Studies in Systems, Decision and Control 171

Vyacheslav Kharchenko
Yuriy Kondratenko
Janusz Kacprzyk *Editors*

Green IT Engineering: Social, Business and Industrial Applications

Springer

Studies in Systems, Decision and Control

Volume 171

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

The series "Studies in Systems, Decision and Control" (SSDC) covers both new developments and advances, as well as the state of the art, in the various areas of broadly perceived systems, decision making and control–quickly, up to date and with a high quality. The intent is to cover the theory, applications, and perspectives on the state of the art and future developments relevant to systems, decision making, control, complex processes and related areas, as embedded in the fields of engineering, computer science, physics, economics, social and life sciences, as well as the paradigms and methodologies behind them. The series contains monographs, textbooks, lecture notes and edited volumes in systems, decision making and control spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution and exposure which enable both a wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/13304

Vyacheslav Kharchenko · Yuriy Kondratenko
Janusz Kacprzyk
Editors

Green IT Engineering: Social, Business and Industrial Applications

Springer

Editors
Vyacheslav Kharchenko
Department of Computer Systems, Networks and Cybersecurity
National Aerospace University "KhAI"
Kharkiv, Ukraine

and

Center for Safety Infrastructure-Oriented Research and Analysis
Public Company RPC Radiy
Kropyvnytskyi, Ukraine

Yuriy Kondratenko
Department of Intelligent Information Systems
Petro Mohyla Black Sea National University
Mykolaiv, Ukraine

Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland

ISSN 2198-4182 ISSN 2198-4190 (electronic)
Studies in Systems, Decision and Control
ISBN 978-3-030-00252-7 ISBN 978-3-030-00253-4 (eBook)
https://doi.org/10.1007/978-3-030-00253-4

Library of Congress Control Number: 2018953316

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Green IT engineering (green information technologies engineering) is a special kind of computer technology based on energy saving and modern efficient information technologies. It could be considered as dual-purpose technology for (a) improving energy efficiency of computer systems and (b) increasing productivity, safety, and environmental performance of different industrial technological processes as well as information processing in social, business, and other applications.

The purpose of this book is to present a systematic account of research directions, results, and challenges, and a description and analysis of applications of green IT engineering and green IT approaches in different fields of human activity. Special attention is paid to the implementation of the green specialized software, programmable and hardware components, communications, cloud and IoT-based systems, and IT infrastructures to the engineering, business, social, and economy/management domains.

The monograph is a "de-facto" Volume 3 of the three volume sets in which the first book has been "Green IT Engineering: Concepts, Models, Complex Systems Architectures" and the second book has been "Green IT Engineering: Components, Networks and Systems Implementation", both are published by Springer in 2017. The present third volume, with the same first part of the title, i.e., "Green IT Engineering", aims at motivating researchers and practitioners from different IT domains to follow and advocate the ideas of green values for social, business, and industrial applications.

The first versions of all contributions were discussed at the workshops and seminars (United Kingdom, Italy, Portugal, Sweden, Ukraine) held within the international TEMPUS-project GreenCo and the ERASMUS + project ALIOT, as well as at the international conferences DESSERT, IDAACS, ICTERI during 2013–2018.

The book has a chapter-oriented structure according to the "vertical" structure of the green IT, from design and optimization of green IT systems to their applications in social, educational, business, and engineering fields.

The chapters are prepared according to a general paradigm and a unified scheme of contents, and describe step-by-step implementation processes of green IT in economy, management, and engineering, to just name a few.

In terms of structure, the 25 chapters of the book, presented by authors from different countries from all over the world: Estonia, Italy, Germany, Greece, Malaysia, Norway, Poland, Russia, Slovakia, Ukraine, the United Kingdom, and the USA, are grouped into four parts: (1) development and optimization of Green IT systems, (2) modeling and experiments with green IT systems, (3) industry and transport Green IT systems application, and (4) social, educational, and business aspects of Green IT systems application.

The chapters have been meant to provide an easy to follow introduction to the topics that are addressed, including the most relevant references, so that anyone interested in them can start his/her introduction to the topic through these references. At the same time, all of them correspond to different aspects of a work in progress being carried out in various research groups throughout the world and, therefore, provide information on the state of the art of some of these topics.

The first part, "Development and Optimization of Green IT Systems", includes six contributions: Chapter "Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues", by E. Kern, A. Guldner, and S. Naumann, introduces the idea of a green software and its engineering. Complementary to the definitions and models in the addressed field, a more practical insight is given by presenting illustrative examples of energy measurements of software products. While these aspects show that the research effort is increasingly dealing with software issues of Green IT, these mainly scientific ideas have hardly reached practical relevance, at least so far. Following the life cycle perspective of software products, the authors present examples of concepts on how to increase the awareness of green software, proposing (a) to implement continuous energy efficiency measurements during the development phase and (b) to create an eco-label for software products to inform about their environmental friendliness. The aim of the chapter is to point out solutions that may lead to a more environmentally friendly way of developing software, a well-informed procurement behavior regarding software products, and a more sustainable user behavior concerning information and communication technology (ICT).

E. D. Stetsuyk, D. A. Maevsky, E. J. Maevskaya, and R. O. Shaporin, in their Chapter "Information Technology for Evaluating the Computer Energy Consumption at the Stage of Software Development", describe an approach based on the absolute value estimation of the computer devices energy consumption. It can help to choose an optimal energy saving solution at the software development stage. There are three stages for the energy consumption estimation in the proposed information technology. The first stage is the scanning of assembler code, determination of assembler instruction, and their volume in bits. At the second stage, the method for the estimated power consumption based on RAM types is applied. The third stage boils down to the estimation of the full energy consumption. For the implementation of information technology, the program tool

is developed. This tool helps to create a green software for checking the energy consumption of computer devices at all stages of development.

In their Chapter "Green IT Engineering: Green Wireless Cooperative Networks", S. Fu and J. Wu consider the methods for securely performing the calculations required for the fundamental modular arithmetic operations, namely, the multiplication and exponentiation using mobile, embedded, remote, or distant computational resources which offer the possibility of development of the green information processing systems. These methods are targeted at the distributed paradigms of cloud computing resources and the Internet of Things (IoT) applications. They provide security by avoiding the disclosure to the cloud resource of either the data or the user secret key. Simultaneously, environmental effects of processing are minimized by the simplifications of the operations and by transferring demanding calculations to energy-efficient data centers. The developed modular multiplication algorithm provides a faster execution time on low complexity hardware in comparison with the existing algorithms and is oriented toward the variable value of the modulus, especially with the software implementation on microcontrollers and smart cards whose architectures include a small number of bits.

