Environmental Issues of Software: How to Bridge from Science to Society

---

Von der Fakultät Nachhaltigkeit
der Leuphana Universität Lüneburg zur Erlangung des Grades
Doktorin der Naturwissenschaften

– Dr. rer. nat –

genehmigte Dissertation von
Eva Kern
geboren am 27. April 1988 in Neuwied

Eingereicht am:	24.4.2018
Mündliche Verteidigung (Disputation) am:	30.11.2018

Erstgutachter:	Prof. Dr. Andreas Möller
Zweitgutachter:	Prof. Dr. Stefan Naumann
Drittgutachterin:	Prof. Dr. Birgit Penzenstadler

Die einzelnen Beiträge des kumulativen Dissertationsvorhabens sind oder werden wie folgt veröffentlicht:

Beitrag 1: Kern, Eva; Dick, Markus; Naumann, Stefan; Hiller, Tim. 2014. **Impacts of software and its engineering on the carbon footprint of ICT.** In: Environmental Impact Assessment Review 52, 53–61. http://dx.doi.org/10.1016/j.eiar.2014.07.003.

Beitrag 2: Kern, Eva; Hilty, Lorenz M.; Guldner, Achim; Maksimov, Yuliyan V.; Filler, Andreas; Gröger, Jens; Naumann, Stefan. 2018. **Sustainable software products – towards assessment criteria for resource and energy efficiency.** In: Future Generation Computer Systems. https://doi.org/10.1016/j.future.2018.02.044

Beitrag 3: Kern, Eva. **Environmental issues of software & its labelling: an end user perspective** [under review for publication in: Sustainable Computing, Informatics and Systems]

Beitrag 4: Kern, Eva. 2016. **Communicating Environmental Issues of Software: Outline of an Acceptance Model**. In: Advances and New Trends in Environmental Informatics, Wohlgemuth, V., Fuchs-Kittowski, F., and Wittmann, J. (Eds.). Springer International Publishing, Cham., 335-345. http://dx.doi.org/10.1007/978-3-319-44711-7_5

Beitrag 5: Kern, Eva; Guldner, Achim; Naumann, Stefan. 2018. **Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues**. In: Kharchenko, Vyacheslav; Kondratenko, Yuriy; Kacprzyk, Janusz (Eds.): Green IT Engineering: Social, Business and Industrial Applications, Volume 3 Springer-Series "Studies in Systems, Decision and Control" https://www.springerprofessional.de/including-software-aspects-in-green-it-how-to-create-awareness-f/16162438

Veröffentlichungsjahr: 2018

Zusammenfassung

Heutzutage wird unsere (westliche) Welt von Digitalisierung geprägt. Diese wird mit Hilfe von Informations- und Kommunikationstechnologien realisiert, die erhebliche Mengen an Energie verbrauchen. In der öffentlichen Debatte werden negative Effekte der Digitalisierung auf die Umwelt mit dem Energieverbrauch von Hardware, insbesondere mit von limitierten Akkulaufzeiten mobiler Geräten, in Verbindung gebracht. Jedoch fehlt es bei vielen an Bewusstsein für Umweltwirkungen, die durch Software, als Treiber für Hardware, hervorgerufen werden, auch wenn das entsprechende Forschungsfeld wächst.

Daher geht die vorliegende Doktorarbeit der Frage nach: *Wie kann Aufmerksamkeit von (a) Entwicklern und (b) Nutzern für Umweltwirkungen von Software geschaffen werden?* Mit (a) einer Berechnungsmethode für den ökologischen Fußabdruck von Softwareprojekten und (b) einem, durch eine Nutzerumfrage evaluierten Konzept für eine Umweltkennzeichnung von Softwareprodukten werden in dieser Arbeit zwei Strategien zur Lösung beider Fragen präsentiert.

Die Doktorarbeit kann so als Basis für weitere Forschung zur Brückenbildung von Wissenschaft zur Gesellschaft im Kontext von Umweltauswirkungen von Software dienen. Die gewonnenen Erkenntnisse können Startpunkte sein für praktische Umsetzungen von Methoden und Werkzeugen, die eine umweltfreundlichere Art der Softwareentwicklung und aussagekräftige Informationen über Umweltwirkungen von Softwarenutzung ermöglichen.

Um die Umsetzung und Weiterentwicklung der Forschungsergebnisse der vorliegenden Arbeit in Gang zu bringen, werden Aufforderungen an verschiedene Stakeholder gerichtet, die als besonders relevant eingeschätzt werden und damit die Zielgruppen der Arbeit darstellen – Forschende, Zertifizierungsstellen, öffentliche Beschaffung & unternehmerisches Beschaffungswesen und Umweltverbände.

Summary

Nowadays, our (western-world) society is characterized by digitalization. This is realized by information and communication technologies, consuming a huge amount of energy. The fact that digitalization comes along with a lot of negative effects onto the environment is slightly known in the case of energy consumption by hardware, especially regarding mobile devices, having a limited battery life. However, awareness of environmental issues of software, being the driver of hardware, is mainly missing, even if the research field addressing corresponding issues is growing.

Thus, the doctoral thesis at hand addresses the question *How to draw (a) developers' and (b) users' attention to environmental issues of software?* By presenting (a) a calculation method of the carbon footprint of software projects and (b) a concept for an eco-label for software products, evaluated by a user survey, the doctoral thesis provides two strategies how to draw the attention to environmental issues of software.

Summarizing, this thesis can act as a basis for further research in bridging from science to society in the context of environmental issues of software. Its findings can be seen as starting points for practical implementations of methods and tools supporting a more environmentally friendly way of developing software and informing about environmental issues of software usage.

In order to get the implementation of the research results of the thesis going, it highlights practical implications for diverse groups of stakeholders – researchers, certifiers, public administration and professional purchasers, and environmental associations – that have been identified as being important for the practical implementation of the presented concepts and, thus, represent the target group of the doctoral thesis.

Acknowledgements – oder einfach: Danke!

Als mich Stefan vor mittlerweile acht Jahren ansprach, ob ich mir vorstellen könne in seinem Forschungsprojekt *GREENSOFT – Green Software Engineering* mitzuarbeiten, hätte ich nicht gedacht, dass diese Mitarbeit irgendwann die Form einer Dissertation annehmen wird. Deshalb gilt insbesondere Stefan ein großer Dank für die Einführung in die wissenschaftliche Welt, seine Begleitung und stete Ermutigung.

Das notwendige „Handwerkszeug" erlernte ich vor allem von und mit Markus, der mich als ehemaliger GREENSOFT-Kollege bis heute mit konstruktiver Kritik und hilfreichen Hinweisen zuverlässig unterstütze. Danke dafür! Achim möchte ich vor allem für seine Geduld und Beharrlichkeit mit meiner Form der Präpositionen und Kommasetzung im Englischen danken. Die richtigen Nachfragen an zahlreichen Stellen halfen mir, sowohl inhaltlich als auch bzgl. Grammatik und Rechtschreibung, meine Arbeit fortlaufend weiter voran zu bringen.

Meine Zeit am Umwelt-Campus, mein *UCB*-Team – Momo, Bianca, Kerstin, Miri, Sophie, Andreas, Michel, Caro usw. – und die Gewissheit, dass ich dort jederzeit willkommen bin, auch wenn ich an die Uni wechsele, gab mir Halt, Zuversicht und das nötige Selbstvertrauen nach vorne zu blicken. Danke! Eine der *Leuphana*-Doktorandin sagte mal „Warum brauchst du ein Netz [in das du fallen kannst], wenn du Wurzeln hast?!". Dem kann ich nur zustimmen: Meine akademisch-kollegial-freundschaftlichen Wurzeln halfen mir mich auf den „Weg zu den zwei Buchstaben" zu wagen.

Aber auch das in den letzten drei Jahren aufgespannte Netz in Lüneburg – ob an der Uni, in der Stadt, meinen WGs oder irgendwo dazwischen – möchte ich nicht mehr missen. Herzlichen Dank, Andreas, für deine Zustimmung zu meinem Promotionsvorhaben, ohne die die Erarbeitung dieser Doktorarbeit nicht möglich gewesen wäre. Vielen Dank für das Vertrauen in meine Arbeit, den Rückhalt und dein Feedback. Für den fortwährenden Austausch und die kritische Reflexion geht auch ein großer Dank an die *AG Fröhliche Wissenschaften II* rund um Hans-Joachim Plewig.

Ein weiterer Dank gilt der UFOPLAN-Gruppe *Sustainable Software Design* für die bereichernde Zusammenarbeit und die Möglichkeit als Doktorandin an diesem Projekt mitzuarbeiten.

Den entscheidenden Anstoß meine Doktorarbeit zu Ende zu bringen, gaben mir zuletzt die vielen gemeinsamen Stunden in der Bib mit Corinna und Franzi – gemeinsam haben wir es geschafft! Danke für Bib, Mensa, Kaffee und *mosaique*-Teilchen!

Birgit, vielen Dank für deine Zusage die Rolle der Drittgutachterin zu übernehmen und deinen motivierenden Zuspruch. Ich bin sehr froh, dass wir uns über RE4SuSy und diverse folgende Konferenzen kennenlernten.

Allem voran war es für mich sehr wertvoll, dass ich Familie und Freunde habe, die mich stets begleitet und unterstützt haben. Danke, Mama, Papa, Yvonne, Sarah, Holger und Manuel – und alle, die wohl nie so ganz verstanden, warum ich das tue, was ich tue, aber mir dabei immer zur Seite standen!

ICE 1691, den 18. April 2018

Apart from the framework paper, the individual contributions constituting this paper-based doctoral thesis have been published or will be published*:

Paper 1 (journal article, peer reviewed, published)

Kern, Eva; Dick, Markus; Naumann, Stefan; Hiller, Tim. 2014. Impacts of software and its engineering on the carbon footprint of ICT. In: Environmental Impact Assessment Review 52, 53–61. http://dx.doi.org/10.1016/j.eiar.2014.07.003.

Paper 2 (journal article, peer reviewed, accepted for publication)

Kern, Eva; Hilty, Lorenz M.; Guldner, Achim; Maksimov, Yuliyan V.; Filler, Andreas; Gröger, Jens; Naumann, Stefan. 2018. Sustainable software products – towards assessment criteria for resource and energy efficiency. In: Future Generation Computer Systems, ISSN: 0167-739X [in press]. The final publication is available at: https://www.journals.elsevier.com/future-generation-computer-systems

Paper 3 (journal article, peer reviewed, submitted)

Kern, Eva. Environmental issues of software & its labelling: an end user perspective [in review for publication in: Sustainable Computing, Informatics and Systems]

Paper 4 (book chapter, peer reviewed, published)

Kern, Eva. 2016. Communicating Environmental Issues of Software: Outline of an Acceptance Model. In: Advances and New Trends in Environmental Informatics, Wohlgemuth, V., Fuchs-Kittowski, F., and Wittmann, J. (Eds.). Springer International Publishing, Cham., 335-345. http://dx.doi.org/10.1007/978-3-319-44711-7_5

Paper 5 (book chapter, peer reviewed, accepted for publication)

Kern, Eva; Guldner, Achim; Naumann, Stefan. 2018. Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues. In: Kharchenko, Vyacheslav; Kondratenko, Yuriy; Kacprzyk, Janusz (Eds.): Green IT Engineering: Social, Business and Industrial Applications, Volume 3 Springer-Series "Studies in Systems, Decision and Control" [in press]

* Effective: 24.4.2018

The following publications are not part of the formal requirements of this doctoral thesis, but they support the argumentation.

Paper A (conference paper, peer reviewed, published)

Kern, Eva; Dick, Markus; Naumann, Stefan; Filler, Andreas. 2015. Labelling sustainable software products and websites: Ideas, Approaches, and Challenges. In: Vivian Kvist Johannsen, Stefan Jensen, Volker Wohlgemuth, Chris Preist, Elina Eriksson (Eds.): Proceedings of EnviroInfo and ICT for Sustainability 2015. 29th International Conference on Informatics for Environmental Protection (EnviroInfo 2015) and the 3rd International Conference on ICT for Sustainability (ICT4S 2015). Copenhagen, September 7 - 9, 2015. Amsterdam: Atlantis Press (Advances in Computer Science Research, 22), 82-91. doi:10.2991/ict4s-env-15.2015.10

Paper B (book chapter, peer reviewed, published)

Kern, Eva. 2017. Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges. In: Otjacques, Benoît; Hitzelberger, Patrik; Naumann, Stefan; Wohlgemuth, Volker (Eds.): From Science to Society – New Trends in Environmental Informatics, Springer International Publishing 2017, 263-273. https://link.springer.com/chapter/10.1007/978-3-319-65687-8_23

Table of Contents

List of Figures

List of Tables

Environmental Issues of Software: How to Bridge from Science to Society

– Framework paper –

1 Introduction & Motivation: Green and Sustainable Software

Our daily lives are more and more connected to information and communication technology (ICT) products: “Digitalization”, “Internet of Things”, “always online”, “digital natives”, “Big Data” are keywords describing the innovations of the last years. Overall, it is hardly possible to imagine todays’ (western-world) society without ICT. However, these developments require a lot of energy. Even if the devices themselves are more and more energy efficient [United Nation Environment Programme 2003; OECD 2015], the overall energy consumption caused by using ICT products is increasing worldwide [van Heddeghem et al. 2014; Andrae and Edler 2015; Malmodin and Lundén 2016]. Moreover, ICT is a relevant contributor to CO_2 emissions [San Murugesan 2008] on the one hand but helps to reduce these emissions on the other hand [Verdecchia et al. 2017].

While environmental impacts caused by hardware products have been known and addressed for several years by terms of “Green IT activities”, the consideration of the software side is, in comparison, more or less at the beginning [Penzenstadler et al. 2014a; Verdecchia et al. 2017]. Indeed, the idea to include software is reasonable 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.” [Hilty et al. 2015] Thus, addressing the software side of Green IT, can have a huge impact on a more sustainable everyday life [Anwar and Pfahl 2017] and refers to the energy that is used by hardware while software is running on it [Ferreira 2011].

The issue of so called “green” or rather “sustainable software” has increasingly become a research focus during the last years [Penzenstadler et al. 2012; Penzenstadler et al. 2014a; Lago et al. 2015; Verdecchia et al. 2017]. Figure 1 depicts the rising number of publications in the research fields “green computing” and “green software”. While the number of papers increases for all of the used key words, the raising numbers for papers regarding “sustainable software” are especially noticeable.

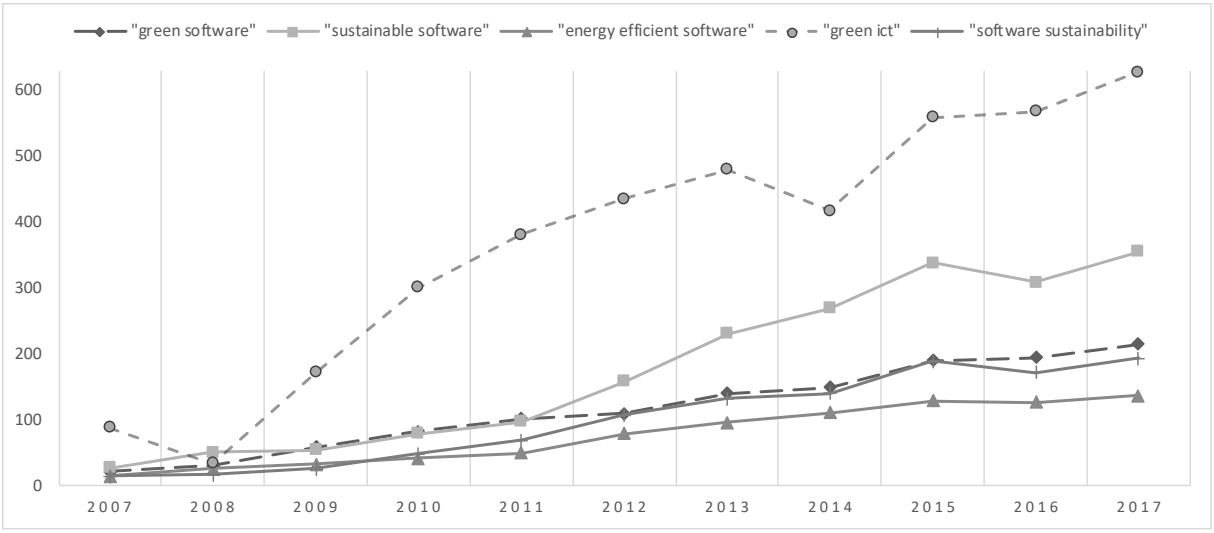

Figure 1 *Development of the number of papers published in the context of "green and sustainable software" from 2007 to 2017 based on Google Scholar results. The search on Google Scholar was conducted on April 18th, 2018 and used the key words: "green software", "sustainable software", "energy efficient software", "green ict", and "software sustainability" (always including the quotation marks) (Source: updated version of [Paper 3, Figure 1])*

However, the transfer of resulting ideas, approaches, procedure models, recommendations, etc. to practical implementations is still missing [Dookhitram et al. 2012; Souza et al. 2014; Chitchyan et al. 2016; Manotas et al. 2016; Pang et al. 2016]. Different studies show that programmers [Manotas et al. 2016; Pang et al. 2016; Groher and Weinreich 2017], students [Dookhitram et al. 2012; Selyamani and Ahmad 2015; Torre et al. 2017], IT managers [Kogelman 2011], and users [Ferreira 2011] are not or just slightly aware of green computing issues. In case of software, the awareness and knowledge are even worse than in hardware contexts. Hence, the overall motivation for this doctoral thesis is to find and prepare solutions to create and increase the awareness of environmental issues caused by using software products. In this context, the focus is set on the users', for the main part, as well as on the developers' perspective (see Subsection 3.3). Nevertheless, groups like administrators, vendors, and researchers cannot be excluded entirely since the groups overlap in many cases. Not only users might look for information on the environmental impacts of the products, also vendors do and might see unique selling points in that kind of information. This also applies for administrators: if developers lay emphasis on an environmental friendly way of production, this attitude should be kept while administrating and maintaining software products. Overall, the perspective of researchers should be kept in mind in order to attain a continuously improving process in the context of addressing environmental issues of software and the awareness of it.

Summarizing, the **aim of the doctoral thesis** is to provide strategies how to inform about environmental issues of software. In order to do so, the following section presents the background information the thesis is based on.

The framework paper for the individual contributions listed in Table 1 is structured as follows: The next Section 2 will present the field of research on green and sustainable software engineering.

Thus, it introduces definitions and provides an insight into current scientific approaches. Based on this theoretical framing, the objectives of the work, including guiding research questions, target groups and research design are described in Section 3. Section 4 presents the solutions addressing the research question of the doctoral thesis before the final Section 5 sums up the main findings, specifies recommendations towards different stakeholders and goes into key aspects and future directions of the work.

2 Theoretical framing: definitions and field of research

In order to define the driving research question, the following subsection will provide background information and give some insights into the current state of research in the context of green software and its engineering. The aim of this section is to present the scientific basis that the doctoral thesis at hand is built upon, in order to be able to bring the research done in line with existing research in the addressed field. Comprehensive discussions on the state of literature and related work can be found in the corresponding Papers 1 to 5, especially [Paper 3, Section 2].

2.1 Green and sustainable software

2.1.1 Definition and characterizations

The doctoral thesis refers to the following definition of "green and sustainable software":

> *"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." [Dick and Naumann 2010; Naumann et al. 2011]*
>
> and uses the terms "green software" and "sustainable software" in an analogous manner.

Apart from this definition by Dick and Naumann [2010], different researchers published definitions in the addressed field during the last years [Taina 2011; Bozzelli et al. 2013; Calero et al. 2013b; Penzenstadler 2013a; Ahmad et al. 2014; Betz and Caporale 2014; Penzenstadler et al. 2014b; Lago et al. 2015; Schmidt 2016]. In doing so, they set different foci:

- Dick and Naumann [2010] explicitly refer to the life cycle of a software product (see Subsection 2.1.2).
- Penzenstadler [2013a] differentiates between "sustainable software" in the sense of energy-efficiency and "software engineering for sustainability" to improve software with regard to different dimensions of sustainability in corresponding application domains.

- Penzenstadler and Femmer [2013], Razavian et al. [2014], and Lago et al. [2015] include a technical dimension of sustainability while talking about sustainable software.
- Calero et al. [2013b] talk about the "capacity of developing a software product in a sustainable manner" and understand sustainability as a non-functional requirement.
- Betz and Caporale [2014] point out that it is important to also include the underlying business processes when integrating sustainability in the context of software systems.
- Ahmad et al. [2014] see the focus on the development aspects for long living systems.

The mentioned definitions and the terminology of green and sustainable software are described, compared, and discussed in a more detailed way in [Mahaux et al. 2011; Kern et al. 2013; Calero and Piattini 2015; Hilty and Aebischer 2015; Kharchenko and Illiashenko 2016] as well as in [Paper 5]. Overall, all of them underline "(1) environmental and resource protection as well as (2) supporting sustainable development, including the three pillars of sustainability, but setting priorities" [Paper A]. Thus, even if they are to be understood in contexts of different perspectives [Penzenstadler 2013b], the motivation for the approaches on sustainable software is the same. All of them refer to sustainable development in the sense of the Brundtland Report [United Nations General Assembly 1987], even if the original three dimensions – environment, economy, and society – are, in some cases, extended by a "human" [Goodland 2002b] or an "individual" [Penzenstadler and Femmer 2013] as well as a "technical" dimension [Penzenstadler and Femmer 2012; Razavian et al. 2014; Lago et al. 2015].

