we multiply the power by time.

Project name	Number of method called	Total time execution (s)	Average time execution (s)	Maximum time execution (s)	Minimum time execution (s)	Total energy consumption (J) IPG	Joulemeter	TEEC	WattsUp-PRO
Stylo1-1	57	44.60	0.782	43.31	3,95E-04	161,92	141,7	270,59	276,9
Stylo10-1	3	50.32	16.77	46.78	0,07	190,33	178,9	329,34	365,7
Stylo11-1	56	73.21	1.307	39.49	1,38E-04	201,36	195,3	400,62	412,4
Stylo12-1	1	39.98	39.98	39.98	39,98	166,31	163,4	302,35	349,1
Stylo13-1	27	60.15	2.227	50.42	2,23E-04	182,88	179,8	307,56	407,6
Stylo14-1	108	51.76	0.479	39.08	1,98E-04	147,41	162,9	275,40	367,1
Stylo15-1	1	46.09	46.09	46.09	46,09	144,33	160	289,87	372,3
Stylo16-1	1	44.99	44.99	44.99	44,99	165,04	164,8	276,49	394,8
Stylo17-1	1	26.14	26.14	26.14	26,14	127,58	137	227,99	276,7
Stylo2-1	46	51.30	1.115	6.008	2,86E-04	183,96	170,7	306,82	421,2
Stylo3-1	4	45.21	11.30	15.33	0,04	147,99	165	270,72	276,8
Stylo4-1	8	47.94	5.993	6.007	1,68E-03	155,67	168,1	316,63	378,1
Stylo5-1	13	84.03	6.464	44.33	2,82E-04	169,56	157,2	289,96	347,2
Stylo6-1	167	44.82	0.268	38.90	2,31E-04	136,11	161,1	281,65	439,1
Stylo7-1	99	7.743	0.078	6.050	3,15E-04	194,69	183,5	340,55	444
Stylo8-1	33	44.70	1.354	37.10	3,73E-03	170,93	167,4	305,10	383,8
Stylo9-1	2	56.02	28.01	46.83	9,19	179,68	188,5	345,84	386,8
					Minimum (J)	127,58	137	227,99	276,7
					Maximum (J)	201,36	195,3	400,62	444
					Minimum Error rate (%)	41,52	40,39	2,19	-
					Maximum Error rate (%)	69,01	63,31	35,86	-

Figure 7.24: Performance and energy results of each project.

We show again that TEEC is closest to WattsUP PRO measured values than others. Evaluating these projects that realized the same goal, we demonstrate also the great interest to follow a sustainable and green software engineering process by respecting each step. Thanks to the fact that we can manipulate the source code, we locate the parts of code that developers could optimize in performance, power and energy in order to obtain sustainable and green software.

In fact, it is possible to save about 40s, 167.3J and reduce the number of methods used from 167 to 8.

7.5 Conclusion

Different types of experiments have been performed to demonstrate the accuracy, efficiency and simplicity of our tool TEEC which allow to estimate the software power consumption taking into account several components. Firstly, we examined the impact of CPU alone on the software energy consumption. Then, we add in TEEC the capacity to estimate respectively, memory, hard disk and network power consumption in order to demonstrate their importance in the software power consumption and show that to assume that the CPU power consumption is enough to estimate software power consumption is not sufficient and accurate because other components also have an important impact on the software power consumption which is not neglected.

Moreover, the fact that TEEC allows to locate part of source code consuming the most energy helps and guides easily and quickly developers to improve the quality of their source code in order to build efficient, sustainable and green software which reduce the greenhouse gas emissions while keeping the same functionalities.

In addition, students mini project showed us the importance of sustainable and green software engineering process like we described in Chapter 4 in order to obtain sustainable and green software products. Students having respected all the criteria defined in each step of the BUA methodology proposed in Chapter 4 have obtained applications consuming less energy than others. This is motivating to conduct these experiments in more complex industrial projects in order to demonstrate the efficiency of this sustainable and green software engineering process.

Chapter 8

Conclusion

"It's more fun to arrive a conclusion than to justify it."

Malcolm Forbes

8.1 Lessons learned

Building sustainable and green software product by reducing the greenhouse gas emissions, we needed to understand the meanings of 'green' and 'sustainable' in software domain. Hence, we proposed our own definitions for the terms "Sustainable Software", "Green Software", "Green with Software" and "Green within Software"after analyzing related literature and related ICT sector publications.