A. Drozd, S. Antoshchuk, J. Drozd, K. Zashcholkin, M. Drozd, N. Kuznietsov, M. Al-Dhabi, and V. Nikul, in Chapter "Checkable FPGA Design: Energy Consumption, Throughput and Trustworthiness", consider the green FPGA design processes based on the energy efficiency and safety for the critical applications. The array structures that are traditionally used in digital components of safety-related systems reduce a checkability of circuits, creating a problem of hidden faults which can be accumulated in a normal mode and reduce the fault tolerance of the circuit and the safety of system in the emergency mode. Soft and cardinal ways of array structure reduction are proposed based on the development of the truncated arithmetical operations implemented in reduced array structures and on the parallelization of calculations in serial codes with the use of bitwise pipelines. The comparative analysis in complexity, throughput, and energy consumption of the iterative array and bitwise pipeline multiplier is executed experimentally with use of Altera Quartus II.

In Chapter "A Prospective Lightweight Block Cipher for Green IT Engineering", A. Andrushkevych, Y. Gorbenko, O. Kuznetsov, R. Oliynykov, and M. Rodinko consider general requirements for modern block ciphers, especially for the implementation at lightweight cryptographic transformations for critical distributed environment applications with a green IT conformance. The overview of the well-known block ciphers and lightweight primitives PRESENT and CLEFIA, defined at ISO/IEC 29192-2, as well as a specification of the lightweight block cipher Cypress are presented. The Cypress performance in software is approximately three times higher than the AES one based on Windows, Linux, and Android platforms.

In Chapter "Lightweight Stream Ciphers for Green IT Engineering", O. Kuznetsov, O. Potii, A. Perepelitsyn, D. Ivanenko, and N. Poluyanenko introduce the symmetric cryptographic transformations, in particular, the stream ciphers. The development of an efficient synchronous stream cipher is reduced to the

construction of a pseudo-random sequence generator with defined cryptographic properties. Limited physical parameters, low power consumption, low computing power, and other characteristic attributes of the "green" IT engineering forces the use of new approaches for designing cryptographic protection tools. The authors propose new methods and hardware/software tools for a lightweight stream encryption that meet the current requirements of the "green" IT engineering.

The second part, "Modelling and Experiments with Green IT Systems", includes eight contributions. In Chapter "Semi-Markov Availability Model for Infrastructure as a Service Cloud Considering Energy Performance", O. Ivanchenko, V. Kharchenko, B. Moroz, L. Kabak, I. Blindyuk, and K. Smoktii present a stochastic approach based on the semi-Markov process modeling in order to determine the availability level and energy performance for the Infrastructure as a Service (IaaS) cloud. Today, the cloud service providers offer diverse IT services using a large number of physical machines (PMs). The authors propose to use information about technical and information state of a control system in order to improve the monitoring process for the IaaS cloud. Special attention is paid to (a) the development of a monolithic semi-Markov availability model for the IaaS cloud and (b) the evaluation of power consumption for non-failed, that is available, PMs.

G. Kuchuk, A. Kovalenko, I. E. Komari, A. Svyrydov, and V. Kharchenko, in Chapter "Improving Big Data Centers Energy Efficiency: Traffic Based Model and Method", discuss the green aspects of data centers (DCs) which provide the appropriate hosting services and data storage. Various approaches to transmission control of integrated data streams, which are the result of traffic aggregation, being transmitted in a DC related to Big Data processing, are analyzed. The authors develop several models and a method which help reduce the DC's resources usage for data processing by 3%. The application of the proposed models allows the use of resources of the DC in a more efficient way providing the reduction of the transmission time and, correspondingly, an increased energy efficiency of the system.

Chapter "A Markov Model of IoT System Availability Considering DDoS Attacks, Patching and Energy Modes", by M. Kolisnyk and V. Kharchenko, is focused on an important task in the conditions of the rapid spread of the IoT, that is the assessment of its availability under the impact of various attacks. The authors (a) analyze the energy modes of network devices operating system in the smart business center (SBC) and specify the organization of an energy-efficient IoT-SBC subsystem; (b) develop and study Markov models of the SBC availability taking into account the energy regimes, the impact of cyberattacks, the reliability of SBC components, and patches on the firewalls vulnerabilities; and (c) analyze possible methods for reducing the power consumption of the SBC devices estimating the availability factor of SBC in the conditions of successful attacks.

In Chapter "Assessing the Impact of EEE Standard on Energy Consumed by Commercial Grade Network Switches", J. El Khoury, E. Rondeau, J.-P. Georges, and A.-L. Kor note that the reduction of the power consumption of network equipment has been both driven by a need to reduce the ecological footprint of the cloud as well as the immense power costs of data centers. As data centers, core

networks, and—consequently—the cloud constantly increase in size, their power consumption should be mitigated. The Ethernet, the most widely used access network, still remains the biggest communication technology used in core networks and cloud infrastructures. As statistics shows, the average utilization rate of the Ethernet links is 5% on desktops and 30% in data centers, so that the power saving potential of EEE could be immense. Experimental results show that the base power consumption of switches does not significantly increase with the size of the switches. Doubling the size of the switch between 24 and 48 ports increases power consumption by 35.39%. EEE has a greater effect on bigger switches, with a power (or energy) gain on the EEE-enabled 48-port switch compared to 2 × EEE-enabled 24-port switch.

C. V. Vasile, C. Pattinson, and A.-L. Kor, in Chapter "Mobile Phones and Energy Consumption", present an analysis of different relevant concepts and parameters that may have impact on energy consumption of Windows Phone applications. This operating system was chosen because there is limited research even though there are related studies for the Android and iOS operating systems. The results for group of experiments are discussed taking into account the duration of the experiment, the battery consumed in the experiment, the expected battery lifetime, and the energy consumption, while the charts display the energy distribution based on the main threads: UI thread, application thread, and network thread.

Chapter "Energy-Efficient Multi-fragment Markov Model Guided Online Model-Based Testing for MPSoC", by J. Vain, L. Tsiopoulos, V. Kharchenko, A. Kaur, M. Jenihhin, J. Raik, and S. Nõmm, describes an efficient online testing method of autonomous mission critical systems (AMCS) in service modes. The multi-fragment Markov models (MFMM) are used for specifying the system reliability and quality-related behavior on high level of abstraction, and the more concrete state and timing constraints are specified explicitly using Uppaal probabilistic timed automata (UPTA). To interrelate these models, the authors demonstrate how the MFMM is mapped to UPTA. The second contribution is the test case selection mechanism for the online identification of AMCS service modes by a model-based conformance testing. The efficiency of active mode sampling is achieved by serializing the test cases for sampling session using hypotheses provided by MFMM.