Even if the focus of this doctoral thesis is set on the environmental dimension of sustainability, similarly to e.g. [Naumann et al. 2011; Bozzelli et al. 2013; Penzenstadler et al. 2014a; Lago et al. 2015], I am aware of the connection between the three or rather five dimensions. Additionally, although I chose one specific definition I am referring to, I do not want to speak out in favor of one single definition in the addressed research field. It is rather possible to evolve different foci and perspectives in that way. That might keep the field active and keep varied discussions ongoing. Nevertheless, the definition by Dick and Naumann [2010] and the focus on environmental issues of software were chosen to provide a common basis for the research done.

2.1.2 Life cycle of software

Talking about "life cycles" within software engineering, one must differentiate between a "life cycle of a software product" and a "software development life cycle". Both approaches can be seen in relation to sustainability or environmental issues. While other references talk about "software development life cycles" [Shenoy and Eeratta 2011; Mahmoud and Ahmad 2013], I refer to the life cycle of a software product in the sense of terms of life cycle assessment (LCA) and a cradle-to-grave approach [Tischner et al. 2000] as presented in [Dick and Naumann 2010; Naumann et al. 2011] and depicted in Figure 2.

Figure 2 *Life cycle of software products (simplified representation of the life cycle proposed by Naumann et al. [2011])*

Thus, the life cycle presents a product lifecycle as described in ISO/TR 14062 [Deutsches Institut für Normung e.V. 2003]. It comprises a development, distribution, usage, deactivation, and disposal phase. The idea of considering a lifecycle of software products is to assess ecological, social, human, and economic impacts during software development [Naumann et al. 2011]. Regarding creating awareness of environmental issues of software, the development and the usage phase are especially important (see Section 3.1).

An earlier approach by Göhring [2004] talks about "three levels of impacts and opportunities of ICT", namely "Effects of ICT supply", "Effects of ICT usage", and "Systemic effects of ICT". Mocigemba [2006] presents "six dimensions of the Sustainable Computing Concept" by distinguishing between focus on the product, the production process, and consumption process – each from the hardware as well as from the software level. He concentrates on social and human issues of computing. A clear focus on the software level and its impacts on sustainable development in regards to life cycle phases is set e.g. by Naumann [2008] and Abenius [2009].

In contrast to the product life cycle of software, a development life cycle approach intends to include sustainability issues in the development phase and distinguishes between the following phases: requirement, design, implementation / development (or) coding, testing, and maintenance [Afzal et al. 2013; Mohankumar and Anand 2015; Sharma et al. 2015]. However, Afzal et al. [2013] and Sharma et al. [2015] mainly reduce sustainability to the sense of energy efficiency. Regarding how to integrate energy efficiency issues into the proposed process, Sharma et al. [2015] propose procedures for each of the phases. For example, they recommend studying the requirements in terms of energy efficiency and validating a green design for the product and manufacturing process. Overall, they suggest that "all the [sustainable development life cycle] stages are similarly connected to the green analysis phase in a cyclic manner". Thus, after completing one of the processes, the green analyses proves if the completion is done in the sense of green computing. Mohankumar and Anand [2015] do not expand on how to integrate sustainability issues.

2.2 Green Software Engineering: field of research

ICT Sustainability and Green IT in general cover, as summarized by Lami and Buglione [2012] and Agarwal et al. [2012], hardware energy efficiency, optimization of algorithms, architecture as well as services, and green software engineering. Within my doctoral thesis, I focus on the latter. An analysis of the research on green software engineering in 2015/16 identified the following sub-domains: 1) sustainable mobile applications, 2) sustainable software design and development, 3) energy-aware

resource scheduling/management, 4) green computing in networks, 5) sustainable requirements engineering, 6) green in big data, cloud and data centers, 7) software energy consumption, 8) other [Anwar and Pfahl 2017]. Both classifications give an insight into topics of the research field. More details of the research within the last years can be found in corresponding literature reviews or similar descriptions of the research field, e.g. [Calero et al. 2013a; Penzenstadler et al. 2014a; Anwar and Pfahl 2017; Verdecchia et al. 2017]. In order to get an overview of the field, Verdecchia et al. [2017] mapped current research activities to the conceptual framework of ICT effects presented by Hilty et al. [Hilty 2008; Hilty and Aebischer 2015], means to direct effects, effects of use, and systematic effects. In addition, they propose to include the category of "people awareness effects".

All of the research studies underline that the addressed field – green ICT and especially the focus on software – is a relatively young research field [Penzenstadler et al. 2014a; Verdecchia et al. 2017]. Even if studies of the field were just published in 2003 onwards [Calero et al. 2013a; Verdecchia et al. 2017], the topic "has received wide-spread attention in the software engineering community" [Penzenstadler et al. 2014a]. However, according to Penzenstadler et al. [2014a], "there are many different roads being explored but there are no methods and models yet that can be considered as established for [Software Engineering for Sustainability (SE4S)]." According to Anwar and Pfahl [2017], most of the research studies done in 2015/16, present a method (64 % of the analyzed papers). These findings go along with an earlier literature study by Penzenstadler et al. [2014a]. Both studies identified the "validation" as a research gap, i.e. the implementation and testing of research approaches in the industry.

Bridging from industry and practice to research activities and vice versa and, thus, developing metrics, "explaining the correlation between energy usage and other quality attributes" [Anwar and Pfahl 2017], could "help in strengthening or correcting the assumptions currently used by software practitioners" [Anwar and Pfahl 2017]. This, as well as the assumption that "research focuses on non-functional requirements like performance and efficiency, where attributes like sustainable and energy efficiency are ignored" [Anwar and Pfahl 2017], and the lack of addressing people's awareness [Verdecchia et al. 2017], underlines the relevance of the research done within the doctoral thesis at hand.

2.3 Awareness of Green Software

Referring to: Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges [Paper B]

Kern, Eva (2017): Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges. In: Otjacques, Benoît; Hitzelberger, Patrik; Naumann, Stefan; Wohlgemuth, Volker (Hrsg.). From Science to Society: New Trends in Environmental Informatics. Springer International Publishing. pp. 263-273, online: https://link.springer.com/chapter/10.1007/978-3-319-65687-8_23

***Abstract**. Issues of green computing, especially on the hardware side, arouse interest of scientific communities since several years. The number of scientific publications and research activities rises. This begs the question if resulting ideas, approaches, and findings find their way from science to society. Hence, similar to different researchers, we conducted a survey regarding corresponding issues. Since we focused on software users, we will compare our results to those addressing students, practitioners, and IT managers. The following paper will provide an insight into awareness of green software and green computing, present a user survey we conducted in the summer of 2016 in Germany, and approaches on how to create awareness to environmental issues of ICT, especially on the software side.*

Within my doctoral thesis, I especially refer to people's awareness as introduced by Lago and Jansen [2010]. Lago, Jansen refer to service-based applications and explain the "people awareness" as "the way users exploit [software as-a-service], by making them realize their energy consumption" [Lago and Jansen 2010]. This interpretation seems to be transferrable to software in general and can be increased by the realization of people's impact on sustainable development while using software. This kind of awareness stands in contrast to process and service awareness[1]. Lago and Jansen [2010] propose to offer green metrics to gather the greenness of services and provide feedback on it and present a corresponding model. Another awareness model is suggested by Jagroep et al. [2017]. Their "awareness model for software energy consumption" is intended to "capture the awareness of stakeholders involved with product development." [Jagroep et al. 2017]

Other studies analyze the awareness of green computing [Kogelman 2011; Dookhitram et al. 2012; Selyamani and Ahmad 2015] or rather green software [Ferreira 2011; Manotas et al. 2016; Pang et al. 2016; Groher and Weinreich 2017; Torre et al. 2017]. In my conference paper "Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges" [Paper B], I address the questions (i) Is there an awareness towards green computing and especially green software? and (ii) How can the awareness towards these topics be raised and/or increased?

Reviewing existing research in this context led to the conclusion that knowledge about green computing of the interviewed programmers, students, practitioners, buyers, suppliers, and managers is given [Ferreira 2011; Kogelman 2011; Pang et al. 2016; Torre et al. 2017]. However, even if they have

[1] Process awareness: "convince decision makers of the urgency of adopting ecological strategies in the processes of managing their IT portfolios and bring IT practitioners to adopt innovative ecological models in the way they engineer SBAs, i.e. in the development process"; service awareness: "define and implement how ecological strategies can be adopted by service-oriented software" (Lago and Jansen [2010])

some knowledge in the context of Green IT, they do not know how to integrate that into their daily routine [Dookhitram et al. 2012; Chitchyan et al. 2016; Pang et al. 2016]. Since studies addressing the group of software users are mostly missing, I addressed this research gap. The user survey done within the doctoral studies is presented in Section 4.2.3 and, in more detail, in Paper 3.

3 Objectives of the doctoral thesis

This doctoral thesis aims at presenting approaches on how to draw attention to environmental issues of software. In order to specify this general objective, the following section introduces the guiding research questions, provides an overview of the relevant papers of the cumulative thesis, describes the target groups and presents the overall research design of the work.

3.1 Guiding research questions

Driven by the thesis that users can make a difference in the resulting environmental impacts caused by software if they are aware of the topic and their possibilities to change something [Ferreira 2011], the thesis focuses, as described above, on the users' perspective. Here, the term "users" refers to those who are using software products on desktop PCs, laptops, smartphones or similar end-user devices in private and professional contexts. They do not need skills in developing, maintaining or administrating software products. The term "users" stands in contrast to developers, administrators, IT managers, purchasers, distributors, and vendors. However, the members of the groups overlap (see Section 1 and Paper 2).

Besides users, the awareness of software developers may also have a large impact within the question on how to counteract the increasing environmental effects of ICT, especially software. They are the ones who have the possibilities and knowledge to influence the resulting impacts of software usage in an early phase of the software development life cycle [Naumann et al. 2011]. Thus, the doctoral thesis goes slightly into the developer's possibilities, as well.

The focus of the doctoral thesis is set on the users' as well as the developers' perspective for several reasons: First, the development and the usage phase do have a significant impact onto environmental effects caused by software [Jagroep et al. 2017; Maevsky et al. 2017]. While the development phase predefines the main characteristics of a software product, using the product directly influences the environment by consuming (natural) resources and energy. Second, in most of the cases, having standard software in mind, users can be change makers in the way of using software or in setting requirements for products that are sold. Besides, meeting the resulting demand for products being more environmentally friendly, software developers are needed. Third, aiming at spreading the idea of green and sustainable software over its whole life cycle, it is necessary to create transparency in this context. This can be reached if a large group is informed. Having the different

stakeholders of (software) sustainability in mind [Penzenstadler et al. 2013; Herzog et al. 2015], users represent the biggest group and just slightly overlap with academia. In that way, it seems to be possible to bridge from science to society. Johann and Maalej [2015] point out "that the inclusion of users and their communities in the engineering processes has a high potential to support sustainable software engineering". Lami et al. [2013] also understand the user as one resource that should be taken into consideration while dealing with the sustainability of the whole software process.

Consequently, the resulting main two-part research question for the doctoral thesis at hand is:

a) How to draw *developers'* attention to environmental issues of software?

b) How to draw *users'* attention to environmental issues of software?

While researching possibilities on how to create awareness of environmental issues of software, two approaches stand out from the collection of ideas: calculating the carbon footprint of software over the different life cycle phases and creating an eco-label for software products to inform about its environmental impacts. These two approaches could be mapped to the groups addressed in this doctoral thesis:

a) *Developers* could be informed about environmental issues of software by calculating the carbon footprint of the development of software products.
b) *Users* could be informed about environmental issues of software by providing an eco-label for software products.

Following these two approaches, further research questions arise:

1. Carbon footprint

RQ1: How can the carbon footprint of the development of software products be calculated?

The aim is to suggest a method to evaluate the development phase of software products from the perspective of environmental issues. The method should be easy to integrate in existing development processes and easy to learn.

2. Eco-label

RQ2: What are criteria for green software products, upon which an eco-label could be based?

The aim is to go one step further in the direction of a collective understanding of green and sustainable software. Additionally, this will move forward the creation of an information medium on green software aiming to notify especially software users about environmental issues of software and its usage.

RQ3: Which aspects of green software products raise the highest interest of users and should be labelled?

The aim is to find out the users' interest in environmental issues of software. Thus, the eco-label should be based upon criteria that (i) have a high relevance from the perspective of environmental impacts and (ii) raise a high interest of those who should be aware of it, meaning especially software users.

RQ4: What are aspects influencing the user acceptance of an eco-label for software products?

The aim is to find out aspects that should be considered while developing an eco-label for software products in order to reach a high user acceptance for the resulting label.

3.2 Overview of relevant papers and application to research questions

As already mentioned, the focus is set on the view of users. Hence, the idea of labelling green software products will be researched in more detail compared to calculating methods for the carbon footprint of software development. The overall focus of this doctoral thesis concentrates on finding possibilities to bring the research topic of green software to those developing and using software products. The different research papers, as part of the cumulative doctoral thesis, contribute to answer the four individual questions RQ1 to RQ4. The combined findings of these papers will address the main research question (see Table 1).

Table 1 *Combination of the research questions RQ1 to RQ4 to answer the guiding main research question*

How to draw *developers'* attention to environmental issues of software?			
#1	[RQ1] How can the carbon footprint of the development of software products be calculated?	[Paper] Kern, Eva; Dick, Markus, Naumann, Stefan; Hiller, Tim (2014): Impacts of software and its engineering on the carbon footprint of ICT. In: Environmental Impact Assessment Review [*published*]	[Section] 4.1 Addressing software developers: carbon footprint of software development
How to draw *users'* attention to environmental issues of software?			
#2	[RQ2] What are criteria for green software products, upon which an eco-label could be based?	[Paper] Kern, Eva; Hilty, Lorenz M.; Guldner, Achim; Maksimov, Yuliyan V.; Filler, Andreas; Gröger, Jens; Naumann, Stefan (2018):	[Section] 4.2.2 Criteria: how to find a basis for labelling green software products

		Sustainable software products – towards assessment criteria for resource and energy efficiency [*in press*]	
#3	[RQ3] Which aspects of green software products raise the highest interest of users and should be labelled?	[Paper] Kern, Eva (tbd): Environmental issues of software & its labelling: an end user perspective. Considered for publication in: Sustainable Computing, Informatics and Systems [*submitted*]	[Section] 4.2.3 Evaluation: user's interest in environmental issues of software
#4	[RQ4] What are aspects influencing the user acceptance of an eco-label for software products?	[Paper] Kern, Eva (2016): Communicating Environmental Issues of Software: Outline of an Acceptance Model. In: Advances and New Trends in Environmental Informatics [*published*]	[Section] 4.2.4 Evaluation: aspects influencing the acceptance of an eco-label for software
Overview of the work done			
#5	[RQ] referring to RQ1 to RQ4	[Paper] Kern, Eva; Guldner, Achim; Naumann, Stefan (2018): Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues. In: Kharchenko, Vyacheslav; Kondratenko, Yuriy; Kacprzyk, Janusz (Eds.): Green IT Engineering: Social, Business and Industrial Applications, Volume 3 Springer-Series "Studies in Systems, Decision and Control" [*in press*]	[Section] referring to the whole frame paper

3.3 Target groups of the doctoral thesis

While users and developers are in the focus of the work for the doctoral thesis at hand (see Subsection 3.1), the target groups of this thesis are those who can push the proposed strategies to reach software developers and users.

Penzenstadler et al. [2013] present a generic list of stakeholders who are important for successfully implementing sustainability issues. The structure of the list follows the five dimensions of sustainability: individual, technical, social, environmental, economic [Penzenstadler and Femmer 2013]. In the context of the doctoral thesis, especially the stakeholders of the environmental dimension seem to be relevant. These are [Penzenstadler et al. 2013]:

- "**Legislation** (state authority): Environment protection laws are in place to ensure sustainability goals. These laws must be reflected in the model.
- The **CSR manager** is often also responsible for environmental aspects.
- Nature conservation **activists** and lobbyists (e.g., WWF, Greenpeace, BUND)"

All of them support activities that are supposed to create more environmentally friendly ways of life in different contexts. Thus, they are asked to take the findings of the thesis at hand and integrate them into their already existing activities. I will outline the specific implementations for each of them in Subsection 5.2.

Herzog et al. [2015] present "Actors for Innovation in Green IT". They distinguish between "Actors Developing Innovation in Green IT" and "Actors Supporting Innovation for Green IT". In the context of this doctoral thesis, the first group seems to be more important. They consist of [Herzog et al. 2015]:

- **Standardization bodies**, who "take material and tools, generated by industry alliances and academia and bring them to standards to be used by governments […] as regulations in laws that must (and can) be enforced."
- **Influential groups**, who **"**propose to enforce and influence [the] development of Green IT by addressing the issue at various levels."
- **Universities and academic institutes**, who "include the groups involved in the development of Green IT through academic research."
- **Company members**, who "share a common purpose, organizing their resources and skills to achieve a well-defined goal."

Bringing these two approaches of lists of stakeholders in sustainability, or rather, Green IT together, it turns out that the groups have interests in common and overlap (see Figure 3). Additionally, they can support each other to reach the shared aim of creating a more environmentally friendly, or rather, sustainable way of live. Figure 3 extracts the individual persons from the lists proposed by Penzenstadler et al. [2013] and Herzog et al. [2015] in order to gather the target groups for this doctoral thesis. The stakeholders of the social and economic dimension [Penzenstadler et al. 2013] as well as actors supporting innovation for Green IT [Herzog et al. 2015] are not mapped within Figure 3 caused by their less relevance for the addressed issue of the doctoral thesis.

Summarizing, the target group of the thesis comprises **researchers**, **certifiers**, **environmental associations** and **public administrations**, as well as **professional purchasers**. Thus, they are understood as the subjects or the active part, which can further drive the issue of creating awareness on environmental issues of software on the basis of the findings of the doctoral thesis forward. Apart from that, there are developers and users, being the objectives or the passive part of the doctoral studies, since they are important to be addressed to reach a more environmentally friendly usage behavior while developing and using software.

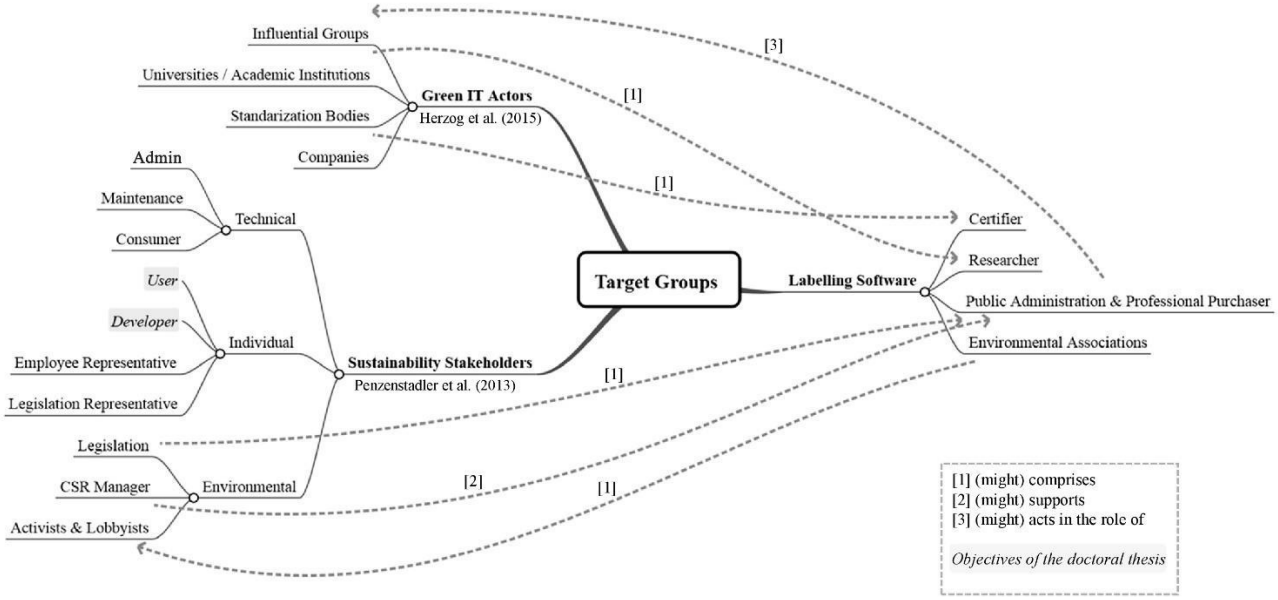

Figure 3 *Identifying the target groups for labelling environmental issues of software based on the approaches by Penzenstadler et al. [2013] and Herzog et al. [2015]*

Researchers (regarding of the objective "developers") and certifiers (regarding the objective "user") generate a special interest. Researchers can create guidelines and tools to further support developers in creating sustainable software, when considering the findings of this thesis. Here, mainly researchers in the field of environmental and sustainability informatics, who are aware of, or rather, have an open mind to transdisciplinary research [Mobjörk 2010; Lang et al. 2012], seem to be interesting to address.

Certifiers are experts in developing program guidelines for labels. They can bring the findings of the thesis towards an eco-label for software and thus create an information tool for users. Hence, they can support the transfer of the research results towards society. It seems to be sensible if they work together with environmental associations, who are the experts in informing consumers about environmental issues. Even if users are a big group in purchasing software, the interests of the group are diverse and thus, it is much harder to find strategies how to address them. Thus, another group who could be helpful in the transformation of the findings of the doctoral thesis towards an application in practice comprises public administrations and/or professional purchasers. Here, the procurement is in

the responsibility of a manageable number of persons, who are in charge of purchasing big amounts of software products. Often, they follow procurement guidelines. The idea is that this group takes the label into account while they are searching for new software products. The effect might be even bigger if the sustainability criteria for software products find its way into the procurement guidelines, than it could have in case of private purchasers. In that way, the more-regulated procurement processes can represent an example for private purchasing processes. While integrating the guidelines in bigger procurement contexts, they can be optimized and thus arranged for the private market.