During our systematic literature review process, we then focused on the works on the software power consumption. We classified them in three groups: hardware, software and hybrid methodologies. In fact, since researchers want accurate power consumption results, they used hardware methodologies using powermeter or printed circuits connected directly to a component. Then, they moved on the hybrid methodologies because of their simplicities. However, in both these cases, it is not possible to connect a hardware to a virtual machine or to measure the power consumption of a particular process. Hence, they oriented their research to power consumption estimation methodologies based on software, which is also considered in this thesis. We first generated a systematic literature on the estimation models of the power consumption of software. We remarked that, in the majority of works, only the power consumption of one component (mostly CPU) is considered, while the others are neglected. Moreover, we noticed a lack of a dynamic manipulation of the source code before the execution of the application in order to guide developers to locate their hotpoints (source code area that is heavily called during execution).

In order to reduce the negative impacts on the environment, it is needed to respect to a software development methodology to obtain sustainable and green software. For this, we proposed BUA (Before Usage After) methodology, which respect sustainable and green criteria after each step of software engineering process. Accordingly, a green analysis step is added after each step of the process (requirements, design and implementation, tests, usage, maintenance and disposal) in order to define and describe sustainable and green criteria. Thus, the methodology necessitates to pass to the next step only when all the criteria of the previous step are validated. If not, a return back at a previous step is performed.

In this research, we proposed GMTEEC (Generic Methodology of a Tool to Estimate the Energy Consumption), which is composed of four layers in order to guide researches to build tool allowing to estimate software power consumption. In fact, researchers proposed tools allowing to

estimate software power consumption based on mathematical formulae defined on the features of each component. However, depending on their research area, they performed interesting choices in their areas, but not interesting in other area. Usually, tools are operating only on Linux or Windows. In several cases, they considered only one component (power consumption) and neglected the impacts of others. Several other criteria which limited the generic of the tool. Hence, the description of the four layers will help and guide researches in their choice to build generic tool to estimate software power consumption.

We applied GMTEEC to build TEEC (Tool to Estimate Energy Consumption). We followed the four layers of GMTEEC to have a great impact in reducing greenhouse gas emissions. For this, we chose Java as programming language because it is the most popular and used language. Sigar and Javassist associated at this language to get dynamically data about different components. We established power consumption mathematical formulae for several components like CPU, memory, hard disk and network in order to estimate accurately and efficiently software power consumption. We also added the capacity to locate part of source code consuming the most energy to help and guide developers to improve their source code to obtain efficient, sustainable and green software.

We experimented the accuracy and efficiency of our proposed tool TEEC in several scenarios. First, we performed tests on Fibonacci sequence only on the CPU power consumption and compared them with Joulemeter tool developed by Microsoft. When we validated the results, we added respectively memory, hard disk and network power consumption formulae in TEEC to show their impacts on software power consumption. We realized tests on several functions and compared the results with a powermeter to demonstrate the importance of considering also these components in the estimation of software power consumption.

Then, when we validated and demonstrated the accuracy, efficiency and the importance of all components in the software power consumption estimation, we carried out two tests to show the capacity of TEEC to locate hotpoints. First, to test the behavior of a particular method which consists of finding an integer among integers in a file. Second, to test a course project which involves developing an application connected to a database, getting information of a product to generate an owl file and taking into account the sustainable and green engineering process. In these two tests, we compared obtained results with the results obtained by different power measurement tools, such as Intel Power Gadget (IPG), Joulemeter and a power meter in order to validate the accuracy and simplicity of TEEC. We noticed that the results closest to the power meter values were provided by TEEC, because it not only considers CPU but also memory, disk and network components power consumptions for a given process. Moreover, TEEC begins to measure the power consumption when we launched the project, whereas for the other tools, we do not know exactly when the power estimation begins. This explains the accuracy of our proposed tool. TEEC manipulates also source code of the application, we know the number of used called methods and their execution time. Thus, developers may identify the fundamental parts of the source code that need to be optimized to build efficient, sustainable and green software.
In the experiments, we observed different software power consumption results having the same functionalities. Hence, an optimized program allows to reduce execution time but also save energy. Respecting all the sustainable and green criteria described in our proposed sustainable and green software engineering process, it is possible to obtain efficient, sustainable and green software, and consequently, reduce the greenhouse gas emissions.