In Chapter "Decrease of Energy Consumption of Transport Telecommunication Networks Using Stage-by-Stage Controlling Procedure", G. Linets and S. Melnikov propose a new approach to the development of an advanced automatic monitoring system and adaptive control of a transport telecommunication network which make it possible to reduce the energy consumption of switching centers during the analysis and identification of faults and failures in the equipment operation. The energy reduction is attained due to the use of a procedure of finding out abnormal situations, error optimization of the identification of system states, the use of a multi-agent setting, and the use of a permission for agents dependent on a hierarchy level of the transport telecommunication network.

I. Turkin and O. Vdovitchenko, in their Chapter "Model and Methods of Human-Centered Personal Computers Adaptive Power Control", consider two main approaches for solving the problem of reducing the computers' power consumption at the hardware and at the operating system levels, by (a) a dynamic change of the processor voltage and frequency; (b) an advanced configuration's interface and power management technology. The chapter is devoted to the development of the user's activity model and method of its identification in the system of PC's power consumption.

The third part "Industry and Transport Green IT Systems Application" includes six contributions. Chapter "Green Assurance Case: Applications for Internet of Things" by V. Sklyar and V. Kharchenko discusses the "green assurance case," in particular, the assurance case methodology for its implementation for the green IT domain related to the Internet of Things. Fundamentals of the concept of an assurance case are explained as the initial part of the article. Safety and security requirements are considered as a part of a typical assurance case. The green assurance case for the IoT includes the following parts: sustainability assurance part, safety and security assurance part, Green IT principles, power consumption parameters part, as well requirements to green features of of the four IoT layers. An example of the proposed green assurance case methodology for a medical IoT system is represented by an application.

O. Kozlov, G. Kondratenko, Z. Gomolka, and Y. Kondratenko, in their Chapter "Synthesis and Optimization of Green Fuzzy Controllers for the Reactors of the Specialized Pyrolysis Plants", present the developed generalized method for the synthesis and optimization of a green fuzzy controllers (FC) for an automatic control systems (ACS) of the reactor's temperature of a specialized pyrolysis plants (SPP). The proposed method gives the opportunity to synthesize and optimize the Mamdani-type green FCs that provide (a) high values of accuracy and quality indicators of temperature control, (b) low energy consumption in the process of functioning, as well as (c) relatively simple software and hardware implementation. The design of the Mamdani-type green FC for the temperature ACS of the pyrolysis reactor of the experimental SPP has been carried out in order to study and validate the effectiveness of the developed method. The developed green FC has a relatively simple hardware and software implementation and makes it possible to attain high values of quality indicators of temperature modes control at a sufficiently low energy consumption that confirms the high efficiency of the proposed method.

In Chapter "Models and Algorithms for Prediction of Electrical Energy Consumption Based on Canonical Expansions of Random Sequences", I. Atamanyuk, V. Kondratenko, Y. Kondratenko, V. Shebanin, and M. Solesvik discuss the development peculiarities and the use of models and algorithms as elements of the green technology to predict the electric energy consumption based on the mathematical apparatus of canonical expansions of random sequences. The developed calculation method does not impose any limitations on the qualities of random sequences of the change of electric energy consumption (exemplified by the requirements of linearity, Markovian behavior, monotonicity, stationarity, etc.). A comparative analysis of the results of a numerical experiment with the use of the

Kalman filter and the linear prediction method confirms the high efficiency of the developed models and algorithms (the relative error of prediction of electric energy consumption is 2–3%).

Chapter "On Quantitative Assessment of Reliability and Energy Consumption Indicators in Railway Systems", by D. Basile, F. Di Giandomenico, and S. Gnesi, addresses cross-validation on a case study from the railway domain by modeling and evaluating it using different formalisms and tools. Stochastic activity networks models and stochastic hybrid automata models of railroad switch heaters, developed for the purpose of evaluating the energy consumption and reliability indicators, are evaluated with the Mobius and Uppaal SMC. The authors compare the obtained results to improve their trustworthiness and to provide insights into the design and analysis of energy saving cyber-physical systems.

A. Orekhov, A. Stadnik, and V. Kharchenko, in Chapter "Cooperative Ecology Human–Machine Interfaces for Safe Intelligent Transport Systems: Cloud-Based Software Case", analyze the available solutions regarding the cooperative intelligent transport systems (ITS) and their human–machine interfaces (HMI). An approach and a design technique of HMIs for ITS based on designing the cooperative HMIs for intelligent transport systems are suggested. Special attention is paid to the architecture of cloud-based HMI for intelligent transport systems that consist of three main parts (the client end, server end, and the core project) and includes data models for the communication protocol and the common utility functions.

Chapter "Optimisation of Bi-directional Pulse Turbine for Waste Heat Utilization Plant Based on Green IT Paradigm", by Y. Kondratenko, S. Serbin, V. Korobko, and O. Korobko, is devoted to the environment preservation problem which is closely connected with issues of rational use of energy and material resources. By analyzing the state of the art of modern energy saving technologies, it is possible to conclude that there is a lack of effective and efficient solutions for utilizing waste low-temperature and cryogenic heat resources. The thermoacoustic heat machines (TAHM) can effectively use such energy sources. TAHM are characterized by high reliability, relative low cost, and environmental safety. Suchan approach will enable the creation of thermoacoustic turbine generators of a high power. The chapter presents the results of experimental research of the "bi-directional turbine—electric generator—TAHM" complex. A special computer-based control system was developed for the registration and analysis of experimental data. The system includes (a) MCU-based pulse-phase control subsystem for thermoacoustic engine heater, (b) data acquisition subsystem, and (c) SCADA-based PC software for data storage and analysis. The control system, designed using the green IT approach paradigm, is described in detail. The results of physical experiments make it possible to determine the boundary conditions for the computational fluid dynamics (CFD) simulation of working processes in the bi-directional turbine. The structural optimization of a turbine is considered based on the CFD modeling results.