3.4 Research design

For the doctoral thesis at hand I chose an exploratory research design. While preparing the thesis and addressing the research questions, presented in Section 3.1, I used different methods [Easterbrook et al. 2008]:

Preparation: Motivation and state of the research

As a first step, before starting the proper research regarding the defined questions, the motivation for the research and the current state of the research had to be analyzed. To do so, I used a meta-analysis. Some of the findings of this analysis can be found in the conference paper "Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges" [Paper B]. Additionally, the results influenced the presentation of "Related Work" [Paper 1, 4], "Literature" [Paper 2], or rather description of the "Background" [Paper 3, 5] in the papers for the cumulative thesis.

RQ1: Calculating the carbon footprint of software development

The development of the calculation method for a carbon footprint of a software development process is based on existing literature, approaches, and guidelines in the context of green software and carbon footprinting [International Organization for Standardization; Taina 2010; British Standards Institution 2011; Williams 2013]. Thus, these approaches and guidelines have been analyzed and, if necessary, transferred to the use case "software". The calculation method has been applied by using statistical data. As a result, an exemplary software company served as a basis for a theoretical case study to evaluate the calculation method.

RQ2: Criteria for green software products

In order to find criteria, my co-authors and I[2] used a combined approach of a literature review, including scientific publications as well as awarding programs for environmental labels. We

[2] I will use the personal pronoun "we" from now onwards when referring to work that has been done together with my co-authors; talking about parts that I researched or published as single author, I will use "I".

analyzed the resulting collection of possible criteria, evaluated and structured the criteria, and presented a set of criteria in a hierarchical order in the end. Then, the application of the set of criteria has been tested in order to show how it can be used to compare software products with similar functionality in terms of environmental issues.

RQ3: Labelling green software products: user's view

Besides demonstrating the application of the set of criteria by implementing corresponding verification methods, the users' perspective, being one of the future target groups for an eco-label for software products, aroused interest. Thus, the criteria ideas have been evaluated by conducting an online user survey, addressing especially those using software (in contrast to developing, administrating, or purchasing software).

RQ4: Aspects influencing the user acceptance of an eco-label for software products

On the one hand, the user survey addressed the criteria ideas for labelling environmental issues of software. On the other hand, the survey considered aspects that might influence the user acceptance of an eco-label for software products. In order to gather these aspects, I used a modeling approach, based on the Technology Acceptance Model 2, published by Venkatesh and Davis [2000].

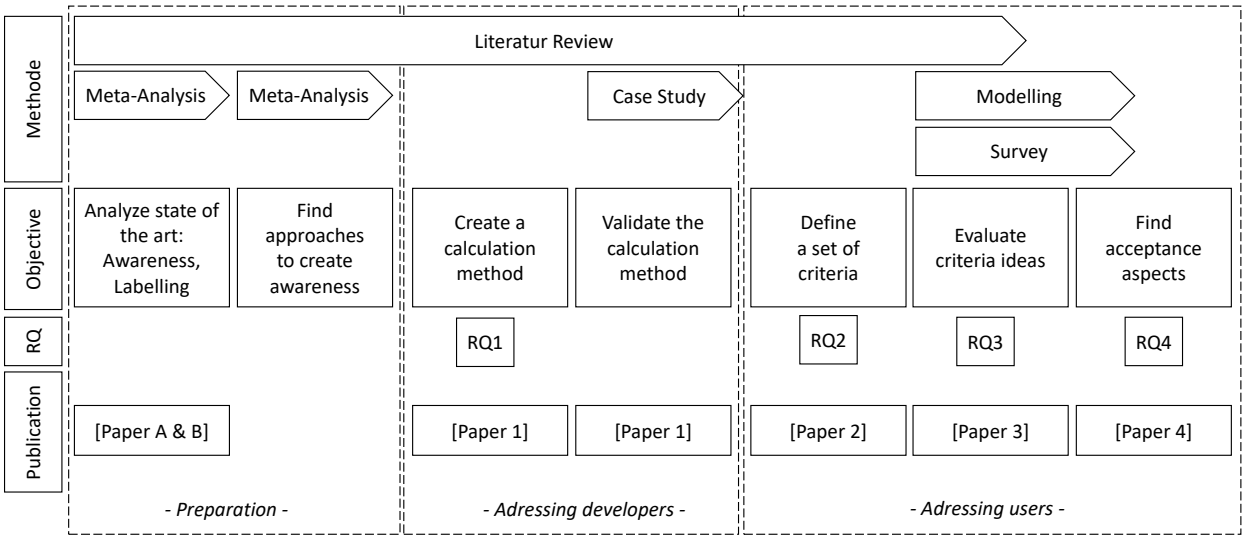

Figure 4 *Methods, objectives, and deliverables of the doctoral thesis referring to the research questions and the two-part structure of the research done*

Summarizing, as depicted in Figure 4, the combination of different research methods lead to the results that will be presented within the next two sections.

4 Solutions: Concepts on informing developers and users about green software issues

Referring to the two approaches presented in context with the guiding research questions (Subsection 3.1), the following section presents both ideas in a more detailed way: how to calculate the carbon footprint of software development as well as how to create an eco-label for software. Certainly, since the focus is set on the user's perspective (see Section 3), the research dwells on the idea of eco-labelling software products. Nevertheless, the work will present answers to both parts of the guiding research question "How to draw developers' and users' attention to environmental issues of software?", as presented in Table 1.

4.1 Addressing software developers: carbon footprint of software development

Referring to: Impacts of software and its engineering on the carbon footprint of ICT [Paper 1]

Kern, Eva, Dick, Markus, Naumann, Stefan, and Hiller, Tim. 2014. Impacts of software and its engineering on the carbon footprint of ICT. Environmental Impact Assessment Review 52, 53–61. http://dx.doi.org/10.1016/j.eiar.2014.07.003.

***Abstract**. The energy consumption of information and communication technology (ICT) is still increasing. Even though several solutions regarding the hardware side of Green IT exist, the software contribution to Green IT is not well investigated. The carbon footprint is one way to rate the environmental impacts of ICT. In order to get an impression of the induced CO_2 emissions of software, we will present a calculation method for the carbon footprint of a software product over its life cycle. We also offer an approach on how to integrate some aspects of carbon footprint calculation into software development processes and discuss impacts and tools regarding this calculation method. We thus show the relevance of energy measurements and the attention to impacts on the carbon footprint by software within Green Software Engineering.*

In order to catch software developers' attention to environmental issues of the software products under development, they need a) to know about the impacts of the overall process and b) tools to be used during the development process to integrate the consideration of these issues during the process. Thus, the following section presents a calculation method for the carbon footprint of a software development phase. The carbon footprint is one possibility to get an impression of environmental impacts and can be calculated during the software development process.

Generally, the "carbon footprint" can be defined as follows:

> *"The carbon footprint is a measure of the exclusive total amount of carbon dioxide emissions that is directly and indirectly caused by an activity or is accumulated over the life stages of a product." [Wiedmann and Minx 2007]*

In terms of software development, the activities causing CO_2 emissions include everything that is connected to the development of a specific software product. Within our theoretical case study,

presented in Paper 1, we considered the following factors: employees, office space, and ICT infrastructure (see Table 2).

Overall, the presented method of calculating the carbon footprint of software development is guided by the ideas of Taina [2010], guidelines of International Standard ISO/DIS 14067.2 [International Organization for Standardization], Publicly Available Specification PAS 2050:2011 [British Standards Institution 2011], and the Green House Gas (GHG) Protocol [Williams 2013].

Table 2 *Factors considered for the carbon footprint calculation [Paper 1, Table 1]*

Factor	**Issue**
Employees	Commuting
Office space	Heating, ventilation, and air conditioning (HVAC)
ICT infrastructure	PCs, workstations
	Backup storage system
	Rack mount servers
	Server administration
	Uninterruptible power supply (UPS)

The method of calculating the carbon footprint by focusing on the development phase comprises the following steps:

1. Defining the factors to be considered in the carbon footprint calculation (see Table 2)
2. Acquisition of data relevant for calculation
3. Assigning the data to the factors considered in the calculation
4. Calculating the emissions per functional unit by adducing average emission rations (to be found in corresponding literature and statistics, depending on the localization of the developing enterprise)
5. Summing-up the emissions of the different factors to get the overall CO_2 emissions

Within Paper 1 of the cumulative doctoral thesis, we presented an approach on how to calculate the carbon footprint of the development of software by defining an exemplary company. The focus was set on the development phase but also mentioned the usage phase since “the more a software is installed after development, the less the development phase is taken into account” [Paper 1]. We chose kg CO_2 equivalents (CO_2e) per person month as the functional unit.

Calculating the carbon footprint of our example company [Paper 1], we found especially the following impacts on the resulting carbon footprint: impacts from commuting, from operating, and from the used resources. The last point refers to “(1) the amount of energy used, (2) the kind of energy used, and (3) the time of consuming energy.” [Paper 1] Improving these aspects can be instrumental in

improving the overall resulting carbon footprint. Generally, it is important that the same conditions are ensured when comparing the carbon footprint of the development of different software products. Thus, it is, for example, not possible to include the emissions for commuting just in one case.

Besides impacts on the results of the methods, we touch on challenges during the calculation process. The biggest challenge we see is the collection of data during the software development process. Thus, we propose (i) to integrate continuous energy efficiency measurements (cf. [Drangmeister et al. 2013], [Paper 1]) in order to monitor the energy consumption during the software development phase and (ii) to use checklists and standard forms to gather the characteristics of the company as well as project relevant data.

Summarizing, calculating the carbon footprint is "one possibility to get an impression of the emissions and to create more transparency and awareness in this context" [Paper 1]. In addition to informing about the environmental impacts of software development in form of CO_2 emissions, the calculation makes it possible to identify optimization potentials. Generally, the proposed method can be conducted by software developers themselves or product managers. Thus, the method "allows software development teams to take over the responsibility for some decisions regarding the development process itself (necessity of meetings, business trips), as well as design decisions that have a direct impact on the environmental hardware requirements and also on some social impacts of their software products. That means, based on the current carbon footprint of the software, the software team can decide if e.g. a business trip is necessary or if it would be better to try to exchange ideas during a conference call." [Paper 1]

Overall, Paper 1 directly addresses RQ1 "How can the carbon footprint of the development of software products be calculated?" Referring to the guiding research question (How to draw developers' attention to environmental issues of software?), it seems to be especially important that (i) it is possible to use the method during the development process and thus continuously observe the resulting carbon footprint to be able to optimize influencing aspects, (ii) the software developers or product managers are able to use the method without huge efforts, and (iii) everyone being part of the developing company can be trained, so that the implementation of the calculation method of the carbon footprint of software projects as well as the results can improve over time when using the method regularly.

4.2 Addressing software users: eco-label for software products

The following section presents the idea of labelling green software products (Subsection 4.2.1) and possible criteria, upon which an awarding guideline could be based (Subsection 4.2.2). By means of a user survey, these ideas have been evaluated. The results are shortly described in Subsection 4.2.3 and

in a more detailed way in Paper 3. Additionally, Subsection 4.2.4 introduces aspects that might influence the acceptance of a future eco-label for software products.

4.2.1 Introduction: labelling green software products

By creating an eco-label for software products, software users should be informed about environmental issues. We assume that the information process leads to a more environmentally friendly usage of ICT and especially software products. This could reduce the negative environmental impacts in this area.

The approach of labelling products is not new. Generally, eco-labels are defined as follows:

> *"Environmental labelling provides an indication of the environmental impact-related characteristics of a product, typically on the package containing the product, by private or public institutions." [United Nations 1997]*

Within my doctoral thesis, I refer to ISO Type I environmental labels [International Organization for Standardization 1999]:

> *"A voluntary, multiple-criteria based, third party programme that awards a license authorizing the use of indicating overall environmental preferability of a product within a particular product category based on life cycle considerations." [Sustainable Business Associates 2006]*

Researching a label for green software lead to just marginal findings: there are some approaches of labels and similar tools in international contexts (e.g. Greenalytics [Zapico et al. 2010], Green Tracker [Amsel and Tomlinson 2010]), especially in the field of a so called "green web" (e.g. Web Energy Archive[3], Green Web Foundation[4]). Indeed, an eco-label based upon standardized awarding criteria, especially on a national level, is missing.

Labelling green software products seems to be sensible for several reasons: First, information on sustainability relevant aspects regarding software are spread. Second, the transparency in this context can be supported and thus it might be possible to transfer an issue of academia to the public. Third, the approach is successfully proven for other kinds of products [Lago and Jansen 2010; Rahbar and Abdul Wahid 2011; Atkinson and Rosenthal 2014].

The first step of conducting research on labelling green software products was to analyze which aspects need to be considered during the label development process. Our findings are summed up in the paper "Labelling sustainable software products and websites: Ideas, Approaches, and Challenges" [Paper A]. In our point of view, it is important (i) to have a common definition to which the label

[3] http://www.webenergyarchive.com
[4] https://www.thegreenwebfoundation.org

refers, (ii) to identify corresponding criteria, (iii) to decide on a suitable form of representation, (iv) to be aware of the proposed target groups of the label, as well as (v) the stakeholders who can support the complete labelling process. The first two aspects are the most important ones to create a consistent eco-label for software products. Our research activities are based on the definition for green and sustainable software products by Dick and Naumann [2010] presented in Subsection 2.1.1. Criteria, upon which the label could be based, are presented in the following Subsection 4.2.2 or in Paper 2 in a more detailed way.

Overall, the label itself is defined by the following three points [Paper 5]:

1. The label should be awarded to environmentally friendly products, i.e. the underlying criteria should evaluate environmental impacts of the product but not include social or economic aspects.
2. It should be awarded to the product itself, i.e. without the development and distribution process, the surroundings, and the developing organization.
3. It should be awarded because of "green in software" aspects. This means, the product itself should be as environmentally friendly as possible. Products that support environmentally friendly processes, or rather make them more environmentally friendly ("green by software") but are not green themselves should not be included.

These points set the framework for the research that will be presented in the next subsections. Thus, when talking about "sustainable" software products, I refer to the "environmental sustainability" [Goodland 2002b]. The terms "green software" and "sustainable software" stand for the same throughout this thesis.

4.2.2 Criteria: how to find a basis for labelling green software products

Referring to: Sustainable software products – towards assessment criteria for resource and energy efficiency [Paper 2]

Kern, Eva; Hilty, Lorenz M.; Guldner, Achim; Maksimov, Yuliyan V.; Filler, Andreas; Gröger, Jens; Naumann, Stefan. Sustainable software products – towards assessment criteria for resource and energy efficiency, Future Generation Computer Systems [in press]

***Abstract**. Many authors have proposed criteria to assess the "environmental friendliness" or "sustainability" of software products. However, a causal model that links observable properties of a software product to conditions of it being green or (more general) sustainable is still missing. Such a causal model is necessary because software products are intangible goods and, as such, only have indirect effects on the physical world. In particular, software products are not subject to any wear and tear, they can be copied without great effort, and generate no waste or emissions when being disposed of. Viewed in isolation, software seems to be a perfectly sustainable type of product. In real life, however, software products with the same or similar functionality can differ substantially in the burden they place on natural resources, especially if the sequence of released versions and resulting hardware obsolescence is taken into account. In this article, we present a model describing the causal chains from software products to their impacts on natural resources, including energy sources, from a life-cycle perspective. We focus on (i) the demands of software for hardware capacities (local, remote, and in the connecting network) and the resulting*

hardware energy demand, (ii) the expectations of users regarding such demands and how these affects hardware operating life, and (iii) the autonomy of users in managing their software use with regard to resource efficiency. We propose a hierarchical set of criteria and indicators to assess these impacts. We demonstrate the application of this criteria set, including the definition of standard usage scenarios for chosen categories of software products. We further discuss the practicability of this type of assessment, its acceptability for several stakeholders and potential consequences for the eco-labelling of software products and sustainable software design.

In order to award a label for environmental friendliness to any product, it is necessary to have awarding criteria. This corresponds to the question what characteristics of green or sustainable software products actually are, based on the definition introduced above. According to Anwar and Pfahl [2017], the non-functional requirement "sustainable" for software products has been mainly ignored by research so far. Hence, we researched literature dealing with definitions, criteria, aspects, and characteristics of green, or rather sustainable software products. The collection process of possible criteria for green and sustainable software comprised research papers as well as program requirements of eco-labels awarding ICT hardware products, e.g. desktop PC and notebooks [EPA ENERGY STAR 2014; RAL gGmbH 2014; TCO Development 2015]. In the latter case, we analyzed if it is possible to transfer the resulting hardware criteria to software products [Paper 5]. The findings have been complemented by our own ideas. Thus, in order to develop the set of criteria we used a combined approach including literature, existing software quality criteria, and software-specific criteria from general sustainability or rather environmental indicators. The resulting collection was sorted as well as checked against duplicates and criteria that might be named differently but have the same meaning. Additionally, the results were checked against the following aspects [Paper 2]: (i) the criterion is observable in the use phase of the software product, (ii) the criterion refers to environmental impacts which are resulting from the operation of a software product, and (iii) the criterion belongs to the characteristic of the product itself. This corresponds with the general set-up for the eco-label, presented in Subsection 4.2.1. All of the criteria included in the final set of criteria possess a high relevance, measurability, feasibility, and discrimination, as described in [Paper 5], based on [Mazijn et al. 2004]. In total, we proposed 25 criteria[5].

The next step was to structure the collection of criteria in a hierarchical order, so that the aspects can be handled in a better way. This resulted in three groups of criteria: issues related to resource efficiency, to potential hardware operating life, and addressing user autonomy (Table 3). On level 2 of the set of criteria, the aspects are differentiated in a more detailed way. On each level, the proposed criterion is clarified by asking a corresponding question. The question gives an impression how to measure or rather demonstrate if the criterion is fulfilled. It is followed by indicators that allow the evaluation of the products regarding sustainability.

[5] Find the set of criteria online on http://green-software-engineering.de/criteria-catalog and in the annex of the framework paper.

Table 3 *Hierarchical order of the criteria within the set of criteria*
(simplified representation of the hierarchical order of the set of criteria in [Paper 2, Table 3])

Level 1	**Level 2**
1 Resource efficiency	1.1 Hardware efficiency 1.2 Energy efficiency 1.3 Resource management
2 Potential hardware operating life	2.1 Backward compatibility 2.2 Platform independence and portability 2.3 Hardware sufficiency
3 User autonomy	3.1 Transparency and interoperability 3.2 Uninstallability 3.3 Maintenance functions 3.4 Independence of outside resources 3.5 Quality of product information

Just to give one example the criterion "backward compatibility" is presented in Table 4:

Table 4 *Description of an exemplary criterion for sustainable software product like done in the set of criteria, presented in Paper 2*

Criterion	Backward compatibility
Belonging to	2. Potential hardware operating life
Describing question	Does the manufacturer of the software product guarantee that the current release can be executed on a reference system that is *n* years old? Thus, the software product can be executed on a standard hardware configuration that has already been in operation for *n* years.
Indicators	a) Initially use the specification by the manufacturer (hardware, older operating systems, older frameworks), since no standard configurations have been defined for previous years. b) When this criterion has been applied for a long enough time period, so that the standard usage scenario can also be executed on earlier standard configurations as well: Can the standard usage scenario still be executed with the current release of the software product on a configuration that was the standard configurations *n* years ago (*n* still needs to be specified, e.g. by certifier, see Section 5.2.2)?

The complete set of criteria and corresponding descriptions can be found in the supporting material of Paper 2 and online on http://green-software-engineering.de/criteria-catalog.

Following the structure of the set of criteria, all of the proposed criteria can be found in a causal model "describing the principal mechanisms by which a software product can influence the demand for natural resources" [Paper 2, Figure 3]. The causal model helps to explain the idea of environmental issues of the virtual product "software" by depicting the connection between hardware requirements of

software and resulting hardware capacities used as well as the influence of usage patterns on these capacities and thus, in a next step, on natural resources.

For the purpose of showing practicability of the set of criteria, we then demonstrated the application of four example criteria (Electricity consumption for a standard usage scenario and a standard configuration, default settings supporting resource conservation, backward compatibility, and uninstallability of programs). In that way, we showed the (i) implementation of the proposed criteria and corresponding measurement methods and (ii) the high relevance of the criteria. The results of the application demonstrate that allegedly similar products are quite different when it comes to environmental issues.

Overall, Paper 2 answers RQ2, regarding possible criteria for green software products, upon which an eco-label could be based (see Table 1). However, the set of criteria is more than a first step towards awarding criteria for an eco-label for software products. By discussing the set and its potential future application from the perspective of different stakeholders, we show further areas of implementation. In our point of view, the set of criteria is interesting for users, purchasers, administrators, developers, and certifier. Hence, it offers scientific as well as practical benefits.