8.2 Future directions

Our proposed tool TEEC allows to estimate the power consumption during software execution due to four components: CPU, memory, hard disk and network. The results of the experiments showed that TEEC gives close results to powermeter in the power consumption estimation. Still,

we remarked a difference, which can be explained by the power consumption due to other components. Hence, we will extend TEEC by integrating power models of new components, e.g. chipset, PSU, etc.

TEEC allows to locate the parts of source code that consume the most energy during the execution. Hence, it guides developers to improve the quality of their source code. Doing this, they need to be experts or to find experts to improve these parts in terms of code efficiency. To simplify the tasks of developers, based on a list of best practices techniques, we will automatize TEEC.

As it stands, TEEC allows to estimate power consumption of a given process. We will extend this capacity to estimate several processes' power consumptions at the same time. Then, based on machine learning techniques [141], we will manage efficiently applications energy consumption.

In addition, TEEC has been tested on a set of chosen projects. Extending these experiments will show the impacts of TEEC during the development of an industrial software project. Besides, we will adapt TEEC in other domains, such as cloud computing, web services, etc.

8.3 Publications

The results obtained in this thesis have been published in international journals and conferences with reading committee. The list of our papers are:

- **Hayri Acar**, Gülfem I. Alptekin, Jean-Patrick Gelas and Parisa Ghodous. *The impact of source code in software on power consumption.* The International Journal of Electronic Business Management (IJEBM). Volume 14, Issue 5, pages 42-52, September 2016.

- **Hayri Acar**, Gülfem I. Alptekin, Jean-Patrick Gelas and Parisa Ghodous. *TEEC: Improving power consumption estimation of software.* In the Proceedings of the 30th International Conference on Environmental informatics (EnviroInfo 2016), Berlin, Germany, September 2016.

- **Hayri Acar**, Gülfem I. Alptekin, Jean-Patrick Gelas and Parisa Ghodous. *Beyond CPU: Considering Memory Power Consumption of Software.* In the Proceedings of the 5th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2016). Rome, Italy, April 2016.

- **Hayri Acar**, Gülfem I. Alptekin, Jean-Patrick Gelas and Parisa Ghodous. *A Green approach to save energy consumed by software.* In the Proceedings of the 3rd International Conference on ICT for Sustainability (ICT4S 2015). Copenhagen, Denmark, September 2015.

- **Hayri Acar**, Gülfem I. Alptekin, Jean-Patrick Gelas and Parisa Ghodous. *Towards a Green and Sustainable Software.* In the Proceedings of the 22nd International Conference on Concurrent Engineering (ISPE 2015). Delft, Netherlands, July 2015.

Moreover, we gave the following talks and presentations:

- **Hayri Acar** and Jean-Patrick Gelas. *TEEC: Logiciel vert et durable.* Conférence EcoInfo, Impact des logiciels sur l'environnement, quid de l'éco-conception ?. Grenoble, France, February 2017.

- **Hayri Acar** and Jean-Patrick Gelas. *Méthodologie de Développement de Logiciel vert.* Avalon team. ENS Lyon, Lyon, France, June 2016.

- **Hayri Acar** and Jean-Patrick Gelas. *Une approche verte pour économiser de l'énergie consommée par des logiciels.* Thesis days. Lyon, France, November 2016.

Bibliography

[1] Ericsson. Energy and Carbon Report. Technical report, 2013.

[2] NetMarketShare. Desktop Operating System Market Share, 2017.

[3] NetMarketShare. Mobile/Tablet Operating System Market Share, 2017.

[4] Paul Dempsey. Rambus CEO calls for collaboration and an architectural focus for memory, 2013.

[5] Matt. Average Power Use Per Server, 2015.

[6] Steven Sinofsky. Windows 7 Energy Efficiency, 2009.

[7] UNFCCC. ADOPTION OF THE PARIS AGREEMENT: Proposal by the President to the United Nations Framework Convention on Climate Change. volume 21932, pages 1–32. Paris, 2015.

[8] Gartner. Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emissions, 2007.

[9] SMARTer2020. GeSI SMARTer2020: The Role of ICT in Driving a Sustainable Future. 2012.

[10] Coral Calero and Mario Piattini. *Introduction to Green in Software Engineering*, pages 3–27. Springer International Publishing, 2015.

[11] Cambridge dictionary. Sustainable definition.

[12] Oxford dictionaries. Definition of Sustainable.

[13] Birgit Penzenstadler, Ankita Raturi, Debra Richardson, Coral Calero, Henning Femmer, and Xavier Franch. Systematic mapping study on software engineering for sustainability (SE4S). *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, (November):1–14, 2014.