The fourth part "Social, Educational and Business Aspects of Green IT Systems Application", including five contributions. V. Hahanov, O. Mishchenko, T. Soklakova, V. Abdullayev, S. Chumachenko, and E. Litvinova, in Chapter "Cyber-Social Computing", discusses structures of cyber-social computing which are considered as components of cloud-driven technologies for an exact monitoring and moral governance of the society. The main trends in the development of the cyber-physical structure presented in Gartner's Hype Cycle 2017 are described with the aim to apply them in science, education, transport, industry, and state structures. An expanded description of technologies is focused on the development of the smart digital world, green cities, and 5G telecommunications. The recommendations are given for leveraging the top 10 technologies of 2017 in business, scientific, and educational processes of higher education. Special attention is paid to the technologies of emotional–logical computing, metrics for measuring social relations along horizontal and vertical connections, rules and simulation of human behavior, and cyber-physical model of a green statehood for the metric management of resources and citizens.

In Chapter "Architecture for Collaborative Digital Simulation for the Polar Regions", M. Solesvik and Y. Kondratenko focus on offshore oil and gas operations which continuously move to the High North and to the Arctic which implies that problems of possible ship accidents and catastrophes are extremely dangerous for the nature. The authors analyze the recent literature on the new digital simulation technology, propose new ways for the utilization of this novel technology in the maritime industry, present fresh insights related to this architecture, software and hardware related to joint digital simulation platforms for planning safe maritime operations in the Arctic, and effective search and rescue operations. The utilization of joint digital simulation platforms will allow to minimize pollution effects during the navigation and accidents.

Chapter "Computer Support for Decision-Making on Defining the Strategy of Green IT Development at the State Level", by I. Shostak, I. Matyushenko, Y. Romanenkov, M. Danova, and Y. Kuznetsova, presents a complex estimation method of an innovative potential of the national economy in order to determine the prospects of the development of the green IT technology. The methodological basis of the method is cluster analysis. Data analysis is used for predicting the development of the green IT technologies on the dynamic time series. As a predictive model, the authors use the Brown model and a special structural parametric synthesis method that implements the identification of a time series by a preliminary analysis and processing of initial data, and identification of the trend and formation via interval estimates. The results of forecasting may be used, along with other initial data, for a foresight-type research on the definition of the green IT prospective directions of development. Special attention is paid to the generalized structure of interactive decision support system for the development of the green IT at the state level.

K. Czachorowski, M. Solesvik, and Y. Kondratenko, in their Chapter "The Application of Blockchain Technology in the Maritime Industry", present fresh insights related to the application of the novel blockchain technology for reducing

pollutions, analyze recent literature on blockchain technology, and propose ways of the utilization of blockchain technology in the maritime industry. The technology has a broad range of applicability making it possible to connect the supply chain more efficiently, providing the exchange and visibility of time-stamped proofed data, and decreasing the industry operational costs with intermediaries and increasing security. It also allows a full visibility for all parties involved with a proof of work, facilitating the class societies inspections, port state control, and audits compliance. The results of the study also show that cases of blockchain application in other fields increase the industry willingness to its application in the maritime industry. Having blockchain implementations, specialized third parties would increase the implementation possibility and the industry willingness due to reduced costs and friction.

In Chapter "Software Engineering Sustainability Education in Compliance with Industrial Standards and Green IT Concept", I. Turkin and Y. Vykhodets present a scientific approach to building the course "Software Engineering Sustainability" to introduce the sustainable concept into educational programs for software engineers. The authors consider the peculiarities of the course design with description of the course content and visible positioning green IT.

To briefly summarize, the chapters selected for this book provide an overview of some crucial problems in the area of green IT engineering and of approaches and techniques that relevant research groups within this area are employing to try to solve the problems considered.

We would like to express our deep appreciation to all authors for their contributions as well as to reviewers for their timely and interesting comments and suggestions. We certainly look forward to working with all contributors again in nearby future.

We also wish to thank Dr. Tom Ditzinger, Dr. Leontina di Cecco, and Mr. Holger Schaepe from Springer for their dedication and help to implement and finish this publication project on time maintaining the highest publication standards.

Kharkiv, Ukraine — Prof. Dr. Sc. Vyacheslav Kharchenko
Mykolaiv, Ukraine — Prof. Dr. Sc. Yuriy Kondratenko
Warsaw, Poland — Prof. Dr. Sc. Janusz Kacprzyk
April 2018

Contents

Part I
Development and Optimization of Green IT Systems

Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues

Eva Kern, Achim Guldner and Stefan Naumann

Abstract While counteracting the increasing demand for natural resources and especially energy of ICT, first successes have become apparent by activities comprised by the term "Green IT". Nowadays, many of the current activities lay emphasis on the hardware side of Green IT. However, software issues play a significant role in defining system and hardware requirements as well as the amount of energy consumed by ICT devices and the underlying infrastructure. Thus, the following chapter introduces the idea of green software and its engineering. Complementary to definitions and models in the addressed field, a more practical insight is given by illustrating exemplary energy measurements of software products. While these aspects show that the research is increasingly dealing with software issues of Green IT, these mainly scientific ideas have hardly reached practical relevance, so far. Hence, following the life cycle perspective of software products, we present two exemplary concepts on how to increase awareness of green software: addressing especially software engineers, we propose to implement continuous energy efficiency measurements during the development phase. With regards to software users, we propose to create an eco-label for software products to inform about their environmental issues and thus create more transparency in this context. To do so, we present and evaluate criteria and indicators, upon which a label could be based. The chapter concludes with a summary and proposes future activities in the addressed field. Overall, the aim of the chapter is to point out solutions that

E. Kern (✉)
Leuphana University of Lüneburg, Universitätsallee 1, 21335 Lüneburg, Germany
e-mail: mail@nachhaltige-medien.de

E. Kern
University of Applied Sciences Trier, Environmental Campus Birkenfeld, Birkenfeld, Germany

A. Guldner · S. Naumann
University of Applied Sciences Trier, Environmental Campus Birkenfeld, Institute for Software Systems, Birkenfeld, Germany
e-mail: a.guldner@umwelt-campus.de

S. Naumann
e-mail: s.naumann@umwelt-campus.de

V. Kharchenko et al. (eds.), *Green IT Engineering: Social, Business and Industrial Applications*, Studies in Systems, Decision and Control 171,
https://doi.org/10.1007/978-3-030-00253-4_1

might lead to a more environmentally friendly way of developing software, a well-informed procurement behavior regarding software products, and a more sustainable user behavior concerning ICT.