4.2.3 Evaluation: user's interest in environmental issues of software

Referring to: Environmental issues of software & its labelling: an end user perspective [Paper 3]

Kern, Eva. Environmental issues of software & its labelling: an end user perspective [in review]

***Abstract**. The energy consumption caused by ICT products is raising. Activities to counteract here are comprised by the term "Green IT". While there was a focus on the hardware side in research activities in the context of Green IT for a long time, research is increasingly addressing software issues. However, it seems as if the topic of environmental issues of software did not reach software users and developers so far. The following paper presents a user survey addressing the awareness of and the interest in environmental issues of software. It turned out that the subject is new to the participants of the survey even if they are generally interested in environmental topics. Nevertheless, most of them can imagine taking environmental issues into account while searching for new software products. The aim is to include the findings in the development of an eco-label for software. Hence, we come up with corresponding recommendations that are based on the survey results.*

The process of searching for criteria for green and sustainable products, presented in the previous section, included mainly researcher. First ideas have been discussed with stakeholders from research, business, and public institutions. Even if all of them are also software user, they look at the issue from a different perspective than those who are solely using software and not caring about anything else like administrating, purchasing, developing or awarding products. Thus, in order to develop an eco-label that raises the interest of its future target group, the user of this potential label have been integrated in the label development process by implementing a user survey. During August and September 2016, 712 online questionnaires have been completed anonymously (see Table 5). By implementing an online survey and spreading it using mailing lists, social media, and similar web-based activities, the

topic that is mainly based in research discussions was, to some extent, brought to society. According to Anwar and Pfahl [2017], using a survey as research method is not that common in the context of green software engineering. Thus, the survey provides new findings and possesses a relatively unique characteristic.

The survey addresses the awareness of and the interest in environmental issues of software and focuses on the view of end users. Here, the terms "users" and "end users" (used synonymously) refer to those who are using software products and thus are the target group of the proposed eco-label. The idea is that this group takes the label into account while they are searching for new software products. Users stand in contrast to developers, administrators, IT managers, purchasers, distributors, and vendors. However, the members of the groups overlap as pointed out in Section 3.1.

Table 5 *Key data of the survey on the awareness of and the interest in environmental issues of software, conducted in 2016 [Paper 5, Table 1]*

Method	Online survey
Structure of the survey	1. Demographic characteristics of the participants 2. Evaluation of criteria for green software products 3. Aspects influencing the acceptance of an eco-label for software products
Participants	Internet users from Germany (and German speaking neighboring countries)
Sample	n = 854
Completed questionnaires	712 (revised data set)
Survey period	16/08 to 05/10/2016

Looking at the survey results, most of the participants assign themselves to the role of software users (78 %). Overall, regarding their demographic data, they can be characterized as follows:

- Software users (78 % of the survey participants)
- Female (55.8 %)
- 20-39 years (68.7 %)
- Higher university degree, means PhD or master's degree (48.3 %)
- Working in the field of natural science / geography / computer science (52.5 %)

In order to analyze their attitude towards environmental topics, the survey comprised questions regarding eco-labels that award, among others, ICT devices. It turned out that most of the participants knew at least one of the presented labels (Blue Angel, Energy Star, TCO Certified): Only 4.6 % answered that they do not know any of them. However, there is quite a lack of knowledge regarding the details of the awarding criteria and awarded products. This is also pointed out by other studies

addressing eco-labels in context of Green IT [Ferreira 2011; Ramachandiran 2012; Selyamani and Ahmad 2015]. Additionally, more than half of the participants (61.4 %) stated that they are willing to pay more for organic products. About one out of four (23.7 %) catch up on environmental topics. In my opinion, it is important to keep the characteristic of the survey participants in mind while interpreting the results of the survey.

Even if the participants know environmental labels and are interested in environmental topics, more than half of them never paid head to an eco-label for software products (56.3 %). Hence, that shows that there is a lack of awareness. This complies with comparable studies in this field addressing other target groups referred above. The participants are generally interested in the addressed field but have never connected their interest to software although they are using software products regularly.

Indeed, 23.2 % of the participants can imagine taking environmental issues as one possible buying criterion when searching for software products. 51.1 % would do that if there are products with the same functionality, some being "green" and others being "not green". Thus, functionality is the main aspect when it comes to selection criteria for software products.

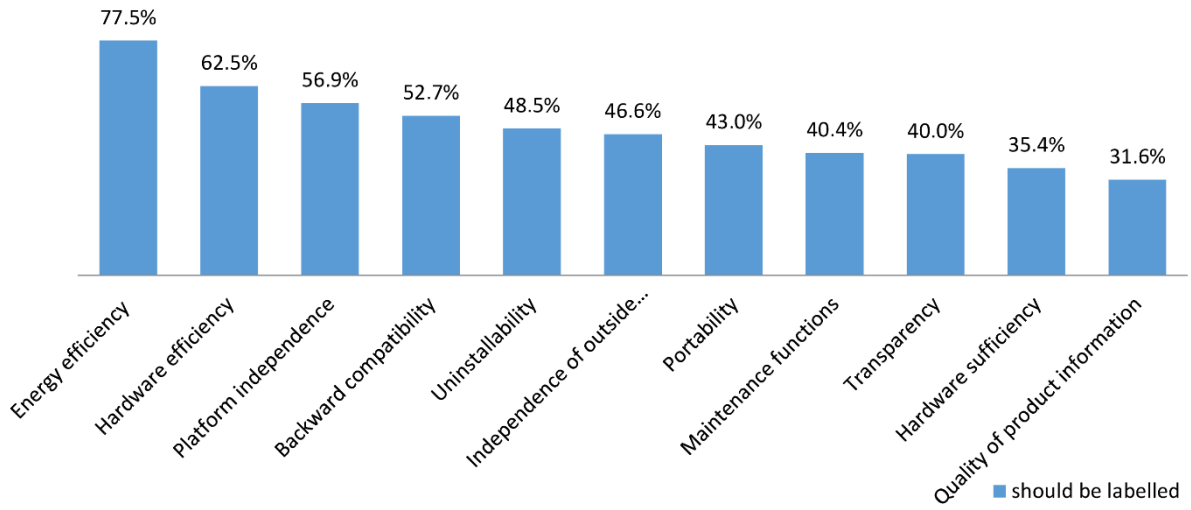

Figure 5 *Answers to the question "Which ones of these environmental issues of software should be labelled?", sorted by the numbers of mentions of "should be labelled"*

Next to the awareness of environmental issues of software, the interest in those issues has been part of the user survey. Thus, I integrated an extract of the set of criteria (see Subsection 4.2.2) into the questionnaire. The survey participants had to point out which ones of the environmental issues of software should be labelled. For the rating, I used a four-point Likert scale [Likert 1932] ranging from "should be labelled", "should rather be labelled", "should rather not be labelled" to "should not be labelled", and "I do not know".

The results, depicted in Figure 5, show that especially energy and hardware efficiency aspects gain the users' interest (77.5 % of the responders say, "energy efficiency" should be labelled, 62.5 % vote for "hardware efficiency"). According to the set of criteria [Paper 2], this reference to issues like

- Energy consumption of the software product that is used to execute a standard usage scenario
- Recommended system requirements and resulting hardware requirements
- Hardware utilization during normal use

Overall, the results show that there seems to be an interest in environmental impacts of software. Even if the awareness is not given so far, the willingness can be seen. This is also shown by the ideas for further criteria for green and sustainable software the survey participants have been asked for.

The additionally mentioned aspects can be summed up to those issues that are listed in Table 6. The table lists the suggestion by the survey participants that are named at least twice in the text field asking for additional ideas for criteria (sorted by number of mentioned). The ideas are combined with the criteria presented in Subsection 4.2.2 and Paper 2. However, within the set of criteria we described the criteria for sustainable software products in a more detailed way than presented in the user survey.

Table 6 *Summary of proposed aspects by the participants of the survey in comparison to the criteria collected in the set of criteria for sustainable software [Paper 2]*

Suggestion	**Mentioned in the set of criteria [Paper 2]?**	**Comments / why not mentioned**
Environmental and sustainable aspects regarding the producer	No	Focus on the software product itself (not the development process)
Use of external resources	Yes, belongs to hardware efficiency	(estimated)
Updates & durability	Yes, belongs to transparency (and interoperability)	
Working conditions during the software development	No	Focus on the software product itself (not the development process)
Software-delivery	Yes, belongs to hardware efficiency	
Licensing	No	Focus on environmental aspects
Energy	Yes (in different manners)	
Adequate / scalable performance	Yes, belongs to hardware efficiency	
Data security	No	Focus on environmental aspects
Background activities	Yes, belongs to transparency (and interoperability)	
Undesirable additional programs	No	Focus on environmental

		aspects
Operability and usability	No	Focus on environmental aspects
Data usage	Yes, belongs to resource efficiency	
Interfaces / standards	Yes, belongs to transparency (and interoperability)	
Backward compatibility	Yes, belongs to backward compatibility	
Compatibility to other software products	Yes, belongs to hardware efficiency	

Summarizing, the topic of environmental issues of software is new to the survey participants even if they are generally interested in related topics. Especially efficiency aspects raise the interest of the interviewed persons. Thus, these should be in the focus while further developing an eco-label for software products. Hence, the results of the ranking of the criteria for sustainable software products by the participants survey directly address RQ3 "Which aspects of green software products raise the highest interest of users and should be labelled?".

4.2.4 Evaluation: aspects influencing the acceptance of an eco-label for software

Referring to: Communicating Environmental Issues of Software: Outline of an Acceptance Model [Paper 4]

Kern, Eva. 2016. Communicating Environmental Issues of Software: Outline of an Acceptance Model. Advances and New Trends in Environmental Informatics, Wohlgemuth, V., Fuchs-Kittowski, F., and Wittmann, J. (Eds.). Springer International Publishing, Cham., 335-345. http://dx.doi.org/10.1007/978-3-319-44711-7_5

***Abstract**. During the last years, the research activities regarding software and its environmental impacts could find their way into the field of "Green IT". Thus, researchers became aware of the fact that software is one of the drivers of the energy consumption by ICT. However, the awareness for these aspects could be much higher – especially in the non-scientific area. On the one side, software developers should be aware of green strategies of software engineering. On the other side, those using the products need to be responded to the effects of using ICT products onto the environment. One idea to transfer the environmental effects of software into a social issue is to create an eco-label for software products. Next to defining criteria, such a label could be laid on, and methods to evaluate software products, it seems to be helpful to identify aspects influencing the acceptance of a certification for green software products. In this context, acceptance means taking the eco-label into account while searching for new software. Hence, the following paper aims at identifying those aspects by applying the Technology Acceptance Model (TAM 2) to the specific case of labelling green software products. We will present a first version of an acceptance model for a label for green software products. It is still work in-progress and needs to be evaluated as a next step. Generally, the aim is to create a tool that can be used to develop an eco-label for software products that will be strongly accepted.*

So far, there is no standardized eco-label for software products. Nevertheless, aspects influencing the acceptance of such a label seem to be interesting for the label development process. Thus, potential aspects that might influence the acceptance of the future label are summed up within the acceptance

model “Labelling Green Software Products” (LGSP, Figure 6). The first version of the model is presented in Paper 4 in detail. Within this frame paper for the doctoral thesis, the model is outlined and combined with the results of the third part of the survey.

The model comprises three kinds of aspects: (i) constructs related to the Technology Acceptance Model (TAM) [Davis Jr 1986], (ii) constructs related to the person of interest, and (iii) constructs related to the product itself. It is based on the Technology Acceptance Model 2 presented by Venkatesh and Davis [2000] and complemented by aspects that might further influence the acceptance of an eco-label for software products, while TAM aspects that do not fit in the use case “eco-labelling software products” were deleted (applies to: relevance, output quality, and voluntariness; the explanation for doing so can be found in Paper 4). Additionally, some of the constructs suggested by Venkatesh and Davis [2000] have been transferred to the case of eco-labelling software, taking into consideration that the label is still under development. Thus, instead of talking about an “attitude towards using”, the proposed LGSP model terms “intention to use” referring to the “general notion that an attention to a certification of environmental impacts caused by software makes sense” [Paper 4]. Similarly, the originally proposed “actual system use” in the Technology Acceptance Model [Davis Jr 1986] is replaced by “(potential) usage behavior”. Both models – TAM 2 and LGSP – are talking about image. While this construct refers to a person in TAM 2, the LGSP sees the image, i.e. the reputation, of the eco-label in general and especially for software products. Additionally, the model includes hypotheses about the relation of the constructs (H). They are presented in Paper 4 in detail. These hypotheses have been taken as starting point for recommendations for developing an eco-label of software products.

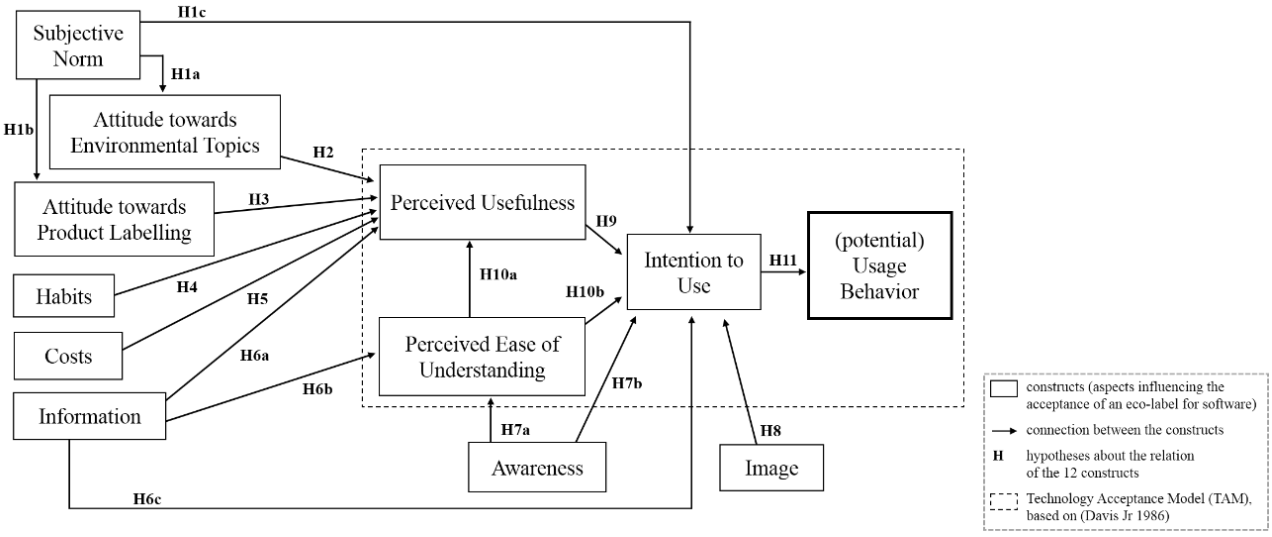

Figure 6 *Outline of an acceptance model for a label for green software products (LGSP Acceptance Model) [Paper 4, Figure 1]*

Regarding the results of the user survey, a perceived ease of understanding and a perceived usefulness seem to be especially important. The label should be directly comprehensible (71.3 % of

the participants agree, 24.4 % rather agree) and preferably self-explaining (53.4 % agree, 39.0 % rather agree).

The participants appreciate the possibility to find further information on the eco-label and its awarding criteria (56.7 % agree to "It is important for me to be able to read up about the product label in a detailed way."; 40.0 % rather agree and 29.4 % agree to "I am ready to use additional information sources to understand a product label."). They understand the awarding of environmental issues of the software product as an enhancement of the product and tend to pay more for the product (Table 7). For the participants, the reliability of the label is more important than the reputation of the label or its critique. Summarizing, from the participants' view, an eco-label seems to be sensible (51.7 % agree to "Labelling green software products makes sense to me.") and has the potential to raise the interest towards Green IT issues (33.1 % agree, 42.6 % rather agree). Thus, a general interest in an eco-label for software products seems to be given [Paper 3]. The future label development process should keep the proposed acceptance issues in mind.

Table 7 *Results of the evaluation of statements belonging to the aspect "costs"*

Statement	**I agree**	**I rather agree**	**I rather not agree**	**I do not agree**	**I do not know**
Software that is awarded being "green" is accepted to be more expensive.	13.8 %	**46.1 %**	27.2 %	6.5 %	6.5 %
I think, the reduction of environmental issues of software is an excellent value of money.	24.4 %	**43.4 %**	19.9 %	5.6%	6.6 %
The value of a software product is increased by labeling its environmental issues.	25.1 %	**43.7 %**	16.6 %	6.9%	6.6 %

Please note: The answers have been interpreted against the assumption „the higher the agreement towards the statements, the more important is the aspect for the responder.

Overall, the LSPG Acceptance Model [Paper 4 and Figure 6] addresses RQ4 "What are aspects influencing the user acceptance of an eco-label for software products?" In the survey participants' view the aspects "perceived usefulness" and "perceived ease of understanding", regarding the label and its statements, or rather the idea behind the label, seem to be especially important. They expect additional information on the eco-label, a certification that is easy to understand, self-explaining, and useful in the purchasing process of software. Thus, the findings of the survey can be summed up by presenting four recommendations for the development of an eco-label for software [Paper 3]: "(i) Keep it simple and understandable. (ii) Use the prominence of existing eco-labels to push the awareness of an eco-label for software. (iii) Keep the market relevance of products in mind. (iv) Set the focus on publishing a certification that is understandable at first glance, instead of creating a complex information system."

5 Conclusion and future directions

Summarizing, this doctoral thesis can act as a basis for further research in bridging from science to society in the context of environmental issues of software. Additionally, its findings can be seen as starting points for practical implementations of methods and tools (i) supporting a more environmentally friendly way of developing software (Subsection 4.1) and (ii) informing about environmental issues of software usage (Subsection 4.2). This framework paper outlines the central aspects of the five papers included in this cumulative doctoral thesis (Table 1). The following section presents the main findings and thus the contribution to the research field of environmental and sustainability informatics [Naumann 2008], condenses the implications for the target groups of the thesis (see Subsection 3.3), points out strength as well as limits and concludes with an outlook for future work.

5.1 Summary of main findings and contribution to research

In this doctoral thesis, the focus is on the question how to bring the scientific ideas of green software to developers and users (not being part of the scientific community). To do so, four research questions have been formulated (Table 1), derived from the main research question:

How to draw (a) developers' and (b) users' attention to environmental issues of software?

First, the starting idea regarding addressing developers is that they could be informed about environmental issues of software by calculating the carbon footprint of the development of software products.

RQ1: How can the carbon footprint of the development of software products be calculated?

Referring to this question, Paper 1 presents a method on how to calculate the carbon footprint of the development phase of software products. The calculation method includes employees, office space, and ICT infrastructure as factors that influence the resulting CO_2 emissions. It turns out that especially the commuting of the employees has a big influence on the carbon footprint of the software development phase. In addition to the calculation method for the carbon footprint, we recommend gathering the energy consumption of the software product during the development phase. Thus, a method for a continuous measurement of the energy consumption of the developed software product is presented that can be applied during the development process. Approaches to this are described in Paper 1 and Paper 5. Both methods – carbon footprint calculation and continuous energy measurement – introduce additional effort for the developers. However, this overhead will be more and more reduced when the developers are trained in the usage of the methods. It is important that the methods do not change the software engineering process of the developers in general.

Second, the starting idea regarding addressing users is that they could be informed about environmental issues of software by providing an eco-label for software products.

RQ2: What are criteria for green software products, upon which an eco-label could be based?

Possible criteria for green and sustainable software products have been presented in form of a set of criteria, published in Paper 2. To do so, we developed a causal model to have a guideline to structure the criteria and analyzed existing literature, guidelines and awarding criteria in the addressed field. The resulting structure comprises three levels: (1) Resource efficiency, (2) Potential hardware operating life, and (3) User autonomy. Thus, the focus is set on environmental impacts that occur during the usage phase of a software's life cycle.

RQ3: Which aspects of green software products raise the highest interest of users and should be labelled?

In order to find out the interest of software users, being the potential main future target group for the proposed eco-label for software, an online user survey has been conducted [Paper 3]. One of the main findings is that the survey participants did not think about environmental issues of software, even if they are relatively interested in environmental topics. However, they can imagine this kind of label. It turns out that they are mainly interested in energy and hardware efficiency aspects. In contrast, the interest in the quality of product information is small. Nevertheless, none of the proposed criteria (descending interest: energy efficiency, hardware efficiency, platform independence, backward compatibility, uninstallability, independence of outside resources, portability, maintenance functions, transparency, hardware sufficiency, quality of product information) was voted as "should not be labelled".

RQ4: What are aspects influencing the user acceptance of an eco-label for software products?

In addition to analyzing the awareness of environmental issues of software and evaluating the criteria for green software products, potential aspects influencing the user acceptance of an eco-label for software have been addressed by the user survey and summed up in the so called "Labelling Green Software Products (LGSP) Acceptance Model" [Paper 4]. For those participating in the survey, it is important to directly understand the label and its statements, or rather the idea behind the label ("perceived ease of understanding" and "perceived usefulness"). This is accompanied by a high relevance of the information content of the product certification itself. Overall, aspects related to the product itself seem to be more important than those related to the person of interest (the users themselves).

Summarizing, the main findings of the doctoral studies and contributions to research are:

Finding: There are various definitions and corresponding criteria for "green" or rather "sustainable software" so that it is hardly possible to look through all the different approaches.

Contribution: Using an exploratory research design and combining scientific approaches and publications with a practical orientation, e.g. awarding criteria, we presented a set of criteria for green and sustainable software products [Paper 2]. By concentrating on the effects on the environment, occurring while using software products (according to a life cycle of software products as proposed in [Naumann et al. 2011]), and green-in-software aspects, the set of criteria has a clear focus. However, it clearly provides the opportunity to be extended in different directions, e.g. covering also effects of developing software products.

Finding: The awareness of environmental issues of software is missing, on the private as well as on the professional side of using ICT, for different user groups – starting from developers, over administrators, managers to software users.