[14] United Nations Commission. Report of the World Commission on Environment and Development: Our Common Future. Technical report, 1987.

[15] B Penzenstadler and H Femmer. A generic model for sustainability with process- and product-specific instances. *GIBSE 2013 - Proceedings of the 2013 Workshop on Green in Software Engineering, Green by Software Engineering*, (June 2015):3–7, 2013.

[16] ICT4S. The international conferences ICT4S – ICT for Sustainability.

[17] Lorenz M. Hilty, Bernard Aebischer, Göran Andersson, and Wolfgang Lohmann. *Proceedings of the First International Conference on Information and Communication Technologies for Sustainability.* 2013.

[18] Sm Easterbrook. Climate change: a grand software challenge. *... of the FSE/SDP workshop on Future of software ...*, pages 99–103, 2010.

[19] B. Donnellan, C. Sheridan, and E. Curry. A Capability Maturity Framework for Sustainable Information and Communication Technology. In *IEEE Computer Society*, 2011.

[20] Helen Hasan, Alemayehu Molla, and Vanessa Cooper. Towards a green IS taxonomy. In *SIGGreen Workshop*, pages 1–22, Barcelona, Spain, 2012.

[21] ERICSSON. Ericsson energy and carbon report. Technical Report November, 2014.

[22] C Calero, M F Bertoa, and M Á Moraga. A systematic literature review for software sustainability measures. In *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*, pages 46–53, 2013.

[23] M. Dick, J. Drangmeister, E. Kern, and S. Naumann. Green software engineering with agile methods. In *2013 2nd International Workshop on Green and Sustainable Software (GREENS)*, pages 78–85, San Francisco, CA, USA, May 2013.

[24] Birgit Penzenstadler, Ankita Raturi, Debra Richardson, Coral Calero, Henning Femmer, and Xavier Franch. Systematic Mapping Study on Software Engineering for Sustainability (SE4S). In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 14:1—-14:14, New York, NY, USA, 2014. ACM.

[25] T. Johann, M. Dick, E. Kern, and S. Naumann. Sustainable development, sustainable software, and sustainable software engineering: An integrated approach. In *2011 International Symposium on Humanities, Science and Engineering Research*, pages 34–39, June 2011.

[26] Christian Manteuffel and Spyros Ioakeimidis. A systematic mapping study on sustainable software engineering: a research preview. In *9th {Student} colloquium*, pages 35–40, 2012.

[27] Nadine Amsel, Zaid Ibrahim, Amir Malik, and Bill Tomlinson. Toward Sustainable Software Engineering (NIER Track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 976–979, New York, NY, USA, 2011. ACM.

[28] Martin Mahaux, Patrick Heymans, and Germain Saval. Discovering Sustainability Requirements: An Experience Report. In *Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, REFSQ'11, pages 19–33, Berlin, Heidelberg, 2011. Springer-Verlag.

[29] Merriam-Webster.

[30] Richard T Watson, Marie-Claude Boudreau, and Adela J Chen. Information Systems and Environmentally Sustainable Development: Energy Informatics and New Directions for the is Community. *MIS Q.*, 34:23–38, 2010.

[31] S. Murugesan and P. A. Laplante. It for a greener planet [guest editors' introduction]. *IT Professional*, 13(1):16–18, Jan 2011.

[32] Adrian T. Sobotta, Irene N. Sobotta, and John Gøtze. *Greening IT: How Greener IT Can Form a Solid Foundation For a Low-Carbon Society.* 2010.

[33] Ranjit Bose and Xin (Robert) Luo. Green IT adoption: a process management approach. *International Journal of Accounting & Information Management*, 20(1):63–77, 2012.

[34] Alemayehu Molla, Vanessa A. Cooper, and Siddhi Pittayachawan. IT and Eco-sustainability : Developing and Validating a Green IT Readiness Model. *ICIS 2009 Proceedings*, 1(Paper 141):1–17, 2009.

[35] Bob Steigerwald and Abhishek Agrawal. Developing Green Software. *Intel White Paper*, pages 1–11, 2011.

[36] Juha Taina. Good, Bad, and Beautiful Software – In Search of Green Software Quality Factors. *CEPIS UPGRADE*, 2011(4):22–27, 2011.

[37] K. Erdélyi. Special factors of development of green software supporting eco sustainability. In *2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 337–340, Sept 2013.