Keywords Green software · Energy measurements · Energy efficiency
Eco-label · Green software development · Environment

1 Introduction

Regarding a sustainable development, as firstly published in the Brundland Report [1] and nowadays put straight into 17 United Nations' Sustainable Development Goals (SDGs),[1] information and communication technologies (ICT) play a special part: On the one hand, ICT consumes a lot of resources during its production and usage [2] and will consume even more in the future [3]. On the other hand, these technologies can support environmental processes and even make processes more sustainable in many different branches [4]. The latter is referred to by the term "Green by IT".

In order to counteract the rising resource consumption by ICT, activities in the field of Green IT found its way into research, industry, and private households. While they are mainly focused on hardware aspects, we emphasize the software side, since "software characteristics determine which hardware capacities are made available and how much electric energy is used by end-user devices, networks, and data centers" [5]. Thus, "referring to software energy consumption or efficiency, we are actually referring to the energy consumed by the hardware when induced by the software to perform some task(s)" [6]. Referring to a sustainable development, Maevsky et al. [7] show the economic relevance of considering the software side of Green IT by presenting a corresponding calculation model. However, we focus on the environmental issues of sustainable software within the following chapter. Additionally, they claim that the "amount of electrical energy consumed by the computer is largely dependent on the program compilation quality and Software optimization degree." [7]

Within the Green IT community, as well as the groups specialized in Green Software, research activities are increasing during the last years. This is, for example, shown by raising numbers of publications, conferences, information events, workshops, and individually reported in sustainability and environmental reports of the big players in the worldwide IT market. However, especially referring to the software side, the awareness on environmental aspects could be higher in industry and in daily (private) routines. Different studies show that even if the interviewed person knows Green IT strategies, the transfer towards practical

[1] https://sustainabledevelopment.un.org.

implementation and into daily routines is missing—on the professional as well as on the private side [8–10].

Thus, after introducing the research field "green software engineering" (Sect. 2), we will present approaches on how to support especially software developers and users in considering environmental issues of software (Sect. 3). The approaches are linked to the life cycle of software products, based on the "from cradle to cradle" principle. Generally, the concepts can contribute to reducing the overall carbon footprint of ICT and to the United Nations' SDG 12 "Ensure sustainable consumption and production patterns". The chapter concludes with a brief summary and an outlook on potential future research activities (Sect. 4).

2 State of the Art: Green Software and Its Field of Research

The following section will introduce the topic of green software in general and present definitions for central terms, since "[defining] and developing adequate support requires a commonly accepted definition of what sustainability means in and for software engineering" [11]. However, we do not aim at presenting a comprehensive literature overview with the following section. This can be found e.g. in [12, 13].

2.1 Definitions for Green and Sustainable Software: Overview, Commonalities, and Distinctions

Similarly to Green IT and Green by IT (see Introduction), in combining sustainability issues to the software side, "the differentiation can be made by distinguishing software (engineering) for sustainability, which is related to the absolute definition, and sustainable software (or sustainability in software engineering), which is related to the relative definition" [14]. While the so-called "absolute definition" refers to approaches that reduce the environmental impact of a process by using software, Taina [15] talks about "Software Engineering for the Planet (SEP)" referring to the "sustainable by software" dimension. According to their understanding, "Green IT belongs to SEP [and] includes other areas as well". These are [15]: (1) Software support for green education, (2) Green metrics and decision support, (3) Lower IT energy consumption, and (4) Support for better climate and environment models.

Concepts on "sustainable (in) software" deal with possibilities to improve the software product itself. Even if referring to sustainable software in the sense of the Brundlandt Report, many approaches rank environmental aspects first (e.g. [12, 16, 17]). In addition to environmental, Lago et al. [18] describe five dimensions that could be interpreted as five perspectives on sustainable software:

- Social sustainability: Which effects do software systems have on the society (e.g. communication, interaction, government…)?
- Environmental sustainability: How does software affect the environment during, inter alia, development and maintenance?
- Technical sustainability: How can software be created so that it can easily adapt to future changes?
- Economic sustainability: How can software systems be created so that the stakeholders' long-term investments are as safe as possible from economic risks?
- Individual sustainability: How can software be created and maintained in a way that enables developers to be satisfied with their job over a long period of time?

Dividing the concept of sustainability into different dimensions—let it be three, four or five—or applying criteria to it makes the sustainability of a product feasible. Hence, several researchers developed corresponding criteria (e.g. [15, 16, 19, 20]). While Taina [15] distinguishes between feasibility, (carbon footprint, energy, travel, etc.), efficiency (CPU intensity, idleness, etc.), and sustainability (reduction, beauty, etc.), Calero et al. [20] divide sustainability into the three sub characteristics energy consumption, resource optimization, and perdurability. In our project "Sustainable Software Design"[2] we focus on the first of the proposed categories and will publish a set of criteria for sustainable software products. It is hierarchically structured in the main criteria "resource efficiency", "potential hardware operating life", and "user autonomy". The criteria are operationalized by appropriated indicators. In this context, Betz et al. [21] differentiate between "resource oriented indicators", taking environmental aspects of sustainability into account, and "well-being oriented indicators", referring to "human needs aspects of sustainability".

Bringing the different approaches together, the sustainability of software products can be interpreted as a non-functional requirement [21, 22] and thus "the sustainability assessment would be considered as another quality aspect to be taken into account by developers, in accordance with the priorities and requirements imposed for the product being developed" [22]. Seeing it as a quality criterion for software, sustainability is an improvement of software products. Taking this approach, literature talks about "green/sustainable software engineering/development". The aim of this process is the "development of the software product for long living systems that can meet the needs of the present to the future generations with the integration of the three pillars sustainability concept i.e. (environment, economic, social) as to fulfill the requirements in a timely basis." [23]. Thus, "sustainability of a software product [can be defined] as the capacity of developing a software product in a sustainable manner" [20].

Summarizing, the commonalities in the understanding of the term "(green and) sustainable software" are [24]: "(1) environmental and resource protection as well as (2) supporting sustainable development including the three pillars of

[2] http://green-software-engineering.de/en/projects/ufoplan-ssd-2015.html.

sustainability, but setting priorities." None of the definitions stands in total contrast to the other ones but each of them represents a specific perspective that must be kept in mind while talking about green, or rather, sustainable software, since it is strongly dependent on the currently defined view [11]. For the purpose of this chapter, we refer to our definition of green and sustainable software: "[Green and sustainable] software is 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." [16]

We lay emphasis on environmental issues of sustainability and use the terms "green software" and "sustainable software" in an analogous manner. A more comprehensive engagement with the terminology can be found in e.g. [17, 25].