Contribution: The presented set of criteria for green and sustainable software products [Paper 2] can be taken as basis for awarding criteria for an eco-label for software products. The criteria ideas have been evaluated by a user survey [Paper 3]. The results imply the users' interest in environmental issues of software. Based on the results of the survey, recommendations have been formulated that will be further detailed in the next Subsection 5.2. Generally, the lack of addressing people's awareness [Verdecchia et al. 2017] has been approached. The user survey can be seen as one step forward in creating knowledge on green issues of software in order to enable the user to change something in the context of impacts on the environment caused by software [Ferreira 2011]. Additionally, the presented LGSP Acceptance Model [Paper 4] summarizes aspects that might be relevant in case of acceptance of the future eco-label and should be integrated in the label development process.

Finding: Environmental issues of software raise the interest of software developers and users, even if the functionality of software products is priority 1.

Contribution: This thesis presents two approaches on how to create, or rather increase the awareness of environmental issues of software. One idea is to calculate the carbon footprint of a software development process in order to get an impression of its environmental impact and to provide this information to software developers [Paper 1]. Another idea is to create a label

for green software products in order to address especially software users [Papers 2, 3, 5]. According to the Agenda 21 for the United Nations, providing information could be seen as one possibility to support an environmentally friendly selection behavior [Horne 2009]. Hence, by also letting (especially) non-academics know that there is a connection between software usage and environmental issues, they could be enabled to act more environmentally friendly while developing and using ICT in general and software, as a driver of the energy consumption of hardware, in particular. These changes in behavior might lead to less environmental impacts of ICT. The approaches can be understood as starting points for environmental education and communication processes in the context of green and sustainable software.

Finding: The practical integration of methods to decrease the environmental impacts of software development and corresponding tools are missing.

Contribution: In order to get an impression of the environmental impact of a software product and to take sustainability issues into account while developing software, we propose to continuously integrate energy efficiency measurements into the development process. Additionally, calculating the carbon footprint of a software development phase provides information on environmental impacts of software. The approaches presented in Paper 1 can be integrated in the development phase. 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 and Pfahl [2017]. Over time, providing information of the energy consumption of software can help the developers to get a feeling for the energy values of the products they are working on [Manotas et al. 2016]. 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" [Paper 1]. Hence, in order to meet the requirements of practitioners [Groher and Weinreich 2017], our approach for energy efficiency measurements can be adapted in existing working structures very well.

The main meta-contribution of this thesis is the integration of the view of users into current research activities in the field of environmental and sustainability informatics. It expands the view of the scientific community of the field of environmental and sustainable informatics upon the big group of users and also on awareness aspects. Until now and as far as I am aware, especially the users'

perspective was mostly ignored. However, users are identified as being of high relevance because they are important change makers in bringing research ideas towards implementation and thus having the desired effects.

Integrating awareness strategies into the software development process and the idea of labelling green software can be seen as innovative concepts. They can help to structure the ideas of characterizing green and sustainable software products. Thus, the approach enlarges transparency and, in the long run, knowledge. By doing so, it provides a sound basis for further research.

As a next step, these findings will be transferred towards implications and thus differentiated by the four main target groups identified in Subsection 3.3. In that way, the demands can be specifically formulated for each of the target groups. Thus, it is easier to define the next steps to go on with the findings of this doctoral thesis and bring the approaches into practice.

5.2 Implications: recommendations towards stakeholders based on the findings

"Just having some knowledge on sustainability is not enough. It is crucial that sustainability knowledge is relevant to the particular business' endeavors and properly communicated, which could lead to improvements of social and environmental performance." [Johnson 2015]

Transferring this statement towards the case of green software, it is important that the idea of informing about environmental issues is brought towards those who are part of the software life cycle. In this context, Carrington et al. [2010] talk about a gap between an intention to do something and the actual behavior. Here, we set the focus on developers (see Subsection 4.1) and users (see Subsection 4.2). While those are the objectives of the information approaches (Subsection 3.3), further stakeholders should be addressed in order to bring the ideas from science to society. The following subsections present practical implications for diverse groups of stakeholders that have been identified as being important for the practical implication of the presented concepts and, thus, represent the target group of the doctoral thesis (see Subsection 3.3).

Some of the implications have already been discussed in context of the set of criteria for sustainable software products in Paper 2 and in connection with the recommendations for the label development process presented in Paper 3.

5.2.1 Implications for researchers

The research regarding environmental impacts of software is just in its infancy [Penzenstadler et al. 2014a; Verdecchia et al. 2017], especially when it comes to user integration. Within my doctoral studies, I took a step forward in aiming at a collective understanding of green software, on the one

hand, and at closing the gap between science and society, or rather industrial practice regarding information on green software, on the other hand. However, these starting points need to be further researched and developed.

Within further research activities, software developers should be seen as important stakeholders in moving forward in green software engineering. Thus, it seems to be sensible to include them into research activities since software developers are the ones who also should accept and especially implement changing requirements in software engineering. The information in form of a carbon footprint of the software development process and information regarding the energy consumption of the software product, as results of energy measurements, should be extended by recommendations on how to improve the software product. So far, implications on how to transfer the knowledge regarding energy efficiency of software into practical routines are missing [Selyamani and Ahmad 2015; Chitchyan et al. 2016; Manotas et al. 2016; Pang et al. 2016]. Researchers in the field of environmental informatics are asked to find solutions to counteract here. Doing so, it seems to be sensible to integrate industry and individual programmers. The activity could aim at elaborating guidelines for software developers on (i) how to interpret the results of calculating the carbon footprint of software development and energy measurement data of software products and (ii) how to improve the development process as well as the software product in regards to environmental issues based on the results.

Additionally, future research activities should address software users. Thus, researchers in the field of sustainability and environmental informatics are asked to extend the development process of an eco-label for software. Possible next steps could be the improvement of the presented approaches to prove the validity of the set of criteria and thus supporting certifiers in elaborating program requirements for an eco-label for software. As soon as there is a draft for such a label, the user acceptance can be analyzed by researchers, e.g. based on the hypothesis presented in Paper 4. This could be done by implementing further surveys and thus extending the research methods used in the field of green software engineering [Anwar and Pfahl 2017].

Overall, researchers in the field of environmental informatics are asked to continue the process on the basis of (i) the set of criteria presented in Paper 2, (ii) the evaluation results of the user survey and (iii) the corresponding recommendations presented in Paper 3. Beyond that, the application methods [Paper 2] should be further evaluated and extended by tools as well as guidelines for those researchers and practitioners who are new to integrating environmental issues in their work. In that way, they could support certifiers. Similarly, the methods for calculating the carbon footprint of software development [Paper 1] should be proven by corresponding case studies.

5.2.2 Implications for certifiers

So far, there is no standardized eco-label for software products, even if the interest in such a label seems to be given (see Subsection 4.2.3). Thus, being experts in this field, certifiers are asked to encourage the label development process – preferably in close collaboration with researchers of the field of environmental informatics. This means that the set of criteria should be transferred into program guidelines for an eco-label, e.g. the Blue Angel. The presented set of criteria for sustainable software products [Paper 2] outlines an initial point for this process. In doing so, the recommendations based on the survey results presented in Paper 3 should be taken into consideration:

1. Keep it simple and understandable.
2. Use the prominence of existing eco-labels to push the awareness of an eco-label for software.
3. Keep the marked relevance of products in mind.
4. Set the focus on publishing a certification that is understandable at first glance instead of creating a complex information system.

The details of creating an eco-label are regulated in ISO 14024 [International Organization for Standardization 1999]. One possibility is to follow the steps described in the Product Sustainability Assessment (PROSA) method (definition of objective, analysis of market and context, brainstorming, sustainability assessment, strategy development [Grießhammer et al. 2007]). Overall, it seems to be important to set an emphasis on providing information on the label, its awarding criteria, and the products being awarded, in order to support transparency, interest, and understanding.

5.2.3 Implications for public administration and professional purchasers

Since the issue of environmental impacts caused by software over its whole life cycle is new, corresponding strategies to integrate these aspects in procurement processes are missing. In this context, public administration and professional purchasers could serve as models and thus motivators for private users. Professional purchasers are responsible for the acquisition of many software products and thus have a considerable influence on resulting environmental effects of these products while they are used in companies and organizations. Furthermore, public administration has purchasing directives that could include environmental aspects for software products. Thus, public administration and professional purchasers are asked to extend the procurement guidelines for software by environmental impacts inspired by the set of criteria [Paper 2]. In that way, they would create a (higher) demand for an eco-label for software products. In the long run, private users can follow the example. Overall, the big players in regard to software purchasing can act as a model in the consideration of environmental aspects of software.

5.2.4 Implications for environmental associations

The user survey implemented in the context of the doctoral studies and similar studies [Ferreira 2011; Durdik et al. 2012; Souza et al. 2014; Chitchyan et al. 2016; Manotas et al. 2016; Pang et al. 2016; Groher and Weinreich 2017] show that there is a lack of knowledge about environmental impacts of software. With regard to information activities on environmental topics, environmental associations are especially an active source of information for society, enjoying consumers' trust. Thus, environmental associations are asked to inform about environmental issues of software. They are well experienced in bringing complex topics to society in an understandable way. In order to start a corresponding information and education process, adequate teaching materials are required. They should be created in a joint project of researchers in the field of green software and experts from environmental associations. Special interest could be laid on the explanation of the consequences of usage behavior in using ICT devices. The aim is to point out potentials on how to use ICT devices and software running on them in a way that is more environmentally friendly and, when extended, sustainable. This could start with small steps like turning off background activities of software.

In this context, the carbon footprint of a software product as well as an eco-label for software could be helpful information media. Creating awareness is a first step towards a "greener way" of using ICT and – in the long run – to reduce the impacts on the environment caused by ICT.

Overall, the connection between the activities of the different stakeholder groups and, thus, an approach for collaboration in the context of eco-labelling is presented by Herzog et al. [2015]. They point out that, firstly, industry alliances and/or academic researchers provide materials and tools that are, secondly, taken by global and local standardization organizations to become standards to, thirdly, be transferred towards regulations, recommendations, and labels published on a governmental level. This procedural approach is transferrable to the activities presented in this thesis having the target groups, described in Section 3.3, in mind.

5.3 Strengths, limits and future directions

The doctoral thesis at hand provides innovative research results that will be presented in detail in the following section. Additionally, I will challenge the research done and look out on potential future directions in the scientific field of green and sustainable software.

5.3.1 Strengths

By providing a structured set of criteria for sustainable software products, including the view of software users into research activities in this field, and evaluating scientific ideas with respect to applicability and social interest, the doctoral thesis extends existing research results.

Combined approach of existing research ideas resulting in a hierarchical set of criteria

Since the start of the research activities in this field, the question on how to characterize green and sustainable software raises the interest of several researchers and scientific projects. However, a combined and structured approach of existing criteria ideas is missing so far. This research gap has been addressed by presenting the set of criteria to evaluate environmental issues of software. The strength of this set of criteria is the clear hierarchical order of criteria and indicators and the clear focus on impacts on the environment as well as the usage phase.

Summarizing, the doctoral thesis at hand makes one step ahead in aiming towards a collective understanding of green and sustainable software. Thus, it presents a basis for future research activities.

Integration of practitioners

Providing new scientific results is one issue. However, since the field of research is of high relevance for society, it seems to be important to integrate those who can act as change makers in this context. This has been done by implementing a survey focusing on the perspective of software users. Additionally, in the case at hand, it is essential to provide solutions for software developers who are in charge of satisfying the needs of users, i.e. providing green(er) software products. To do so, programmers need information, tools, and knowledge.

Summarizing, the doctoral thesis at hand makes one step ahead in bringing scientific ideas to society. Thus, it presents a starting point for increasing the awareness of environmental issues of software, especially regarding software developers and users.

Evaluation

While the calculation methods for carbon footprinting software and the energy measurements of software are mainly theoretical scientific approaches so far, the idea of an eco-label for software products was additionally evaluated by software users. In doing so, it turned out that there is an interest in the proposed criteria for green software and the overall label. Thus, it is possible to provide recommendations that are based on empirical data of more than 700 survey participants.

Summarizing, the doctoral thesis at hand makes one step ahead in integrating the view of users in research activities. Thus, it presents scientific results that are evaluated by non-academia.

5.3.2 Limits

The doctoral studies extend current research activities by aspects of user integration and information issues. However, there are some limitations of the work done: reliability of data, proof of concept, and setting of priorities.

Reliability of data

The limits of the user survey are discussed in detail in Paper 3. Summarizing, the main weakness of the study is the reliability of the data. Since most of the surveys' participants already have a personal interest in environmental topics and a similar demographical background (being in the age of 20 to 39 years, holding an academic degree, and working in the field of natural science / geography / computer science), the results might be biased. However, they provide an impression of the societal evaluation of the proposed criteria by including the opinion of 712 persons.

Proof of concept

While the evaluation of the set of criteria count as a strength of the doctoral thesis, such kind of proof of concept is missing in case of the calculation method for the carbon footprint of software development and of the continuous energy efficiency measurements. However, the calculation method is tested with statistical data provided by different independent sources. This is an indication for the plausibility of the calculation method. The gap of missing data from a real use case will be addressed in future research activities. The measurement method for the energy efficiency of software products has been tested by several test cases [Kern et al. 2011; Dick et al. 2012; Guldner et al. 2018], even if the transfer towards continuous integration within bigger or even industrial software project is missing so far.

Setting of priorities

In order to be able to address specific research issues, a well-defined frame for the doctoral studies had to be found. Hence, this follows along with decisions for one direction excluding other points of interests. This can be interpreted both as strength and as limitation. Here, this applies for the focus on environmental aspects of sustainability and the usage phase. Furthermore, software has been considered in a general way. Nevertheless, I am aware of the connection of the dimensions of sustainability as defined in the Brundtland report (environment, social, economy) [United Nations General Assembly 1987] and further extensions (e.g. [Goodland 2002a; Penzenstadler and Femmer 2013; Razavian et al. 2014; Lago et al. 2015]). I decided to concentrate on one aspect of sustainability as a first step in order to make the approach more concrete. Taking a broader look at software [IEEE Computer Society 2010] allows to create a general overview of the objective at first and then specialize the results towards the diverse kinds of software, e.g. separating it into groups of desktop software, mobile software, web-based software, etc. However, the set of criteria has been developed having diverse kinds of software in mind [Paper 2].

5.3.3 Future directions

Researching environmental issues of software and the awareness of this topic resulted in new findings on the one hand and brought up additional need for research on the other hand.

Testing of methods

In order to prove the calculation method for the carbon footprint of software development, it should be implemented in projects that differ in the field of application and in size. As soon as there is data of carbon footprint calculations, it might be sensible to set up a data base for sharing these results. Thus, when calculating the carbon footprint of a software development process for the first time, own outcomes can be interpreted in comparison to existing results. The effort of software projects with regards to CO_2 emissions will be more concrete by having this opportunity to compare.

Further development and optimization

Based on the results of testing the methods for evaluating software development processes, regarding their carbon footprint, and software under development, regarding energy consumption, the methods can be further developed and improved. This is also possible in the case of the presented set of criteria and its application [Paper 2]: the findings can be transferred to awarding criteria for an eco-label that might be weighted, or procurement guidelines. The application of the criteria could be elaborated by extending the test cases and providing corresponding guidelines on how to implement test procedures.

Extension

As already pointed out, the doctoral thesis at hand takes a specific view on sustainable software, e.g. by setting priorities on environmental aspects, usage phase, and software products in general. Thus, next steps in research could extend this point of view by expanding the criteria by social and economic aspects and addressing also impacts in the development phase of software. Similarly, the carbon footprint of different life cycle phases could be calculated. In this context, especially the comparison between development and usage phase seems to be interesting as theoretically discussed in Paper 1. As soon as there is a national version of an eco-label or a sustainability certification for software products, the question of how to bring this label into an international context could be an issue.

References

ABENIUS, S. 2009. Green IT & Green Software - Time and Energy Savings Using Existing Tools. In *EnviroInfo 2009: Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools. proceedings of the 23rd International Conference Environmental Informatics - Informatics for environmental protection, sustainable development and risk management, September 09 - 11, 2009, HTW Berlin, University of Applied Sciences, Germany*, V. WOHLGEMUTH, B. PAGE AND K. VOIGT, Eds. Shaker, Aachen, 57–66.

AFZAL, S., SALEEM, M.F., JAN, F., AND AHMAD, M. 2013. A Review on Green Software Development in a Cloud Environment Regarding Software Development Life Cycle:(SDLC) Perspective. *International Journal of Computer Trends and Technology (IJCTT) 4*, 9, 3054–3058.

AGARWAL, S., NATH, A., AND CHOWDHURY, D. 2012. Sustainable Approaches and Good Practices in Green Software Engineering. *IJRRCS 3*, 1, 1425–1428. http://scholarlyexchange.org/ojs/index.php/IJRRCS/article/view/9903/7030.

AHMAD, R., BAHAROM, F., AND HUSSAIN, A. 2014. A Systematic Literature Review on Sustainability Studies in Software Engineering. In *Proceedings of KMICe*.

AMSEL, N., AND TOMLINSON, B. 2010. Green Tracker: a tool for estimating the energy consumption of software. In *CHI EA '10: Proceedings of the 28th international conference extended abstracts on Human factors in computing systems*, Association for Computing Machinery, Ed. ACM, New York, 3337–3342.

ANDRAE, A.S.G., AND EDLER, T. 2015. On global electricity usage of communication technology: trends to 2030. *Challenges 6*, 1, 117–157.

ANWAR, H., AND PFAHL, D. 2017. Towards greener software engineering using software analytics: A systematic mapping. In *Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on*, 157–166.

ATKINSON, L., AND ROSENTHAL, S. 2014. Signaling the green sell: the influence of eco-label source, argument specificity, and product involvement on consumer trust. *Journal of Advertising 43*, 1, 33–45.

BETZ, S., AND CAPORALE, T. 2014. Sustainable Software System Engineering. In *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on*, 612–619.

BOZZELLI, P., GU, Q., AND LAGO, P. 2013. *A systematic literature review on green software metrics*. Technical Report: VU University Amsterdam. http://sis.uta.fi/pt/TIEA5\ Thesis\ Course/Session\ 10\ 2013\ 02\ 18/SLR\ GreenMetrics. pdf.

BRITISH STANDARDS INSTITUTION. 2011. *PAS 2050:2011. Specification for the assessment of the life cycle greenhouse gas emissions of goods and services*. http://www.bsigroup.com/Standards-and-Publications/How-we-can-help-you/Professional-Standards-Service/PAS-2050.

CALERO, C., BERTOA, M.F., AND ANGELES MORAGA, M. 2013a. A systematic literature review for software sustainability measures. In *2nd International Workshop on Green and Sustainable Software (GREENS)*, 46–53.

CALERO, C., MORAGA, M., AND BERTOA, M.F. 2013b. Towards a software product sustainability model. *arXiv preprint arXiv:1309.1640*.

CALERO, C., AND PIATTINI, M. 2015. Introduction to Green in Software Engineering. In *Green in Software Engineering*, C. CALERO AND M. PIATTINI, Eds. Springer, 3–27.

CARRINGTON, M.J., NEVILLE, B.A., AND WHITWELL, G.J. 2010. Why ethical consumers dont walk their talk. Towards a framework for understanding the gap between the ethical purchase intentions and actual buying behaviour of ethically minded consumers. *Journal of Business Ethics 97*, 1, 139–158. http://www.jstor.org/stable/pdf/40929378.pdf.

CHITCHYAN, R., BECKER, C., BETZ, S., DUBOC, L., PENZENSTADLER, B., SEYFF, N., AND VENTERS, C.C. 2016. Sustainability design in requirements engineering: state of practice. In *Proceedings of the 38th International Conference on Software Engineering Companion*, 533–542.

DAVIS JR, F.D. 1986. *A technology acceptance model for empirically testing new end-user information systems: Theory and results*. PhD Thesis, Massachusetts Institute of Technology.

DEUTSCHES INSTITUT FÜR NORMUNG E.V. 2003. *Umweltmanagement-Integration von Umweltaspekten in Produktdesign und -entwicklung. Deutsche und englische Fassung ISO/TR 14062:2002*. DIN-Fachbericht. Beuth, Berlin.

DICK, M., KERN, E., JOHANN, T., NAUMANN, S., AND GÜLDEN, C. 2012. Green Web Engineering - Measurements and Findings. In *EnviroInfo 2012: Man Environment Bauhaus: Light up the Ideas of Environmental Informatics. Proceedings of the 26th International Conference on Informatics on Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management; Federal Environment Agency, Dessau, 2012*, H.-K. ARNDT, G. KNETSCH AND W. PILLMANN, Eds. Shaker Verlag, Aachen, 599–606.

DICK, M., AND NAUMANN, S. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany*, K. GREVE AND A. B. CREMERS, Eds. Shaker, Aachen, 706–715.

DOOKHITRAM, K., NARSOO, J., SUNHALOO, M.S., SUKHOO, A., AND SOOBRON, M. 2012. Green computing: an awareness survey among university of technology, mauritius students. In *Conference Proceeding of International Conference on*

Higher Education and Economic Development, Mauritius. Available from http://tec. intnet. mu/pdf% 20downloads/confpaper/confpaper091224. pdf.

DRANGMEISTER, J., KERN, E., DICK, M., NAUMANN, S., SPARMANN, G., AND GULDNER, A. 2013. Greening Software with Continuous Energy Efficiency Measurement. In *Workshop Umweltinformatik zwischen Nachhaltigkeit und Wandel, Koblenz 2013*, 940–951.

DURDIK, Z., KLATT, B., KOZIOLEK, H., KROGMANN, K., STAMMEL, J., AND WEISS, R. 2012. Sustainability guidelines for long-living software systems. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, 517–526.