[38] Nathalie Bachour. Green IT Project Management. *Sustainable ICTs and Management Systems for Green Computing*, pages 146–178, 2012.

[39] S. Murugesan. Making it green. *IT Professional*, 12(2):4–5, March 2010.

[40] K. Mohan, B. Ramesh, L. Cao, and S. Sarkar. Managing disruptive and sustaining innovations in green it. *IT Professional*, 14(6):22–29, Nov 2012.

[41] Birgit Penzenstadler. What does Sustainability mean in and for Software Engineering? *Proceedings of the 1st International Conference on ICT for Sustainability (ICT4S)*, (January 2013), 2013.

[42] N. Bachour and L. Chasteen. Optimizing the value of green it projects within organizations. In *2010 IEEE Green Technologies Conference*, pages 1–10, April 2010.

[43] Power Supply Calculator. No Title, 2015.

[44] OuterVision Power Supply Calculator, 2017.

[45] Wu Chun Feng and Heshan Lin. The green500 list: Year two. *Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010*, (May 2017), 2010.

[46] Power efficiency in high performance computing. *Ipdps*, pages 1–8, 2008.

[47] Analyzing the Energy Efficiency of a Database Server. *the 2010 International Conference*, page 231, 2010.

[48] Zichen Xu, Yi-Cheng Tu, and Xiaorui Wang. PET: Reducing database energy cost via query optimization. *Proceedings of the VLDB Endowment*, 5(12):1954–1957, 2012.

[49] Ravi A. Giri and Anand Vanch. Increasing Data Center Efficiency with Server Power Measurements. (January):8, 2010.

[50] Rafael Vidal Aroca and Luiz Marcos Garcia Gonçalves. Towards green data centers: A comparison of x86 and ARM architectures power efficiency. *Journal of Parallel and Distributed Computing*, 72(12):1770–1780, 2012.

[51] IBM. IBM Systems: Power Executive 1.02 Release Notes. 2006.

[52] Intel Corporation and N E C Corporation. Intelligent Platform Management Interface Specification v2.0 rev. 1.1 E6 Markup. 2014.

[53] D McIntire, T Stathopoulos, and W Kaiser. etop-Sensor Network Application Energy Profiling on the LEAP2 Platform. In *2007 6th International Symposium on Information Processing in Sensor Networks*, pages 576–577, 2007.

[54] SCHNEIDER ELECTRIC. Metered Rack PDU, 2017.

[55] SynapSense Power Suite^{TM}. Power Monitoring, 2017.

[56] Texas Instruments. High-or Low-Side Measurement , Bi-Directional CURRENT / POWER MONITOR with I 2 C ^{TM} Interface, 2013.

[57] Barbara Kitchenham and Pearl Brereton. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, 55(12):2049–2075, 2013.

[58] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic Mapping Studies in Software Engineering. *12Th International Conference on Evaluation and Assessment in Software Engineering*, pages 68–77, 2008.

[59] Michael D. Powell, Arijit Biswas, Joel S. Emer, Shubhendu S. Mukherjee, Basit R. Sheikh, and Shrirang Yardi. CAMP: A technique to estimate per-structure power at run-time using a few simple parameters. *Proceedings - International Symposium on High-Performance Computer Architecture*, (Hpca):289–300, 2009.

[60] P. K. Gupta and G. Singh. A framework of creating intelligent power profiles in operating systems to minimize power consumption and greenhouse effect caused by computer systems. *Journal of Green Engineering*, 1(2):145–163, 2011.

[61] Ramon Bertran, Marc Gonzalez, Xavier Martorell, Nacho Navarro, and Eduard Ayguade. Decomposable and Responsive Power Models for Multicore Processors using Performance Counters Categories and Subject Descriptors. *Proceedings of the 24th ACM International Conference on Supercomputing*, pages 147–158, 2010.

[62] Jason Mars, Lingjia Tang, and Mary Lou Soffa. Directly characterizing cross core interference through contention synthesis. *Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers - HiPEAC '11*, page 167, 2011.

[63] Georg Hager, Jan Treibig, Johannes Habich, and Gerhard Wellein. Exploring performance and power properties of modern multi-core chips via simple machine models. *Concurrency Computation*, 28(2):189–210, 2016.

[64] Sriram Sankaran. Predictive Modeling Based Power Estimation for Embedded Multicore Systems. CF '16, pages 370–375, New York, NY, USA, 2016. ACM.