2.2 *Green Software Models: Quality Models, Development Models, and Reference Model*

Based on a mutual understanding of green and sustainable software, different models comprise even more aspects that belong to the addressed research field. Our GREENSOFT Model (Fig. 1), presented in [16], takes a broad look at green software and its engineering.

The reference model comprises four parts: the life cycle of software products, sustainability criteria and metrics, procedure models, and recommendations and

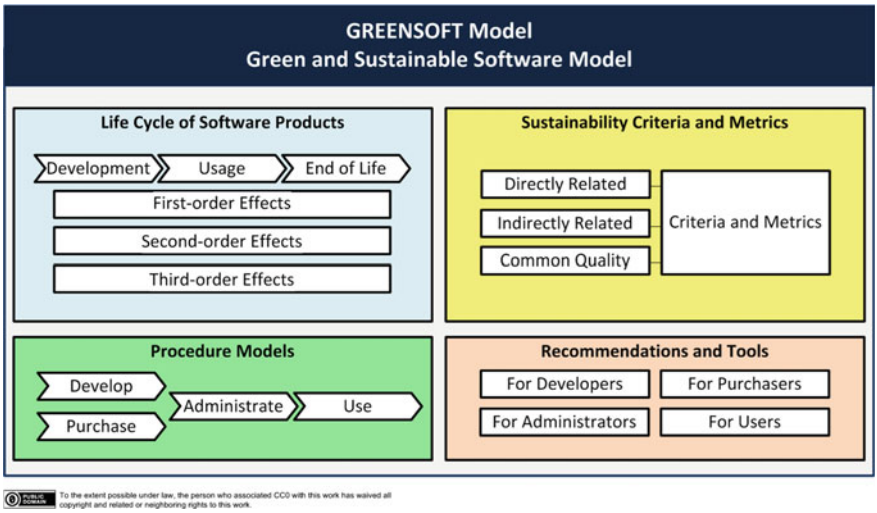

Fig. 1 The GREENSOFT model, a reference model for green and sustainable software engineering [27]

tools. Thus, it structures concepts, strategies and processes in the field of Green (by) IT, focusing on software aspects. Following the approach of a reference model, e.g. as presented in [26], the idea is to present an overview of the field and to structure several aspects in the context of green and sustainable software engineering. Hence, other models can be assigned to the GREENSOFT Model.

Existing life cycle approaches in the context of green software (being the first part of the model), from which criteria and metrics can be derived, will be presented in the next section. Generally, this approach considers ecological, social, and economic aspects of products over their whole life cycle.

The second model part "Sustainability Criteria and Metrics" covers general criteria and metrics regarding quality of software and allows a classification of the sustainability of the products. Based on the approach by Berkhout et al. [28], we distinguish between first-order ("direct environmental effects of the production and use of ICTs"), second-order ("indirect environmental impacts related to the effect of ICTs on the structure of the economy, production processes, products and distribution systems"), and third-order effects ("indirect effects on the environment, mainly through the stimulation of more consumption and higher economic growth by ICTs ('rebound effect'), and through impacts on life styles and value systems"). Calero et al. [29] use the modelling approach to go deeper into characterization. They take the ISO Model 25010 as starting point and identify characteristics to be included in the model in order to explicitly integrate sustainability aspects of software. Possible characteristics of green and sustainable software products are also summarized within our quality model, presented in [19]. Both models [19, 29] can be placed in the second part of the GREENSOFT Model. Even more possibilities on how to include green software characteristics in existing standard software quality models, are proposed by Gordieiev et al. [30]. They analyze well-known software quality models with regards to green and reliability issues and the evaluation of these characteristics over time. In conclusion, they "assume that the next general [Software Quality Model] will include [Green Software] characteristics in an explicit form" [30].

Lami et al. [31] distinguish between the "software lifecycle" and "software processes" and identify sustainability factors belonging to different processes. They present a model to enable an "evaluation of the degree of process sustainability of an organization" and, in a next step, improve the sustainability of soft-ware processes. Thus, referring to Fig. 1, the model presented by Lami et al. belongs to "Procedure Models". Here, procedure models refer to the development and purchasing processes of software as well as user support while administrating and using the product. In this context, verification regarding efforts and costs is another important aspect of developing software. Thus, we recommend logging the corresponding efforts and costs of the process, especially to be able to contrast sustainable software engineering with "standard procedures". One example how to do so is the "Process Model for Green and Sustainable Software Engineering", that includes a "Sustainability Review and Preview" as presented in [32].

Finally, the GREENSOFT Model mentions "Recommendations and Tools". Thereby, it points out the importance to integrate different roles into the activities

aiming at creating a more sustainable ICT and aims at putting the model into practice: developers, purchasers, administrators, and users need context-specific knowledge, in form of recommendations, tools, and methods to support a sustain-able way of developing, purchasing, administrating, and using software. Exemplarily for the whole software life cycle, we present tools to create awareness and, in the long run, deeper knowledge, of environmental issues of software and thus a way of developing and using software in a green and sustainable manner.

3 Creating Awareness of Green Software: Life Cycle Perspective

The proposed life cycle for software products orients towards a cradle-to-grave approach [33]. The development phase considers impacts on a sustainable development directly or indirectly caused by activities belonging to the software development. Examples for those impacts are the energy consumption caused by ICT devices that are used to program the software and by commuting of the software developers that results in CO_2 emissions. Additionally, also social aspects find their way into this consideration, e.g. working conditions [34]. Following the development phase, the distribution of the product plays a role regarding a sustainable software life cycle. Here, resource consumption caused by the chosen data medium, or rather the download size, are examples for first-order effects. The us-age phase results especially in environmental effects: software induced energy and resource consumptions, hardware requirements and portability with regards to durability are some of the relevant aspects. However, considering issues like transparency and accessibility also relates this phase to social sustainability. Looking at the end of a software's life, deactivation (in regard to e.g. backup size, long term storage of data) and disposal (e.g. packaging, data medium) have an influence on the sustainability of the considered product.

The objective of the life cycle of software products included in the GREENSOFT Model "is to enable stakeholders to assess impacts on [sustainable development]" [16]. Thus, the question "Who are the stakeholders in context of a life cycle of (sustainable) software products?" arises. While Penzenstadler et al. [35] identify stakeholders, who are important for successfully implementing sustainability issues, Herzog et al. [36] list actors developing and actors supporting innovations in the context of Green IT. Concerning our research, we especially take into account stakeholders of the development and the usage phase. This is due to the fact that if it is possible to create or increase their awareness of green software, this can yield the biggest impact, since they are directly connected to software products—either in an active (developing) or a passive (using) role. In both cases, tools to support a more sustainable way of developing and especially using software products are required. Thus, we present two possible tools within the following section.