EASTERBROOK, S., SINGER, J., STOREY, M.-A., AND DAMIAN, D. 2008. Selecting empirical methods for software engineering research. *Guide to advanced empirical software engineering*, 285–311.

EPA ENERGY STAR. 2014. *ENERGY STAR Program Requirements Product Specification for Computers: Eligibility Criteria, Version 6.1*. Environmental Protection Agency.

FERREIRA, M. 2011. *Green Software Awareness Survey. Results*. Presented at Report Workshop Green Software Architecture, Tuesday 7 June 2011, Amsterdam, Netherlands, Amsterdam.

GÖHRING, W. 2004. The memorandum "sustainable information society". In *Proceedings 18th International Conference Informatics for Environmental Protection, EnviroInfo, Geneva*.

GOODLAND, R. 2002a. *Encyclopedia of global environmental change, chapter Sustainability: Human, social, economic and environmental*. Wiley & Sons.

GOODLAND, R. 2002b. *Encyclopedia of global environmental change, chapter Sustainability: Human, social, economic and environmental*. Wiley & Sons.

Governing Concil of the United Nations Environment Programme. 2003. *Background Paper for the Ministerial Level Consultations: Promoting sustainable consumption and production patterns*, Nairobi.

GRIEßHAMMER, R., BUCHERT, M., GENSCH, C.-O., HOCHFELD, C., MANHART, A., REISCH, L., AND RÜDENAUER, I. 2007. *PROSA - Product Sustainability Assessment. Guideline*. http://www.prosa.org/fileadmin/user_upload/pdf/leitfaden_eng_final_310507.pdf.

GROHER, I., AND WEINREICH, R. 2017. An Interview Study on Sustainability Concerns in Software Development Projects. In *Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on*, 350–358.

GULDNER, A., GARLING, M., MORGEN, M., NAUMANN, S., KERN, E., AND HILTY, L.M. 2018. Energy Consumption and Hardware Utilization of Standard Software. Methods and Measurements for Software Sustainability. In *From Science to Society*. Springer, 251–261.

HERZOG, C., LEFÉVRE, L., AND PIERSON, J.-M. 2015. Actors for Innovation in Green IT. In *ICT Innovations for Sustainability*. *Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 49–67.

HILTY, L., LOHMANN, W., BEHRENDT, S., EVERS-WÖLK, M., FICHTER, K., AND HINTEMANN, R. 2015. Green Software. Final report of the project: Establishing and exploiting potentials for environmental protection in information and communication technology (Green IT). Report commissioned by the Federal Environment Agency, Berlin, Förderkennzeichen 3710 95 302/3, 23.

HILTY, L.M. 2008. *Information technology and sustainability. Essays on the relationship between ICT and sustainable development*. Books on Demand, Norderstedt.

HILTY, L.M., AND AEBISCHER, B. 2015. ICT for Sustainability: An Emerging Research Field. In *ICT Innovations for Sustainability*. *Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 3–36.

HORNE, R.E. 2009. Limits to labels: The role of eco-labels in the assessment of product sustainability and routes to sustainable consumption. *International Journal of Consumer Studies 33*, 2, 175–182.

IEEE COMPUTER SOCIETY. 2010. *Systems and Software Engineering-Vocabulary*, ISO/IEC/IEEE 24765: 2010.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Carbon footprint of products - Requirements and guidelines for quantification and communication*, 14067.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. 1999. *Environmental labels and declarations - Type I environmental labelling - Principles and procedures (ISO 14024:1999) 13.020.50*, DIN EN ISO 14024:2001-02. http://iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?ics1=13&ics2=020&ics3=50&csnumber=23145.

JAGROEP, E., BROEKMAN, J., VAN DER WERF, J.M.E., BRINKKEMPER, S., LAGO, P., BLOM, L., AND VAN VLIET, R. 2017. Awakening awareness on energy consumption in software engineering. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Society Track*, 76–85.

JOHANN, T., AND MAALEJ, W. 2015. Democratic mass participation of users in Requirements Engineering? In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, 256–261.

JOHNSON, M. 2015. *Sustainability Knowledge Management in Small and Medium-Sized Enterprises: Investigating the Effects of Sustainability Management Tools*. Dissertation, Leuphana Universität Lüneburg.

KERN, E., DICK, M., JOHANN, T., AND NAUMANN, S. 2011. Green Software and Green IT: An End User Perspective. In *Information Technologies in Environmental Engineering. Proceedings of the 5th International ICSC Symposium on*

Information Technologies in Environmental Engineering (ITEE 2011), P. GOLINSKA, M. FERTSCH AND J. MARX-GÓMEZ, Eds. Springer, Berlin, 199–211.

KERN, E., DICK, M., NAUMANN, S., GULDNER, A., AND JOHANN, T. 2013. Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In *ICT4S ICT for Sustainability*. *Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 87–94.

KHARCHENKO, V., AND ILLIASHENKO, O. 2016. Concepts of Green IT Engineering: Taxonomy, Principles and Implementation. In *Green IT Engineering: Concepts, Models, Complex Systems Architectures*, V. KHARCHENKO, Y. KONDRATENKO AND J. KACPRZYK, Eds. Springer, 3–19.

KOGELMAN, C.-A. 2011. CEPIS Green ICT Survey – Examining Green ICT Awareness in Organisations: Initial Findings. Carol-Ann Kogelman on behalf of the CEPIS Green ICT Task Force. *CEPIS UPGRADE XII*, 4, 6–10. http://www.cepis.org/upgrade/media/kogelman_2011_42.pdf.

LAGO, P., AND JANSEN, T. 2010. Creating environmental awareness in service oriented software engineering. In *International Conference on Service-Oriented Computing*, 181–186.

LAGO, P., KOÇAK, S.A., CRNKOVIC, I., AND PENZENSTADLER, B. 2015. Framing sustainability as a property of software quality. *Communications of the ACM 58*, 10, 70–78.

LAMI, G., AND BUGLIONE, L. 2012. Measuring Software Sustainability from a Process-Centric Perspective. In *Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012 Joint Conference of the 22nd International Workshop on*, 53–59.

LAMI, G., FABBRINI, F., AND FUSANI, M. 2013. A Methodology to Derive Sustainability Indicators for Software Development Projects. In *Proceedings of the 2013 International Conference on Software and System Process*. ACM, New York, NY, USA, 70–77.

LANG, D.J., WIEK, A., BERGMANN, M., STAUFFACHER, M., MARTENS, P., MOLL, P., SWILLING, M., AND THOMAS, C.J. 2012. Transdisciplinary research in sustainability science: practice, principles, and challenges. *Sustainability science 7*, 1, 25–43.

LIKERT, R. 1932. A technique for the measurement of attitudes. *Archives of psychology 22*, no. 140, pp. 1–55.

MAEVSKY, D.A., MAEVSKAYA, E.J., STETSUYK, E.D., AND SHAPA, L.N. 2017. Malicious Software Effect on the Mobile Devices Power Consumption. In *Green IT Engineering: Components, Networks and Systems Implementation*, V. KHARCHENKO, Y. KONDRATENKO AND J. KACPRZYK, Eds. Springer, 155–172.

MAHAUX, M., HEYMANS, P., AND SAVAL, G. 2011. Discovering Sustainability Requirements: An Experience Report. In *Requirements Engineering: Foundation for Software Quality*. *17th Intern. Working Conference, REFSQ 2011, Essen, Germany, March 28-30, 2011*, Proceedings, D. M. BERRY AND X. FRANCH, Eds. Springer, Berlin, Heidelberg, 19–33.

MAHMOUD, S.S., AND AHMAD, I. 2013. A Green Model for Sustainable Software Engineering 2013. *International Journal of Software Engineering and Its Applications 7*, 4, 55–74. http://www.sersc.org/journals/IJSEIA/vol7_no4_2013/5.pdf.

MALMODIN, J., AND LUNDÉN, D. 2016. The energy and carbon footprint of the ICT and E&M sector in Sweden 1990-2015 and beyond. *ICT4S-16*, 209-218. http://www.atlantis-press.com/php/download_paper.php?id=25860385.

MANOTAS, I., BIRD, C., ZHANG, R., SHEPHERD, D., JASPAN, C., SADOWSKI, C., POLLOCK, L., AND CLAUSE, J. 2016. An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of the 38th International Conference on Software Engineering Companion*, 237–248.

MAZIJN, B., DOOM, R., PEETERS, H., VANHOUTTE, G., SPILLEMAECKERS, S., TAVERNIERS, L., LAVRYSEN, L., AND VAN DUQUE RIVERA, J. 2004. *Ecological, Social and Economic Aspects of Integrated Product Policy. Integrated Product Assessment and the Development of the Label 'Sustainable Development´ for Products.* CP/20. SPSD II - Part I - Sustainable production and consumption patterns.

MOBJÖRK, M. 2010. Consulting versus participatory transdisciplinarity: A refined classification of transdisciplinary research. *Futures 42*, 8, 866–873.

MOCIGEMBA, D. 2006. Sustainable Computing. *Poiesis & Praxis: International Journal of Technology Assessment and Ethics of Science 4*, 3, 163–184. http://springerlink.com/content/uuu134v142p0g521/?p=0ee983bcf4e34a9d886563d1a9a90172&pi=1.

MOHANKUMAR, M., AND ANAND, M.K. 2015. A Green IT Star Model Approach for Software Development Life Cycle. *International Journal of Advanced Technology in Engineering and Science 3*, 1, 548–559.

NAUMANN, S. 2008. Sustainability Informatics – A new Subfield of Applied Informatics? In *EnviroInfo 2008: Environmental Informatics and Industrial Ecology*. *proceedings of the 22nd International Conference Environmental Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management, September 10 - 12, 2008, Leuphana University Lueneburg, Germany*, A. MÖLLER, B. PAGE AND M. SCHREIBER, Eds. Shaker, Aachen, 384–389.

NAUMANN, S., DICK, M., KERN, E., AND JOHANN, T. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *SUSCOM 1*, 4, 294–304. doi:10.1016/j.suscom.2011.06.004.

OECD. 2015. *OECD Digital Economy Outlook 2015*. http://dx.doi.org/10.1787/9789264232440-en.

PANG, C., HINDLE, A., ADAMS, B., AND HASSAN, A.E. 2016. What do programmers know about software energy consumption? *IEEE Software 33*, 3, 83–89.

PENZENSTADLER, B. 2013a. Towards a Definition of Sustainability in and for Software Engineering. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 1183–1185.

PENZENSTADLER, B. 2013b. What does Sustainability mean in and for Software Engineering? In *ICT4S ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich.

PENZENSTADLER, B., BAUER, V., CALERO, C., AND FRANCH, X. 2012. Sustainability in software engineering: A systematic literature review.

PENZENSTADLER, B., AND FEMMER, H. 2012. A generic model for sustainability. *Technical Report, Technische Universität München*. https://mediatum.ub.tum.de/attfile/1121449/hd2/incoming/2012-Nov/561736.pdf.

PENZENSTADLER, B., AND FEMMER, H. 2013. A Generic Model for Sustainability with Process- and Product-specific Instances. In *Proceedings of the 2013 workshop on Green in/by software engineering*, S. MALAKUTI, Ed. ACM, [S.l.], 3–8.

PENZENSTADLER, B., FEMMER, H., AND RICHARDSON, D. 2013. Who is the advocate? stakeholders for sustainability. In *Green and Sustainable Software (GREENS). 2nd International Workshop on Green and Sustainable Software*, 70–77.

PENZENSTADLER, B., RATURI, A., RICHARDSON, D., CALERO, C., FEMMER, H., AND FRANCH, X. 2014a. *Systematic Mapping Study on Software Engineering for Sustainability (SE4S) - Protocol and Results* ICS2 221. Institute for Software Research, University of California, Irvine, Irvine.

PENZENSTADLER, B., RATURI, A., RICHARDSON, D., AND TOMLINSON, B. 2014b. Safety, security, now sustainability: the non-functional requirement for the 21st century.

RAHBAR, E., AND ABDUL WAHID, N. 2011. Investigation of green marketing tools' effect on consumers' purchase behavior. *Business strategy series 12*, 2, 73–83.

RAL GGMBH. 2014. *Basic Criteria for Award of the Environmental Label: Computers*, RAL-UZ 78a. https://produktinfo.blauer-engel.de/uploads/raluz_uz/e-UZ-078a.zip.

RAMACHANDIRAN, C.R. 2012. Green ICT practices among tertiary students: A case study. In *Business, Engineering and Industrial Applications (ISBEIA), 2012 IEEE Symposium on*, 196–201.

RAZAVIAN, M., PROCACCIANTI, G., TAMBURRI, D.A., AND OTHERS. 2014. Four-Dimensional Sustainable E-Services. *EnviroInfo, Sep*.

SAN MURUGESAN. 2008. Harnessing Green IT: Principles and Practices. In *IT Professional*, 24–33.

SCHMIDT, B. 2016. Sustainability Knowledge about Software Parts in Software Engineering Processes.

SELYAMANI, S., AND AHMAD, N. 2015. Green Computing: The Overview of Awareness, Practices and Responsibility Among Students in Higher Education Institutes. *Journal of Information Systems Research and Innovation*. https://seminar.utmspace.edu.my/jisri/download/Vol9_3/JISRI_dec15_P4_Asnita.pdf.

SHARMA, S.K., GUPTA, P.K., AND MALEKIAN, R. 2015. Energy efficient software development life cycle-An approach towards smart computing. In *Computer Graphics, Vision and Information Security (CGVIS), 2015 IEEE International Conference on*, 1–5.

SHENOY, S.S., AND EERATTA, R. 2011. Green software development model: An approach towards sustainable software development. In *India Conference (INDICON), 2011 Annual IEEE*, 1–6.

SOUZA, M.R. de, HAINES, R., AND JAY, C. 2014. Defining sustainability through developers eyes. Recommendations from an interview study. In *Proceedings of the 2nd Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE 2014)*.

SUSTAINABLE BUSINESS ASSOCIATES. 2006. *Environmental Labelling. An Overview*. http://www.sba-int.ch/spec/sba/download/Publications%20principales/environmental%20labelling.pdf.

TAINA, J. 2010. How Green Is Your Software? In *Software Business. First International Conference, ICSOB 2010, Jyväskylä, Finland, June 21-23, 2010. Proceedings*, P. TYRVÄINEN, M. A. CUSUMANO AND S. JANSEN, Eds. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 151–162.

TAINA, J. 2011. Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In *Green ICT: Trends and Challenges*, J.-C. LOPEZ-LOPEZ, G. SISSA AND L. NATVIG, Eds., 22–27.

TCO DEVELOPMENT. 2015. *TCO Certified Desktops 5.0*. http://tcodevelopment.com/files/2015/12/TCO-Certified-Desktops-5.0.pdf.

TISCHNER, U., DIETZ, B., MAßELTER, S., SCHMINCKE, E., PRÖSLER, M., RUBIK, F., AND HIRSCHL, B. 2000. *How to do EcoDesign? A guide for environmentally and economically sound design*. Verlag form, Frankfurt am Main.

TORRE, D., PROCACCIANTI, G., FUCCI, D., LUTOVAC, S., AND SCANNIELLO, G. 2017. On the presence of green and sustainable software engineering in higher education curricula. In *Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials*, 54–60.

UNITED NATIONS. 1997. *Glossary of Environment Statistics, Studies in Methods*. United Nations New York, NY.

United Nations General Assembly. 1987. *Report of the World Commission on Environment and Development. Our common future*. UN document no. A/42/427 English, New York.

VAN HEDDEGHEM, W., LAMBERT, S., LANNOO, B., COLLE, D., PICKAVET, M., AND DEMEESTER, P. 2014. Trends in worldwide ICT electricity consumption from 2007 to 2012. *Computer Communications 50*, 64–76. https://biblio.ugent.be/publication/5782416/file/5782424.pdf.

VENKATESH, V., AND DAVIS, F.D. 2000. A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management science 46*, 2, 186–204.

VERDECCHIA, R., RICCHIUTI, F., HANKEL, A., LAGO, P., AND PROCACCIANTI, G. 2017. Green ICT Research and Challenges. In *Advances and New Trends in Environmental Informatics. Stability, Continuity, Innovation*, V. WOHLGEMUTH, F. FUCHS-KITTOWSKI AND J. WITTMANN, Eds. Springer, 37–48.

WIEDMANN, T., AND MINX, J. 2007. A definition of'carbon footprint'. *Ecological economics research trends 2*, 55–65.

WILLIAMS, D. 2013. *GHG Protocol Product Life Cycle Accounting and Reporting Standard ICT Sector Guidance. Chapter 7: 7 Guide for assessing GHG emissions related to software*. http://www.ghgprotocol.org/files/ghgp/GHGP-ICT-Software-v2-9-26JAN2013.pdf. Accessed 6 February 2013.

ZAPICO, J.L., TURPEINEN, M., AND BRANDT, N. 2010. Greenalytics: a tool for mash-up life cycle assessment of websites. In *EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany*, K. GREVE AND A. B. CREMERS, Eds. Shaker, Aachen, 754–763.

Declaration on the author contributions

The following tables list the papers included in this cumulative doctoral thesis and provide information on authors' contributions to the articles as well as their publication status (24.4.2018).

Article #	Title	Authors	Author status	Weighting factor	Publication status	Conference contribution
1	Impacts of software and its engineering on the carbon footprint of ICT	EK, MD, SN, TH	Co-author with equal contribution	1.0	Published in: Environmental Impact Assessment Review 52, pp. 53–61. 2014. http://dx.doi.org/10.1016/j.eiar.2014.07.003 (JCR=3.094, SJR = 1.37 (coverage: 1980-1983, 1985-ongoing))	EnviroInfo 2013
2	Sustainable software products – towards assessment criteria for resource and energy efficiency	EK, LH, AG, YM, AF, JG, SN	Co-author with equal contribution	1.0	Accepted for publication in: Future Generation Computer Systems [in press] Article status: https://authors.elsevier.com/tracking/article/details.do?aid=4079&jid=FUTURE&surname=Kern (JCR=3.997, SJR=1.15 (coverage: 1984-ongoing))	---
3	Environmental issues of software and its labelling: an end user perspective	EK	Single author	1.0	Submitted to Sustainable Computing: Informatics and Systems on 14.08.17 (JCR=1.800, SJR=0.36 (coverage: 2011-ongoing))	---
4	Communicating Environmental Issues of Software: Outline of an Acceptance Model	EK	Single author	1.0	Published in: Wohlgemuth, V., Fuchs-Kittowski, F., and Wittmann, J. (Eds.): Advances and New Trends in Environmental Informatics, Springer International Publishing, pp. 335-345. 2016.	EnviroInfo 2016*
5	Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues	EK, AG, SN	Co-author with predominant contribution	1.0	Accepted for publication in: Kharchenko, V.; Kondratenko, Y.; Kacprzyk; J. (Eds.): Green IT Engineering: Social, Business and Industrial Applications, Volume 3 in the book series "Studies in Systems, Decision and Control" [in press]	---

JCR = InCites Journal Citation Reports, SJR =SCImago Journal Rank

Authors' contributions to the articles

Contribution	Paper 1	Paper 2	Paper 3
Concept of the research approach	EK, MD	LH, SN, EK, YM, AF	EK
Literature research and review	EK	EK	EK
Development of the calculation method	EK, MD, TH	---	---
Development of the model approach	---	LH	---
Elaboration of the set of criteria	---	LH, SN, EK, YM, AF	EK, AG, SN
Application and evaluation of the criteria	---	AG	AG, EK
Development of the case studies	TH	EK	---
Writing of the manuscript	EK, MD, TH	EK, LH, AG, YM, SN	EK, AG, SN
Revision of the manuscript	EK, MD, TH, SN	EK, LH, AG, YM, SN, AF, JG	EK, AG, SN

EK= Eva Kern[1,2]

MD = Markus Dick[2#]

SN = Stefan Naumann[2]

TH = Tim Hiller[2#]

LH = Lorenz M. Hilty[3,4,5]

AG = Achim Guldner[2]

YM = Yuliyan V. Maksimov[3,6]

AF = Andreas Filler[2,7]

JG = Jens Gröger[8]

[1] Leuphana University Lueneburg, Universitätsallee 1, 21335 Lueneburg, Germany

[2] Institute for Software Systems in Business, Environment, and Administration, Trier University of Applied Sciences, Environmental Campus Birkenfeld, P.O. Box 1380, 55761 Birkenfeld, Germany; [#] former employees

[3] Department of Informatics, University of Zurich, Zurich, Switzerland

[4] Technology and Society Lab, Empa Swiss Federal Laboratories for Materials Science and Technology, St. Gallen, Switzerland

[5] KTH Royal Institute of Technology, Stockholm, Sweden

[6] i4Ds Centre for Requirements Engineering, University of Applied Sciences Northwestern Switzerland (FHNW), Windisch, Switzerland

[7] Institute of Technology Management, University of St. Gallen, Dufourstrasse 40a, 9000 St. Gallen, Switzerland

[8] Sustainable Products & Material Flows Division, Oeko-Institut, Schicklerstraße 5-7, 10179 Berlin, Germany

Conferences

EnviroInfo 2013 — 27th International Conference on Informatics for Environmental Protection, 02. – 04. September 2013, Hamburg (Germany), http://enviroinfo.eu/de/events/conference

EnviroInfo 2016 — 30th International Conference on Informatics for Environmental Protection, 14. – 16. September 2016, Berlin (Germany), http://enviroinfo2016.org; *3rd Prize Best Paper Award

Papers included in this cumulative doctoral thesis

Paper 1: Kern, Eva; Dick, Markus; Naumann, Stefan; Hiller, Tim. 2014. Impacts of software and its engineering on the carbon footprint of ICT.