[65] Aras Atalar, Anders Gidenstam, Paul Renaud-Goud, and Philippas Tsigas. Modeling Energy Consumption of Lock-Free Queue Implementations. *Proceedings - 2015 IEEE 29th International Parallel and Distributed Processing Symposium, IPDPS 2015*, pages 229–238, 2015.

[66] Guoming Tang, Weixiang Jiang, Zhifeng Xu, Fangming Liu, and Kui Wu. Zero-Cost, Fine-Grained Power Monitoring of Datacenters Using Non-Intrusive Power Disaggregation. *Proceedings of the 16th Annual Middleware Conference on - Middleware '15*, pages 271–282, 2015.

[67] Yawen Chen, Jason Mair, Zhiyi Huang, David Eyers, and Haibo Zhang. A State-based Energy / Performance Model for Parallel Applications on Multicore Computers Analytical Model : State-based model. (1), 2015.

[68] Bogdan Marius Tudor and Yong Meng Teo. On understanding the energy consumption of ARM-based multicore servers. *International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 267–278, 2013.

[69] Weisong Shi, Shinan Wang, and Bing Luo. CPT: An Energy-Efficiency Model for Multicore Computer Systems. *Cs.Wayne.Edu*, (201001):1 – 6, 2012.

[70] Hui Chen, Bing Luo, and Weisong Shi. Anole: A Case for Energy-Aware Mobile Application Design. In *2012 41st International Conference on Parallel Processing Workshops*, pages 232–238, 2012.

[71] S. Wang, Youhuizi Li, W. Shi, Lingjun Fan, and A. Agrawal. Safari: Function-level power analysis using automatic instrumentation. In *2012 International Conference on Energy Aware Computing*, pages 1–6, 2012.

[72] Hitoshi Oi. Power-performance analysis of JVM implementations. *2011 International Conference on Information Technology and Multimedia: "Ubiquitous ICT for Sustainable and Green Living", ICIM 2011*, (November), 2011.

[73] Nitin Agrawal, Leo Arulraj, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Emulating goliath storage systems with David. *Tos*, 7(4):1–21, 2012.

[74] Shinan Wang, Hui Chen, and Weisong Shi. SPAN: A software power analyzer for multicore computer systems. *Sustainable Computing: Informatics and Systems*, 1(1):23–34, 2011.

[75] a Noureddine, A Bourdon, R Rouvoy, and L Seinturier. A preliminary study of the impact of software engineering on {GreenIT}. *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, pages 21–27, 2012.

[76] Thierry LEBOUCQ. Green Digital Charter. 2016.

[77] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka Aloke Bhattacharya. Virtual Machine Power Metering and Provisioning. *1st ACM Symposium on Cloud Computing (SoCC '10)*, pages 39–50, 2010.

[78] Intel Power Gadget, 2016.

[79] Thomas Vogelsang. Understanding the Energy Consumption of Dynamic Random Access Memories. *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2010.

[80] Nagendra Gulur. A Comprehensive Analytical Performance Model of DRAM Caches. pages 157–168, 2015.

[81] Chi-Kang Chen, Hsin-I Wu, Chi-Ting Hsiao, and Ren-Song Tsay. An Accurate and Flexible Early Memory System Power Evaluation Approach Using a Microcomponent Method. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES '16, pages 3:1—-3:8, New York, NY, USA, 2016. ACM.

[82] Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Samira Khan, and Onur Mutlu. Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost. *ACM Transactions on Architecture and Code Optimization*, 12(4):1–29, 2016.

[83] Mingyu Gao, Christina Delimitrou, Dimin Niu, Krishna T. Malladi, Hongzhong Zheng, Bob Brennan, and Christos Kozyrakis. DRAF: A Low-Power DRAM-Based Reconfigurable Acceleration Fabric. *Proceedings - 2016 43rd International Symposium on Computer Architecture, ISCA 2016*, pages 506–518, 2016.

[84] Qixiao Liu, Miquel Moreto, Jaume Abella, Francisco J Cazorla, and Mateo Valero. DReAM : an Approach to Estimate Per-Task DRAM Energy in Multicore Systems Memory Power (Watts). V, 2011.

[85] Venkata Kalyan Tavva, Ravi Kasha, and Madhu Mutyam. EFGR: An Enhanced Fine Granularity Refresh Feature for High-Performance DDR4 DRAM Devices. *ACM Trans. Archit. Code Optim.*, 11(3):31:1—-31:26, 2014.