Here, we lay emphasis on the development and usage phase of the software life cycle, since an "early awareness of green software could save a significant amount of costs compared to refactoring the software at a later stage." [37]. Maevsky et al. [7] and Chemeris et al. [38] also stress the importance of the software development phase and advances in terms of power consumption when adopted early in the design process. Chemeris et al. [38] show the positive effect of program optimization in case of program loops on the resulting power consumption.

However, in most cases, the programmers do not even know about their influence or rather lack of knowledge on energy and sustainability issues regarding software [8, 9, 39]. Sustainability of software is, according to Groher et al. [39], mainly related to "maintainability and long-term usability". Overall, "[technical] and organizational aspects are clearly in focus, followed by economic aspects. Environmental considerations are missing." [39] If practitioners have knowledge in these contexts, they do just minimally take them into account during software development [8], and just connect them to specific kinds of software products (e.g. mobile development [8]). Generally, the transfer of knowledge to practice is missing [9, 40]. Even if the different studies do not agree on the existing amount of awareness on energy, or rather sustainability issues in software engineering, they agree that the potential is still not exhausted, so far. In total, the "lack of attention to software energy consumption is an issue of priorities" [8].

The same applies for users of software products: 51.1% of the respondents of a user study, we conducted in 2016, can imagine taking "certification of environmental issues of the software product" as one possible buying criterion, if there are products with the same functionality, some being "green" and others being "not green" [41]. Overall, the sample of our survey comprised 712 completed questionnaires, primarily answered by German users (Table 1). The survey participants set the priority on the functionality of a software product.

Similarly to software engineers, users might have some ideas on green computing [10, 42], energy [6], or rather sustainability issues of software [43], but awareness, information, and education are missing [6, 8, 10, 42, 43]. Ferreira [6] provides the hypothesis "Given the clear awareness, respondents are taking energy

Table 1 Key data of the survey on the awareness of and the interest in environmental issues of software, conducted in 2016

Method	Online survey
Structure of the survey	Demographic characteristics of the participants Evaluation of criteria for green software products Aspects influencing the acceptance of an eco-label for software products
Participants	German speaking Internet user
Sample	n = 854
Completed questionnaires	712 (revised data set)
Survey period	16/08 to 05/10/2016

consumption into consideration when buying or developing software applications". In this context, Selyamani et al. [42] point out that "[awareness] is a need to practice an idea of the new things. Without awareness, the practice does not do appropriately and adequately and the other way around."

3.1 Development: Continuous Energy Efficiency Measurements

One way to improve awareness of sustainable software, especially with developers and users, is to provide a means for comparable measurements. While other references talk about "software development life cycles" [44, 45], we focus on tools that can be directly integrated in existing development processes and thus support developers in producing sustainable software products. According to Anwar et al. [13], current research activities in the context of green software engineering are missing supporting tools, e.g. "energy aware testing tools". They especially mention tools that evaluate the energy consumption during development as one example of research gaps that need to be closed. Such kinds of tools are required to produce green software products. Currently, "software practitioners prefer to use static analysis and energy profiling in order to point out energy problems during development" [40].

Mahmoud et al. [44] also follow the idea of identifying how software causes energy and resource consumption but take a broader view and thus present a theoretical approach. Shenoy et al. [45] lay the focus on more general guidelines and recommendations for developing software in a sustainable manner. In comparison to both, we outline a more practical approach and its application.

Taking one step back towards defining sustainability goals for the product to be developed, Mahaux et al. [46], Becker et al. [47], and others lay emphasis on requirements engineering and the design of sustainable software.

In order to get an impression of the environmental impact of the software product and to take sustainability issues into account while developing software, we propose to continuously integrate energy efficiency measurements into the development process. The tool presented below can both be integrated in the development phase or used for stand-alone measurements. In that way, it is possible to provide "actionable timely information, to make useful trade-offs between energy efficiency and other quality attributes" as claimed by Anwar et al. [13]. Over time, providing information about the energy consumption of software can help developers to get a feeling for the energy values of the products they are working on [40]. Thus, this information might positively influence the awareness of software engineers regarding environmental issues of software. Additionally, our approach focuses "on rating energy efficiency during the development process on an on-going basis" and is "based on well-known software testing approaches and [continuous integration], to take energy efficiency into account during the daily

work of a software developer" [48]. Hence, in order to meet the requirements of practitioners [39], our approach for energy efficiency measurements can be adapted in existing working structures very well. Furthermore, the approach can also be used to compare the energy consumption of two or more software products that perform the same task (e.g. standard software like word processors or databases).

The approach is based upon a measurement setup that follows ISO/IEC 14756, as introduced in [49]. It allows the recording of the energy consumption of a system under test (SUT), which runs the software that is to be measured. A power meter measures the energy consumption of the SUT while it runs the software. Then, a workload generator provides and controls the tasks performed by the software on the SUT in a usage scenario. During the scenario, the SUT can additionally monitor its hardware usage statistics (CPU, RAM, HDD, Network traffic, etc.). If the time of all three systems that collect data (SUT, power meter and workload generator) is synced, the data that is collected this way can easily be aggregated and evaluated by means of timestamps attached to the data. Figure 2 depicts the measurement setup.

Using this setup, we can easily measure the energy consumption and hardware utilization of software and compare two software products that perform the same task in a usage scenario. These scenarios need to be tailored to the software product to be examined (e.g. for interactive software like browsers different scenarios are needed than for non-interactive software like databases) and the questions to be answered (e.g. the comparison of two or more software products vs. the changes of one software product over time vs. measuring individual functionalities, etc.). Furthermore, if the SUT that is measured is a continuous integration server and the usage scenario consists of the unit tests of a software under development,