Paper 2: Kern, Eva; Hilty, Lorenz M.; Guldner, Achim; Maksimov, Yuliyan V.; Filler, Andreas; Gröger, Jens; Naumann, Stefan. 2018. Sustainable software products – towards assessment criteria for resource and energy efficiency.

Paper 3: Kern, Eva. Environmental issues of software & its labelling: an end user perspective.

Paper 4: Kern, Eva. 2016. Communicating Environmental Issues of Software: Outline of an Acceptance Model.

Paper 5: Kern, Eva; Guldner, Achim; Naumann, Stefan. 2018. Including Software Aspects in Green IT: How to Create Awareness for Green Software Issues.

The following publications are not part of the formal requirements of this doctoral thesis, but they support the argumentation.

Paper A: Kern, Eva; Dick, Markus; Naumann, Stefan; Filler, Andreas. 2015. Labelling sustainable software products and websites: Ideas, Approaches, and Challenges.

Paper B: Kern, Eva. 2017. Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges.

Impacts of Software and its Engineering on the Carbon Footprint of ICT

– Paper 1 –

Eva Kern[a], Markus Dick[b], Stefan Naumann[a], Tim Hiller[a]

[a] Institute for Software Systems, Environmental Campus Birkenfeld, Campusallee, D-55761 Birkenfeld e.kern@umwelt-campus.de, s.naumann@umwelt-campus.de, tim.hiller@gmx.com

[b] Fritz-Wunderlich-Straße 14, D-66869 Kusel, sustainablesoftwareblog@gmail.com, http://sustainablesoftware.blogspot.de/

This article is an extended and revised version of the paper "Integrating Aspects of Carbon Footprints and Continuous Energy Efficiency Measurements into Green and Sustainable Software Engineering." In: Page, Bernd; Fleischer, Andreas; Göbel, Johannes; Wohlgemuth, Volker (eds.): EnviroInfo 2013 - Environmental Informatics and Renewable Energies. 27th International Conference on Informatics for Environmental Protection. Proceedings of the 27th EnviroInfo 2013 Conference, Hamburg, Germany, September 2-4, 2013. Aachen: 2013, pp. 300-308. TOC online available http://www.shaker.de/de/content/catalogue/index.asp?lang=de&ID=8&ISBN=978-3-8440-1676-5

Abstract. The energy consumption of information and communication technology (ICT) is still increasing. Even though several solutions regarding the hardware side of Green IT exist, the software contribution to Green IT is not well investigated. The carbon footprint is one way to rate the environmental impacts of ICT. Overall, the ICT is responsible for about 600 million tons of CO_2 emissions worldwide per year [Gartner, Inc. 2007]. In order to get an impression of the induced CO_2 emissions of software, we will present a calculation method for the carbon footprint of a software product over its life cycle. We will also offer an approach how to integrate some aspects of carbon footprint calculation into software development processes and discuss impacts and tools regarding this calculation method. We thus show the relevance of energy measurements and the impacts on the carbon footprint by software within green software engineering.

Keywords. Green Software Development; Carbon Footprint; Energy Efficiency Measurements

1 Introduction

In the last few years environmentally sound "green" ICT hardware product design and production, as well as green IT service operation gained a lot of importance. Software as the ultimate cause of hardware requirements and energy consumption shifts only slowly into focus. In general, the software

aspect is insufficiently studied. While there are approaches how to save energy in the context of hardware that are capable to be used in the industry, such solutions are missing on the software side. Therefore, creating awareness and transparency for the topic should be the first step. Indeed, standard techniques to assess the environmental impacts of products and services, like the calculation of a full-fledged carbon footprint (CF) or life cycle assessments, are complicated and laborious. If they are carried out at all, then often by external experts or after the product had already been released to the market. In order to calculate the CF and therefore measuring the impact of ICT and its production on climate change, it is necessary to identify which parts of the production and usage process are particularly suitable for using renewable energies. Of course, ideally the whole production and usage of ICT is driven by renewable energies. But, in practice, this is not the case. So models and tools dealing with questions about carbon footprint and energy consumption of ICT help to make decisions about changes to renewable energies.

In order to solve the problems of missing solutions for energy aspects of software and standard techniques in this context, according to Braungart and McDonough [2009] and Kramer [2012], it is necessary to integrate the consideration of sustainability aspects as early as possible into design processes. Late changes to the product design are much more expensive than early ones, which is also commonly known from software projects.

Facing these challenges, our paper proposes a method and a strategy that complement sustainable software processes that was e.g. presented by Dick and Naumann [2010] and Lami et al. [2012]. The calculation method for the carbon footprint allows software development teams to take over the responsibility for some decisions regarding the development process itself (necessity of meetings, business trips), as well as design decisions that have a direct impact on the environmental hardware requirements and also on some social impacts of their software products. That means based on the current CF of the software, the software team can decide if e.g. a business trip is necessary or if it would be better to try to exchange ideas during a conference call.

The proposed method (carbon footprint calculating) picks up the principle ideas of product carbon footprints (PCF), focusing on the ongoing software development process and claiming to be conductible by software developers or process managers. To calculate the CF, we considered the number of employees as well as their number of working days, IT infrastructure and how they commute, the overall IT infrastructure of the enterprise, and the office space. The resulting CF is given in kg CO2 equivalents. Thus, the method does not consider impacts of hard to estimate upstream chains, e.g. third-party software libraries that are indeed relevant for a PCF, but not for the CF of the development process itself. We also try to answer the question if emissions of commuting employees should be considered in PCFs or not, as standards require [British Standards Institution 2011]. Furthermore, we contrast the exemplary CF of development with an oversimplified CF of computer usage, not for generalization or proof, but to give an impression of the magnitudes.

The proposed strategy (continuous energy efficiency measurement) makes use of continuous integration (CI), to integrate ongoing measurements of the energy efficiency of the software product into the software development process. CI is a widely used approach in software engineering. In conjunction with the test driven development approach, the main goals of CI are to minimize the integration effort and to maximize the feedback to software developers, focusing on software quality. Hence, both approaches are widespread and well approved in practice. In contrast, measuring software-induced energy consumption and rating its energy efficiency as a quality aspect, apart from scientific research and embedded systems, is not really taken into account during software development. Today, there are no practical suggestions on how to monitor and improve software energy efficiency during development.

2 Related Work

2.1 ICT, Renewable Energy, and Environment

Especially the aspect of "Green by IT" plays a significant role in the relation of ICT and renewable energies: ICT solutions have the ability to support and even enable the use of renewable energies by approaches like smart heating, smart lighting, smart grids, etc. Different approaches exist on how to develop smart systems and how to improve them. Rohjans et al. [2012] discuss the necessity of requirements engineering for ICT architectures in order to build up smart grids. They point out that use case development as well as methodologies and specifications known from requirements engineering need to be a part of smart grid planning and implementation. Apart from that there are dependencies and related components in smart grids that can be improved by ICT [Wäfler and Heegaard 2013]. Hence, according to Wäfler and Heegaard [2013], ICT components and services can support the operation of smart grids since monitoring and controlling are fundamental services in that context.

Moreover, ICT can support the production of renewable energies directly by enabling plants like wind farms, biomass combustion, and solar panels. One example are simulation tools used for planning, implementing, and maintaining offshore wind farms [Joschko et al. 2013]. Further ICT applications within the context of environmental challenges are early warning systems [Alfieri et al. 2012; Li et al. 2013] or rather environmental global information systems.

Beside from that aspect, dealing with renewable energies is also an important aspect in the context of data centers. Even the Blue Angel for energy-conscious data centers argues for environmentally friendly sustainable energy sources like renewable energies [RAL gGmbH 2012]. The Blue Angel is a German certification for environmentally friendly products and services. As a metric for green data centers, Belady et al. [2010] define the Carbon Usage Effectiveness (CUE) as annual CO2 emissions by total datacenter energy per IT equipment energy. Deng et al. [2013] deal

with opportunities and challenges of renewable energies in data centers. They point out the advantage of reducing energy consumption and power costs as well as the resulting carbon footprint. They also consider the aspect of some problems of renewable energies such as dependency on weather conditions. Based on an analysis of different existing and potential approaches and research activities, they try to find a solution for the sensible operating of data centers with renewable energies. Indeed, they also affirm the necessity of research in this field.

Another approach of improving the energy efficiency and therewith the ecological friendliness of data centers might be cloud computing. A variety of research work focuses on the relationship between cloud computing and energy efficiency or rather Green IT [Gong et al. 2013; Jing et al. 2013; Kaur et al. 2013]. In this context, Cook and Van Horn [2011] describe Green IT as the sum of energy efficiency and renewable energy.

2.2 Green and Sustainable Software and its Energy Efficiency

Even if sustainable product design is still focusing on the hardware side, there are some approaches regarding the sustainability of software processes: they rank from interpreting and introducing metrics to measure the sustainability of software [Albertao 2004], over concepts concerning software energy consumption [Käfer 2009] to a software project with sustainability requirements as regular software quality requirements [Mahaux et al. 2011].

Different models exist regarding green, sustainable and environmentally friendly software. All of them address the energy aspect and try to find ways to reduce the impacts of software products on the environment. The so called GREENSOFT Model as a reference model for green and sustainable software and its engineering comprises a lifecycle model for software products, sustainability criteria and metrics of software products, a "Green Software Engineering" procedure model, and recommendations for actions as well as tools [Naumann et al. 2011]. Here, Naumann et al. propose to mitigate carbon dioxide emissions that result from development activities by the development process itself. Based on that model, Mahmoud and Ahmad [2013] present a Green Model for Sustainable Software Engineering. This model covers two levels: firstly, the Green Software Engineering Process and secondly, Software that promotes Green ICT. In order to reduce the negative impacts of ICT, e.g. the energy consumption, the model aims to have a full view of how software affects the environment. Shenoy and Eeratta [2011] focus on practices to develop software in an environmentally friendly way. The Green Software Development Life Cycle Model dwells the phases of requirements, designing, implementation, testing, deployment, maintenance, retirement, and supporting processes like quality and infrastructure. The approach is based on the idea that efficient software will indirectly consume less energy by using up less hardware equipment to run.

In order to get data of the energy consumption of software, measurement methods are necessary. There have already been developed several measurement methods and metrics developed to rate the

energy consumption of software. Dick et al. [2011] propose to measure energy consumption of desktop applications during the execution of realistic usage scenarios and load tests for server applications. They take a look at the software from the outside, without knowledge about the implementation details. While this could be the right approach to compare the behavior of standard software products operating in identical domains, the method seems to be inapplicable to measure and rate energy efficiency during the design and development process. Hindle [2012] shows that power consumption potentially varies between different versions and suggests measuring power consumption for each revision to keep up with changes and make developers more sensible about the energy consumption during development. Johann et al. [2012] introduce a method to rate energy efficiency by source code instrumentation. They suggest a definition of energy efficiency in relation to software as the ratio between useful work done and used energy and mention that each software type requires its own metric to rate its energy efficiency. Schubert et al. [2012] developed a software energy profiler to estimate energy consumption without the need of manual source code instrumentation or using metering devices. With these approaches, it becomes tangible to make design decisions affecting the energy consumption while writing the code.

2.3 Carbon Footprint

The PCF ranks among the most discussed indicators for the human and industrial impact on the environment. Recently, there are many publications available discussing the CF of different goods [Wackernagel and Yount 1998; PCF Pilot Project Germany 2009; Gombiner 2011], methods of calculating the CF [International Organization for Standardization 2014], and the representative status of the CF [Galli et al. 2012; Laurent et al. 2012]. Although there are some arguments against the CF as representative for the ecological impacts of goods, we deem the CF a sensible possibility to gather the impact on the environment. In view of this fact, it should also be calculated for the software engineering process, in order to include it into the software life cycle, e.g. proposed in [Naumann et al. 2011]. Vickery and Mickoleit [2013] list four effects of ICT products on other product's environmental footprint: optimization (ICT reduces the environmental impact of other products), dematerialization and substitution (replacing physical products and processes with solutions that have a smaller footprint by ICT), induction effects (if ICT products increase demand for other products) and degradation (more difficult waste management caused by ICT devices embedded in other products).

Overall, the calls for standardization in this context are growing ever louder. Until now, there are some initiatives for standards (PAS2050 [British Standards Institution 2011], Corporate Value Chain [Bhatia et al. 2011] and the draft for ISO 14067 [International Organization for Standardization]). ISO14067 is based on ISO14040 et seq., dealing with ecological audits, and addresses framework conditions, criteria, and a strategy to calculate the CF of products in general. In contrast to that, the Greenhouse Gas Protocol (www.ghgprotocol.org) takes up the emissions caused by the ICT sector, e.g. regarding communication, networks and hardware, as well as software. The calculation of the

energy consumption of software concentrates on the usage phase [Williams 2013]. A first approach of the CF of software products is presented by Taina [2010]. For this purpose, he develops metrics and applies these to an artificial project. The software engineering process is separated into different phases. Based on this, different methods are proposed to calculate the CF of each phase.

Regarding CF and ICT different foci come to mind: on the one hand, one can concentrate on the CF of ICT itself. In the following contribution we will do so, concentrating on the software side. On the other hand, there are solutions how ICT can help to gather and calculate the CF of hardware advices. So to say, it is "Carbon Footprinting of Software" and "Carbon Footprinting by Software". CEMIS (Corporate Environmental Management Information System), a tool developed in the IT-for-Green-Project, should e.g. support environmental management workflows according to standards and guidelines like ISO 14040 and ISO 16128. It is a stakeholder-oriented tool for enterprises [Rapp and Bremer 2013]. On the other hand, there are approaches to record and monitor energy consumption of enterprises which can be a base to calculate the CF of those (e.g. The “Green IT Cockpit” [Opitz et al. 2014])

3 Integrating Aspects of Carbon Footprints

Our CF approach is based on the suggestions by Taina [2010] and will be oriented towards the guidelines of International Standard ISO/DIS 14067.2 [International Organization for Standardization 2014], Publicly Available Specification PAS 2050 [2011], and the GHG Protocol [Williams 2013].

Table 1 *Factors considered for the Carbon Footprint Calculation*

Factor	Issue
Employees	Commuting
Office space	Heating, ventilation, and air conditioning (HVAC)
ICT infrastructure	PCs, workstations
	Backup storage system
	Rack mount servers
	Server administration
	Uninterruptible power supply (UPS)

In order to evaluate the proposed CF study and to get a first impression of the CF of software products during their whole life cycle, we conducted two case studies in micro-enterprises located in Germany and Luxembourg. The studies were based on retrospectively collected data and additional

information of project team members. Afterwards, we used the collected experience to compile a more general example. Here, we have a closer look at the development phase of software and the support and maintenance efforts after the software has been released. We considered the factors that are given in Table 1 for the CF calculation.

The example is especially designed to answer the question whether or not the impact of commuting employees should be considered in CFs of software. Standards require that those emissions are not considered [British Standards Institution 2011]. Unfortunately, using public transport or dematerializing transport, e.g. by means of telework, has the potential to reduce emissions and to consecutively change working and living conditions. Therefore, we give an impression of the magnitudes of emissions of commuting employees depending on the distance.

Even if impacts of the usage phase are not of high importance for this artificial example, due to the fact that there may be too many assumptions regarding usage scenarios, deployment environments, data transfer, etc., we give two oversimplified emission series for an increasing amount of PCs and servers for comparison reasons.

3.1 The Example Micro-Enterprise

As an example company for the CF calculation, we chose a company size of nine employees. Approx. 75% of the companies in the information and communication sector in Germany [Statistik der Bundesagentur für Arbeit 2013] belong to the category of up to nine employees. According to European regulations, companies of up to nine employees are called micro-enterprises [European Commission 2005].

We assume that the employees hold the following positions: one general manager (the owner itself), one accountant, one person for sales and one for customer support. Above that, there are five software developers. To calculate the floor area of the corporation, we assume an office space ratio of 30.5 m² per employee in a team office [Jones Lang LaSalle 2009]. The office space ratio partially includes corridors, meeting rooms, the reception area, etc. The gross floor area of 343.1 m² is assumed by dividing the net floor area of 274.5m² by 0.8 [Energieinstitut der Wirtschaft 2010]. We assume that besides the workstations of the employees, the company operates four rack mount servers, a backup storage system, and a class 2 line-interactive UPS with an efficiency between 95 and 98% [Liebert 2000]. For server administration, they operate an additional workstation. All these systems are assumed to be operated in 24/7 mode.

Regarding the question whether or not the impacts from commuting employees should be considered in CFs of software, we have to assume the distances and the means of transport of the employees. The basis of this assumption forms a report of the German Federal Statistical Office about the traffic behavior of commuters in Germany [Grau 2009]. The given distributions were applied to nine employees and rounded appropriately. For the bus and train distances, the ratio of employees was

nearly equally distributed between the "10-24 km" and the "<10 km" distance categories, so we assigned the remaining two employees according to our own preference (longer distance by train, shorter distance by bus). Table 2 shows the resulting distribution of the base scenario.

Table 2 *Base scenario of employees' one-way commuting distances and means of transport*

One-way distance	by car	by train	by bus	by foot
≥ 50 km				
25 – 49 km	1			
10 – 24 km	2	1		
< 10 km	2		1	1
Same estate				1

3.2 Calculating the Carbon Footprint per Functional Unit

After defining the base data of the micro-enterprise, we are now ready to approximate the emissions per year in kg CO_2 equivalents (CO_2e). The gross floor area is approximated above with 343.1 m². The gross floor area related ratios which are used below were established for office buildings.

The energy ratios for heating and water heating are assumed on average with 111 kWh/m^2 per year [Energieinstitut der Wirtschaft 2010], which results in approx. 38,086 kWh energy for heating per year. We assume a natural gas-driven low-temperature boiler as the heating technology. Its average emission ratio is given with 0.3 kg CO_2/kWh [KfW 2003]. This results in 11,426 kg CO_2e per year for heating.

The energy ratios for electricity comprise lighting, building automation, electricity for operating the heating system, as well as basic IT Infrastructure and workstations. The average electricity consumption is given with 125 kWh/m^2 per year, which results in approx. 42,890 kWh/year.

The electricity for the server infrastructure is based on PCF studies and energy consumption information of a computer manufacturer [Fujitsu Technology Solutions 2010, 2011]. For the terminal, we assume the power consumption of a workstation in idle mode. It is given with 45 W, which results in 394 kWh/year. The power consumption of one server is given with 119 W (SPEC power benchmark at 30% workload), which, for all four servers, amounts to 4,169 kWh/year. We found no adequate data for the storage. Therefore, we assume the power consumption of the backup system with 100 W, which results in 876 kWh/year. The efficiency of the uninterruptible power supply (UPS) is between 95% and 98%. Therefore, we define its efficiency with 96%. The standard load in our scenario is approx. 877 W. However, the maximum load is approx. 900 W, which means that the usual load factor is only round about 70%. Therefore, we assume that the UPS runs with 96% efficiency, which results in UPS losses of 217 kWh/year. The total electricity consumption of the server infrastructure is approx. 5,263 kWh per year.

The above mentioned calculation aggregates to a total electricity consumption of 48,154 kWh per year. Assuming the standard electricity mix emission factor in Germany of 0.5656 kg CO_2e/kWh [Federal Environment Agency (Umweltbundesamt) 2011], this corresponds to 27,236 kg CO_2e deposition per year. In total, electricity consumption and heating are responsible for approx. **38,663 kg CO_2e** emissions per year.

To estimate the emissions of the commuting employees, we lay down the following factors per person: transport by car (one person per car) 0.25 kg CO_2e/km, commuter train 0.053 kg CO_2e/km, urban bus 0.019 kg CO_2e/km [Grabolle and Loitz 2007]. The number of work days per year is assumed to be 220 (considers statutory holidays, leave days, and sick days). Concerning the distance classes presented in Table 1, we assumed the mean distance of each class as the one-way distance. Applied to round-trips, we get a total of 9,234 kg CO_2e per year. According to PCF standards, these emissions are not within the system boundaries and are therefore not considered. If considered, the total emissions per year increase to **47,897 kg CO_2e**.

The functional unit kg CO_2e/PM is computed by dividing the total emissions per year by the number of person months per year that can be invoiced. Here we considered the developers (60 PM) and the support (12 PM). This results in approx. **537 kg CO_2e/PM** without and approx. **665 kg CO_2e/PM** with emissions from commuting employees.

3.3 Calculating the Carbon Footprint of the Example Project

The development phase of the example project is assumed to be 6 months long with a subsequent support and maintenance period of 5 years. Furthermore, we assume that the developers are working full-time on this project, which add up to 30 PM. Thus, the corresponding emissions of the development phase amount to 16,109 kg CO_2e (19,957 kg CO_2e incl. commuting).

To estimate the emissions from the support and maintenance period, we need some kind of an allocation approach to spread the total efforts on all hypothetical projects supported and maintained at a time. Here we simply assume a roll-out of two projects each year, with a fixed support and maintenance period of five years, which means that 10 projects are in the queue simultaneously. If we equally distribute the efforts between the projects, this results in 1.2 PM per project and year. Hence, for the support and maintenance, the overall emissions are then approx. 3,222 kg CO_2e (3,991 kg CO_2e incl. commuting).

Thus, the total emissions of the example project are approx. **19,331 kg CO_2e** or **23,948 kg CO_2e** with emissions from commuting.

4 Discussion on Impacts on the Carbon Footprint Calculation

Regarding the calculation of the CF of software products, there are different impacts we need to deal with. All of them influence the calculation result. In order to compare the CO_2 emissions for different software projects, the same conditions need to be ensured. That means for example that either the whole CO_2 emissions for commuting need to be integrated or otherwise no commuting impacts should be considered.