[86] Wongyu Shin, Jeongmin Yang, Jungwhan Choi, and Lee Sup Kim. NUAT: A non-uniform access time memory controller. *Proceedings - International Symposium on High-Performance Computer Architecture*, pages 464–475, 2014.

[87] Mahdi Nazm Bojnordi and Engin Ipek. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. *Proceedings - International Symposium on High-Performance Computer Architecture*, 2016-April:1–13, 2016.

[88] A Valero, J Sahuquillo, S Petit, P López, and J Duato. Design of Hybrid Second-Level Caches. *IEEE Transactions on Computers*, 64(7):1884–1897, 2015.

[89] Shyamkumar Thoziyoor, Jung Ho Ahn, Matteo Monchiero, Jay B. Brockman, and Norman P. Jouppi. A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. *Proceedings - International Symposium on Computer Architecture*, (June 2008):51–62, 2008.

[90] Micron Technology. TN-41-01: Calculating Memory System Power for DDR3. 2007.

[91] Benjamin S. Parsons and Vijay S. Pai. A mathematical hard disk timing model for full system simulation. *ISPASS 2013 - IEEE International Symposium on Performance Analysis of Systems and Software*, pages 143–153, 2013.

[92] Donald Molaro, Hannes Payer, and Damien Le Moal. Tempo: Disk drive power consumption characterization and modeling. *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pages 246–250, 2009.

[93] Miriam Allalouf, Yuriy Arbitman, Michael Factor, Ronen I. Kat, Kalman Meth, and Dalit Naor. Storage modeling for power estimation. *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference on - SYSTOR '09*, page 1, 2009.

[94] Anthony Hylick and Ripduman Sohan. A Methodology for Generating Disk Drive Energy Models Using Performance Data. *Proc of ACM SOSP Workshop on Power Aware Computing and Systems HotPower*, 80:100, 2009.

[95] Pablo Llopis, Manuel F Dolz, Javier Garcia Blas, Florin Isaila, Mohammad Reza Heidari, and Michael Kuhn. Analyzing the Energy Consumption of the Storage Data Path. *J. Supercomput.*, 72(11):4089–4106, 2016.

[96] Mohammed G Khatib, Zvonimir Bandic, and Santa Clara. PCAP : Performance-aware Power Capping for the Disk Drive in the Cloud. *Fast'16*, 2016.

[97] MIND: A black-box energy consumption model for disk arrays. *2011 International Green Computing Conference and Workshops, IGCC 2011*, 2011.

[98] Surendar Chandra. Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. *History*.

[99] Emanuele Lattanzi, Andrea Acquaviva, and Alessandro Bogliolo. Run-time software monitor of the power consumption of wireless network interface cards. *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, pages 352–361, 2004.

[100] Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Kieran Mansley. Characterizing 10 Gbps network interface energy consumption. *Proceedings - Conference on Local Computer Networks, LCN*, pages 268–271, 2010.

[101] Pablo Serrano, Andres Garcia-Saavedra, Giuseppe Bianchi, Albert Banchs, and Arturo Azcorra. Per-Frame Energy Consumption in 802.11 Devices and Its Implication on Modeling and Design. *IEEE/ACM Transactions on Networking*, 23(4):1243–1256, 2015.

[102] A.a b Garcia-Saavedra, B.c d Rengarajan, P.a Serrano, D.e f Camps-Mur, and X.e Costa-Pérez. SOLOR: Self-Optimizing WLANs With Legacy-Compatible Opportunistic Relays. *IEEE/ACM Transactions on Networking*, 23(4):1202–1215, 2015.

[103] Dynamic Resource Provisioning for Energy Efficiency in Wireless Access Networks: a Survey and an Outlook. *IEEE Communications Surveys & Tutorials*, PP(99):1–1, 2014.

[104] Vageesh D C., M Patra, and C Siva Ram Murthy. Joint placement and sleep scheduling of grid-connected solar powered road side units in vehicular networks. In *2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 534–540, 2014.

[105] Fatemeh Ganji, Anatolij Zubow, Łukasz Budzisz, and Adam Wolisz. On detecting WLAN users communication attempts. *2014 7th IFIP Wireless and Mobile Networking Conference, WMNC 2014*, (May), 2014.

[106] Luca Chiaraviglio, Delia Ciullo, Marco Mellia, and Michela Meo. Modeling sleep mode gains in energy-aware networks. *Computer Networks*, 57(15):3051–3066, 2013.