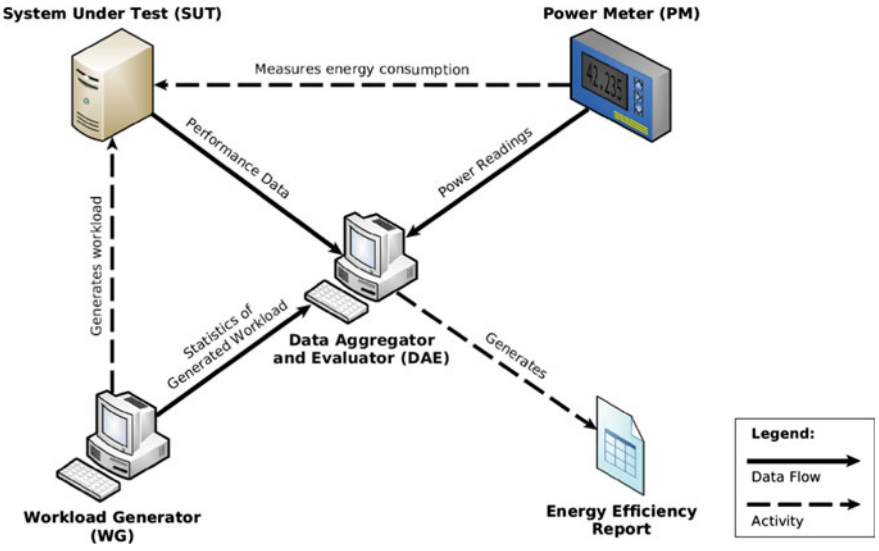

Fig. 2 Setup for measuring power usage and hardware utilization of software (based on [27])

the developers can be provided with a continuous history of the power and hardware consumption during the development phase. An approach for this idea is presented in depth in [50, 51].

To provide an idea of the measurement method and results, in the following we describe an exemplary measurement that we conducted to compare two software products, in this case word processors. The main goal of the measurement was to create a measurement that allowed us to compare the energy consumption and hardware usage of the two products. Hence, we created a usage scenario for this purpose. This scenario is supposed to emulate a user of the software product as realistically as possible. Thus, we first analyzed the tasks that are usually carried out with the products (e.g. typing and formatting text, inserting graphics, creating and updating references and a table of contents, etc.). Then, we checked these tasks against functionalities which may, according to expert opinion, induce high energy demand or high resource utilization (e.g. saving and loading, search and replace, spell-checking, etc.). Additionally, we needed to ensure that all selected actions could be executed with both software products and that the result (in this case the exported.pdf document) is identical for both products. We then measured the energy consumption and hardware utilization of the SUT while performing the scenario (duration approx. 10 min), as described above. Thus, the same work was done with each software product and we could compare the average energy consumption and hardware utilization for this scenario. To ensure that we really measured the energy consumption of the software products, we additionally switched off all possible background processes (like virus scanners, updates, indexing processes, etc.). Furthermore, we repeated the measurements 30 times and took the average to mitigate the impact of outliers. As an example, Fig. 3 shows the resulting graph of the energy measurement.

From the measured average power consumption, we calculated the consumed electrical energy (Wh). Here, it is obvious that word processor 2 consumed more

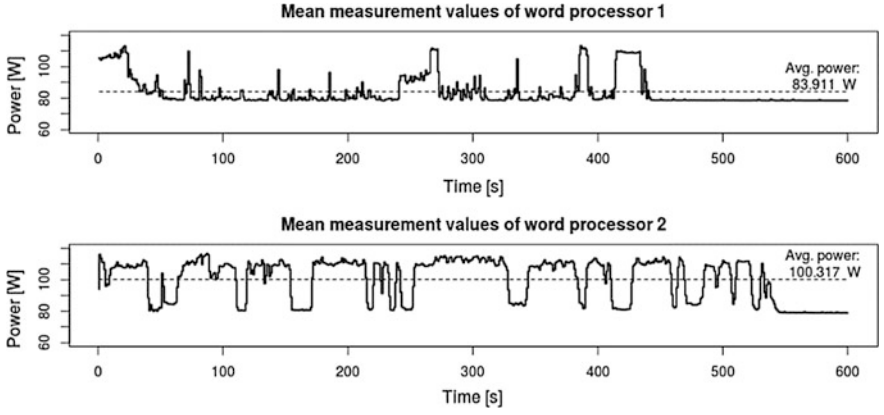

Fig. 3 Comparison of word processors: per second average of power consumption

energy, which was also confirmed by the fact that it took longer to generate the wanted.pdf and that the hardware utilization (especially CPU) during the scenario was higher as well. In this case, word processor 2 used on average 16.72 Wh and word processor 1 only 14.43 Wh.

In general, the measurement method is one possible tool in the field of green software engineering and, thus, belongs to the fourth part of our GREENSOFT Model (Fig. 1). Furthermore, the software energy measurements could create an informative basis for many stakeholders, like programmers, users, purchasers, etc. Just to give an example how the measurements can be interesting for programmers: Maevsky et al. present an experiment to show that the "power consumption increase [s] in a mobile device after its being infected by malicious software." [52] Our idea is to use the measurement results as part of a set of criteria, an eco-label for software products could rely on. Thus, this opportunity for use seems to be especially interesting for purchasers, but also for users. The following section will present the labelling approach in a more detailed way.

3.2 *Distribution and Usage: Labelling Green Software Products*

Eco-labels are supposed to provide information on the environmental friendliness of products to purchasers and users of these products. Different studies show that these approaches can be considered to be successful [53, 54]. Referring to the GREENSOFT Model, depicted in Fig. 1, eco-labels are attributed to "Sustainability Criteria and Metrics" and "Recommendations and Tools". Depending on the point of view, even the "Life Cycle of Software Products" might be relevant in the context of eco-labelling software products.

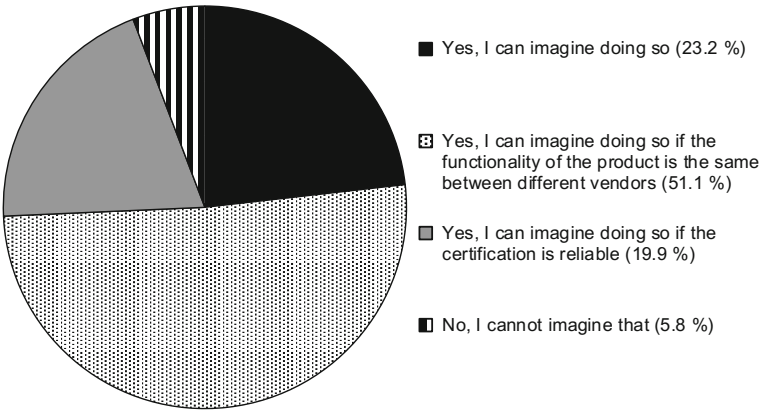

Fig. 4 Results of the question "can you imagine taking 'grading of the environmental impact of the product' as a criterion while searching for a software product?"