[107] Hang-sheng Wang and Li-shiuan Peh. Orion 2.0: A power-performance simulator for interconnection networks.

[108] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, 2003-January:93–104, 2003.

[109] Huazhe Zhang and Henry Hoffmann. Maximizing Performance Under a Power Cap : A Comparison of Hardware , Software , and Hybrid Techniques. *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 545–559, 2016.

[110] PowerPack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, 21(5):658–671, 2010.

[111] Gustavo Rostirolla, Rodrigo Da Rosa Righi, Vinicius Facco Rodrigues, Pedro Velho, and Edson Luiz Padoin. GreenHPC: A novel framework to measure energy consumption on HPC applications. *2015 Sustainable Internet and ICT for Sustainability, SustainIT 2015*, 2015.

[112] Juha Taina. How Green Is Your Software? In *International Conference of Software Business (ICSOB)*, volume 51, pages 151–162, 2010.

[113] Sara S. Mahmoud and Imtiaz Ahmad. A green model for sustainable software engineering. *International Journal of Software Engineering and its Applications*, 7(4):55–74, 2013.

[114] Stefan Naumann, Markus Dick, Eva Kern, and Timo Johann. The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, 1(4):294–304, 2011.

[115] Eugenio Capra, Chiara Francalanci, and Sandra A. Slaughter. Is software "green"? Application development environments and energy efficiency in open source applications. *Information and Software Technology*, 54(1):60–71, 2012.

[116] SIGAR. SIGAR.

[117] Jason Long. Javassist.

[118] ASM. ASM.

[119] BCEL. BCEL.

[120] Psutil. psutil.

[121] PhpSysInfo. phpSysInfo.

[122] PDH. PDH.

[123] PROC. PROC.

[124] Linfo. Linfo.

[125] Hayri Acar, Gulfem I. Alptekin, Jean Patrick Gelas, and Parisa Ghodous. Towards a green and sustainable software. In *Advances in Transdisciplinary Engineering*, volume 2, pages 471–480, 2015.

[126] Stephen Cass. The 2015 Top Ten Programming Languages, 2015.

[127] Javaagent. Javaagent.

[128] Jboss-javassist. Javassist.

[129] Minjoong Kim, Yoondeok Ju, Jinseok Chae, and Moonju Park. A simple model for estimating power consumption of a multicore server system. *International Journal of Multimedia and Ubiquitous Engineering*, 9(2):153–160, 2014.

[130] Lauri Minas and Brad Ellison. The Problem of Power Consumption in Servers. *Intel*, 2009.

[131] Ibrahim Hur and Calvin Lin. A comprehensive approach to DRAM power management. In *14th International Symposium on High Performance Computer Architecture*, pages 305–316, 2008.

[132] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. DRAMSim2: A cycle accurate memory system simulator. *IEEE Computer Architecture Letters*, 10(1):16–19, 2011.

[133] Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P Jouppi. Cacti 5.1. Technical report, 2008.

[134] Hayri Acar, Isiklar Alptekin, Jean-patrick Gelas, and Parisa Ghodous. Beyond CPU : Considering Memory Power Consumption of Software. In *Smartgreens 2016*, 2016.

[135] Brian Fonseca. IDC serves up top 10 storage predictions for 2008.

[136] Anthony Hylick, Ripduman Sohan, Andrew Rice, and Brian Jones. An analysis of hard drive energy consumption. *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS*, 2008.

[137] EFD Software. HD Tune Pro.

[138] John S Bucy, Jiri Schindler, Steven W Schlosser, and Gregory R Ganger. The disksim simulation environment version 4.0 reference manual. Technical report, 2008.

[139] John Zedlewski, Sumeet Sobti, Nitin Garg, Fengzhou Zheng, Arvind Krishnamurthy, and Randolph Wang. Modeling Hard-Disk Power Consumption. In *2nd USENIX Conference on File and Storage Technologies*, pages 217–230, 2003.

[140] Yan Zhang, Sudhanva Gurumurthi, and Mircea R. Stan. SODA: Sensitivity based optimization of disk architecture. In *44th Annual Design Automation Conference*, pages 865–870, 2007.

[141] Leandro Fontoura Cupertino, Jean-Marc Pierson, and Georges Da Costa. *Modeling the power consumption of computing systems and applications through Machine Learning techniques*. PhD thesis, Université Toulouse 3 Paul Sabatier, 2015.