4.1 Impacts from Commuting

In the scenario assumed in Table 2, considering emission from commuting increases the CF of the project (incl. support period) by approx. 24%. For a deeper insight, we compared this base scenario with three further scenarios: in the first scenario, one car driver is switched from "10-24 km" to ">= 50 km"; in the second scenario, one more car from "<10 km" to "= 50 km"; in the third scenario, the usage of cars is eliminated by shifting the employees to the train column, except for "<10 km" who are shifted to the bus-column. The contribution of commuting to the annual CF of the corporation is shown in Figure 1.

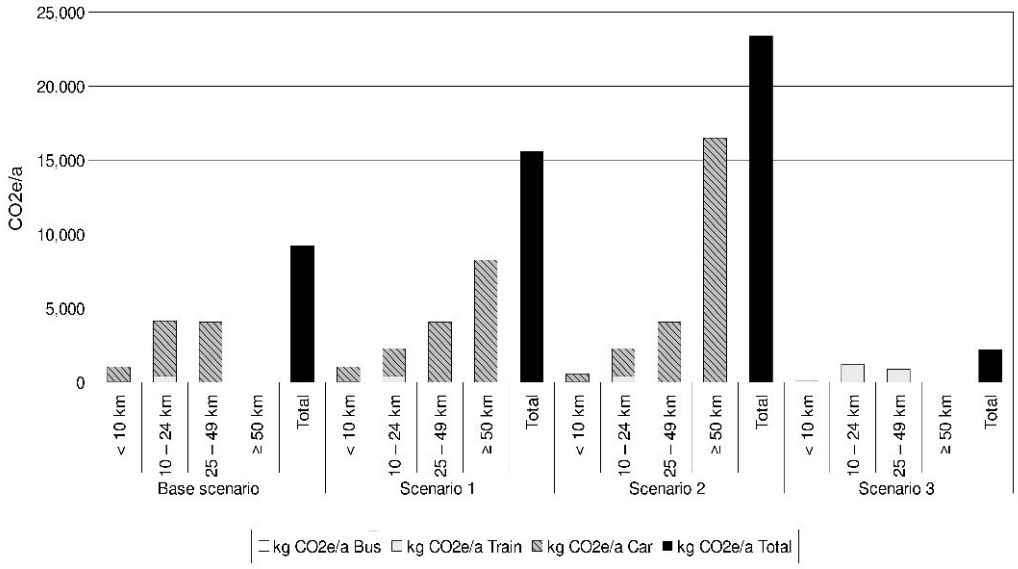

Figure 1 *Comparison of the carbon footprint of different commuting scenarios*

Comparing these scenarios shows that there are a lot of implications regarding the commute aspect. So far, there is no solution how the best scenario would look like, e.g., working remotely is not discussed yet. Commuting of the employees raises a lot of questions that are not addressed in this article.

4.2 Impacts from Operating

While having a closer look at the different impacts of the phases of the software lifecycle, one can notice that, depending on the kind of software, either the development or the usage phase is one of the main factors of the CF calculation. If the developed software is custom software, it is not possible to accumulate the CF on many sold copies of the product. This is the case since just a few products will be sold and hence there are just a small number of usage phases while consuming energy. Here, we posit that the product itself is slightly energy consuming. Additionally, this conclusion depends on the length of the usage phase and the required hardware (Is it necessary to buy new one or not?). Therefore, the dominating factor of the overall CF of custom software will be the development phase in most cases.

Considering standard software, there are many consumers using the software products (e.g. office solutions). In this case, it is sensible to accumulate the development phase on many sold copies of the software. Hence the dominating factor of the resulting CF might be the emissions of the usage phase. Certainly, in this case, there are many users with many different usage scenarios. Therefore, one has to estimate the different types of users, measure the energy consumption of each of the consequent scenarios, and calculate an average CF of the usage phase. In order to make the scenarios more comparable, it might be sensible to use definitions of software user types. Such kinds of typologies are especially available regarding Internet users [Pruulmann-Vengerfeldt 2006; Brandtzæg et al. 2011; Brauckmüller 2011].

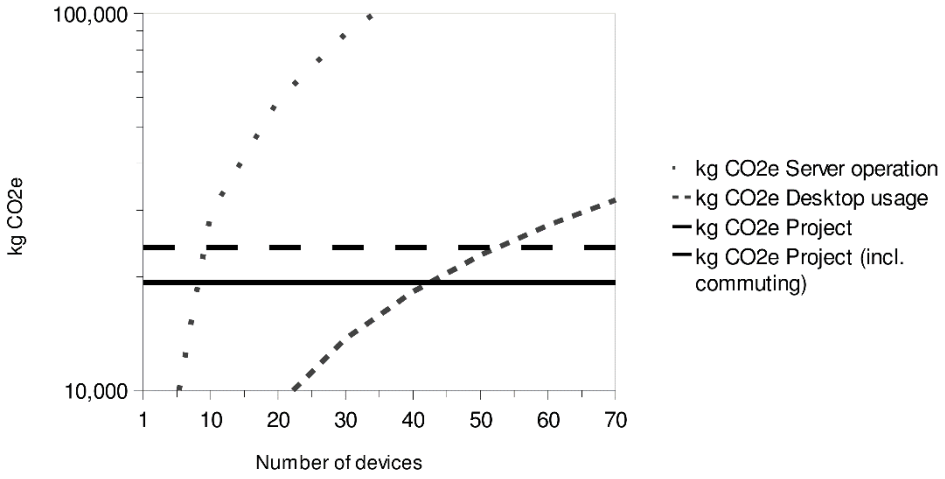

Figure 2 *Project carbon footprint vs. oversimplified usage carbon footprints*

The emissions of the usage phase of a software product are also of high importance. However, we decided to not consider the usage phase in our case study, because we lack basic knowledge about average custom software products, developed in micro-enterprises, e.g., types of software, expected workloads, number of users, or required number of servers. In spite of our concerns, we give an impression of the magnitude of the CF of the five-year usage phase compared to the CF of the

example project. We focused on the development phase. As a scale, we provide two oversimplified CFs of workstation and server usage, where the number of computers increases in increments of ten (Figure 2).

For desktop PCs (approx. 51W) we assume a usage time of 8h per day, 220 days per year, including 1h lunch break where the monitor is in stand-by mode (approx. 34W, 2W stand-by) but the PC is not, which results in approx. 807kWh or 457kg CO_2e per desktop PC during a usage period of 5 years.

For servers (approx. 119W), we assume 24/7 operation (365 days per year), which results in approx. 5,212kWh or 2,948 kg CO_2e per server during a usage period of 5 years.

For simplicity, the impacts of any further required IT infrastructure are not considered.

As calculated in section 3.3, the CF of the software project was approx. 19,331 kg CO_2e with or 23,948 kg CO_2e without commuting.

Comparing this to the energy consumption of PCs and servers during the usage phase of the software (Figure 2), it can be seen that it requires only seven to nine servers (regarding whether we consider commuting or not) to exceed the CF of the project but a lot more desktop PCs (43 to 53).

If we consider that the energy savings (in the sense of first-order effects) that are possible by improving the energy efficiency of a software product, cannot be below the idle power consumption of a computer, it is clear that we need a huge installation base to achieve significant effects, especially if we consider software for desktop PCs.

Hence, regarding custom software made by SMEs with an usually smaller installation base than custom or standard software made by global players, energy efficiency improvements of the algorithmic implementations may be of less importance than improving the production and maintenance processes. Furthermore, energy efficiency of server based software may be of higher importance than that of desktop based software.

However, server based software usually requires some kind of a client software accessing the server. Hence, even if a server software is developed, the energy consumption of the clients accessing the server may be of importance, even if we do not develop them by ourselves (consider, e.g., web applications that make heavy use of JavaScript, where the client is a standard web browser).

Concluding, the calculation in Figure 2 shows that the development phase is more important in case of a small number of installations, especially if server software was developed. However, since several software products are developed as individual software it shows that even the development process is relevant. But one can also conclude that the more installations are expected the more useful it is to invest more energy and time into the development process in order to produce a “greener” software product.

4.3 Impacts from the Used Resources

Overall, the energy consumption, in our example project 86.239 kWh per year, contributes a lot to the resulting CF. It includes the electricity for the server infrastructure and the additional consumption for the workstations, lighting, etc. (cf. section 3.2) during the development phase. In this context, three aspects are to be considered in order to improve the carbon emissions. These are (1) the amount of energy used, (2) the kind of energy used, and (3) the time of consuming energy. The first point is mainly self-explaining: one should try to waste as less energy as possible. Secondly, while trying to keep the CF of software as low as possible, one has to be aware of the fact that the amount of resulting CO_2e depends on the kind of energy used and its conversion. The type of energy also influences the CF. According to Spadaro et al. [2000] renewable energy sources, like hydro, wind, and solar, have a better CO_2 rate, cf. amount of CO_2 emissions over its lifecycle, than fossil fuels such as coal and oil. While trying to find possibilities to optimize the CF of the development phase one should also take into account the possibility of execute compiling or huge automated tests while there is a surplus of energy, e.g. during the night or during noon.

Besides the energy consumption of the development phase, operating the software consumes energy. This means that the above mentioned aspects ((1) amount and (2) kind of energy as well as (3) time of energy consumption) could also influence the resulting CF of the usage phase.

5 General Discussion on Aspects influencing the Carbon Footprint

While having a closer look at an automatically generated code, it is common experience that this kind of code is less efficient than a code written by an experienced developer [Capra et al. 2011]. That means, if code is generated automatically, the CF of the development phase is usually lower (less time and as a result less energy consumption) but the CF of the usage phase will increase (less efficient code and as a result more energy consumption). Thus, one has to decide whether the usage of automatically generated code reduces the overall CF or not. Therefore a detailed analysis and a projection are necessary.

According to a survey on the participation of corporations in open source projects (based on projects hosted on SourceForge.net) by Capra et al. [2009], companies have a significant role even in those projects that are commonly believed to be mostly based on the work of volunteers. Such being the case, this kind of software development cannot be disregarded within the calculation of software's CF, even if the focus is primarily on community projects.

Typical characteristics of open source projects are that many people participate in the process of coding. These programmers can be dependent or independent of a company and might be separated by each other or not. Because of this, it is hard to log each activity concerning one project to estimate its

CF as precisely as possible. A calculation is only possible by using assumptions. These assumptions make the calculation more inaccurate and less comparable than closed source projects by one company or a specific group of coders which can be measured in detail.

Additionally, one is also called to use as less energy as possible and / or use renewable energies to reduce the resulting CO_2 emissions. In order to influence the time of the energy consumption, the characteristic of the software and its architecture is relevant since it is linked to the resulting energy consumption and the possibility to use renewable energies: The architecture makes it possible to defer the wattage to electricity tariff at a reduced rate or at the standard price. If the energy consumption of software is temporal and geographically adaptable, that means if the load transfer can be adapted by software, it is possible to match the energy use with the availability of renewable energies. In the case that it is not possible to use renewable energies locally, it might be a solution to spare own server and host server in green data centers. Indeed, before doing that it is necessary to calculate the CF of both scenarios: savings of CO_2 emissions by using external server vs. CF of the data transfer to these servers. Indeed, you might stick with your own server and operate them with environmentally friendly power instead of building up a green data center that might be more energy efficient.

6 Supporting Tools for the Carbon Footprint Calculation

Since there are many data items to collect during the software development process, it should be as easy as possible to gather the relevant data. Hence, we will describe some tools that can support the collection of the required data.

6.1 Continuous Energy Efficiency Measurements

In order to be able to measure the expected first-order impacts of a software product, developers should be enabled to analyze and rate the energy efficiency with as little effort as possible. This is why we developed a strategy with focus on rating energy efficiency during the development process on an on-going basis. In this article, we introduce a strategy, based on well-known software testing approaches and CI, to take energy efficiency into account during the daily work of a software developer.

The energy consumption of a computer system is linked to execution times and system utilization, which is caused by the software executed on it [Bircher and John 2012; Schubert et al. 2012]. Shorter execution times and less system load often result in less power consumption. The execution times and the system load caused by software could potentially be optimized through better, i.e. faster, algorithms, the use of more suitable data structures, better software design, source code optimizations, etc. All these optimizations should be made during the development process.

The strategy describes how to integrate the monitoring of energy consumption and how to handle the measurements to get an automatically generated energy efficiency report, subsequent to any integration as well as the useful work done during test execution. We analyze and define the requirements to software tests used for rating software energy efficiency and check if well-known test terms (e.g. unit integration, system, and performance tests) fit to these requirements. By defining individual metrics at test level, we are able to rate the energy efficiency on single methods, just as different modules and whole software systems.

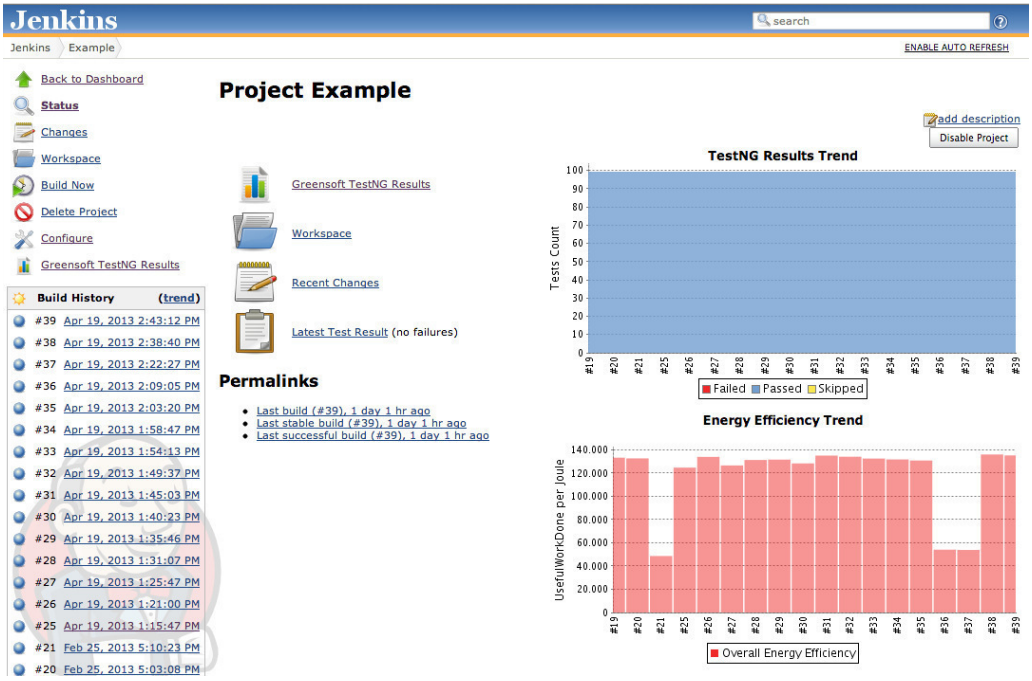

Figure 3 *Screenshot of the Energy Efficiency Trend in the CI Environment*

A deeper look into the build and test execution workflow of the unit testing-framework and the CI environment enables us to extend the workflows and instrument the test execution in order to record system load and energy consumption. We extend the test result report offered by the CI environment, showing the energy consumption of builds, test classes and test methods, and the rating of its energy efficiency caused by test execution and measurement results (Figure 3). According to the goals of CI, this is how we allow maximized feedback on energy efficiency of their software to the developers: by rating the test execution of any build performed by the CI environment.

We evaluated the strategy by applying the exemplary setup on an example project where we implemented two sorting algorithms, heap sort and quick sort, and wrote several performance tests suitable for the measurement of the energy consumption. In this way we were able to show that it is possible to apply the strategy in the field, means how to measure energy consumption with software tests and how daily or per integration reports inside the CI environment can look like. Hence, energy optimization during development process is possible and the effort to do so can be minimized by using established techniques like software testing and CI. The tool is just a prototype and not available so far.

Commonly, the test builds during a software development process are done after every check in into the central repository instead of e.g. one build every night. This includes all of the tests: unit tests, integration tests and automated installations. All versions of which the source code has been changed will be build and tested directly after check in.

Regarding renewable energies, it comes to mind if it might be sensible to shift the builds and tests to a time where the power overflow is high. In general, this is the case during the night. Indeed this cuts across the idea of CI. In this context, the big advantage of CI is that the software developer gets feedback directly so that he gets a feel for interaction of changing code and the resulting energy measurement. We advocate the early feedback about energy consumptions to the developer since he needs to be sensitized for this issue. In a further step, when the developers are aware of the energy consumption of their code, one can think about less builds and therefore less energy consumption during the development process. If the developers see the importance of energy measurements during the development process, one can start to optimize the point in time by taking account of the natural characteristics of renewable energies.

6.2 Acquisition of Data Relevant for Calculation

In order to calculate the CF of a software product during its whole life cycle, a lot of data needs to be collected. Even if one concentrates on the development phase, like we did, one has to collect many data, what takes a lot of time. Since enterprises have not attributed high importance to carbon footprinting so far and the terms as well as the methods are not well known, there need to be tools to support the data collecting in an easy way. It might increase the willingness to use CF calculations, if tools to collect the data are made available, which ideally calculate the CO_2 emissions and which are also easy to use.

Such supporting checklists and standard forms to gather the following data may be:

- Characteristics of the enterprise (e.g. number of rooms, electrical power consumption, details of the infrastructure)
- Project relevant data (hours of work, meetings, travelling, office supplies)

Additionally, data like working hours can be recorded directly in a project management or issue tracking tool. In that way, they can be directly assigned to the different phases of a software life cycle. Hence, we are convinced that it makes sense to extend such tools by adding aspects like energy consumption.

Another aspect to support the attention for the CF is mobile computing: mobile apps can be used to display the CF of e.g. products [Dada et al. 2008] or to help to collect the data. In that way, the user has the possibility to always be informed about the CF. The availability of more information makes it even more transparent to the user.

In general, the first step is to check if it makes sense to calculate a CF. It is not sensible to calculate a CF if the acquisition of the relevant data means a lot of additional effort and raises the CF as a result. One has to balance the costs and benefits first.

7 Conclusion

The climate change is one of the biggest challenges nowadays. One causer of the climate change are the growing CO_2 emissions. Here, different approaches how to reduce these emissions exist. One possibility to get an impression of the emissions and to create more transparency and awareness in this context is to calculate the carbon footprint of goods and services. In that way, it is possible to identify optimization potentials. Regarding the overall ICT, there are different studies and approaches how to calculate the CF of servers etc. While the carbon footprint calculation of products over their whole life cycle is described in ISO 14067, methods are missing in the context of software.

Hence, we presented an approach how to calculate the carbon footprint of software products by defining an exemplary enterprise as a first step. Based on this definition, we calculate the carbon footprint by focusing on the development phase. The resulting CF is converted into the functional unit kg CO_2e per person month.

Since there are many impacts to be considered in the carbon footprint calculation, we discuss these and point out how to address them during the calculation process. Especially commuting is an important aspect. However, the more a software is installed after development, the less the development phase is taken into account.

Summarizing, from our point of view, two aspects are important to make software greener: on the one hand, activities to make the production process greener and on the other hand, actions to make the software product itself greener. In order to achieve the first goal, it is useful to calculate the CF of the software production process as described in our article. Hence, in order to improve the energy efficiency of software itself, we suggest adding this quality feature into CI during the development process. Thereby, it is possible to measure the energy consumption of a new software build continuously. The software developer gets feedback and can immediately see, whether or not a new build might be more energy inefficient. Since one main goal of Green IT is to produce environmentally friendly software, the presented methods help developers and designers without changing their software engineering methods in general.

References

ALBERTAO, F. 2004. *Sustainable Software Engineering*. http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering#about. Accessed 14 April 2017.

ALFIERI, L., SALAMON, P., PAPPENBERGER, F., WETTERHALL, F., AND THIELEN, J. 2012. Operational early warning systems for water-related hazards in Europe. *Environmental Science & Policy 21*, 35–49.

BELADY, C., AZEVEDO, D., PATTERSON, M., POUCHET, J., AND TIPLEY, R. 2010. Carbon Usage Effectiveness (CUE): A Green Grid Data Center Sustainability Metric. *The Green Grid White Paper*.

BHATIA, P., CUMMIS, C., BROWN, A., RICH, D., DRAUCKER, L., AND LAHD, H. 2011. *Corporate Value Chain (Scope 3) Accounting and Reporting Standard. Supplement to the GHG Protocol Corporate Accounting and Reporting Standard*.

BIRCHER, W.L., AND JOHN, L.K. 2012. Complete System Power Estimation Using Processor Performance Events. *IEEE Transactions on Computers 61*, 4, 563–577. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5714687.

BRANDTZÆG, P.B., HEIM, J., AND KARAHASANOVIĆ, A. 2011. Understanding the new digital divide—A typology of Internet users in Europe. *International journal of human-computer studies 69*, 3, 123–138.

BRAUCKMÜLLER, T. 2011. *Digital Society – Six user types in comparison*.

BRAUNGART, M., AND MCDONOUGH, W. 2009. *Cradle to cradle. Remaking the way we make things*. Vintage, London.

BRITISH STANDARDS INSTITUTION. 2011. *PAS 2050:2011. Specification for the assessment of the life cycle greenhouse gas emissions of goods and services*. http://www.bsigroup.com/Standards-and-Publications/How-we-can-help-you/Professional-Standards-Service/PAS-2050.

CAPRA, E., FRANCALANCI, C., MERLO, F., AND ROSSI LAMASTRA, C. 2009. A survey on firms participation in open source community projects. *Open Source Ecosystems: Diverse Communities Interacting*, 225–236. http://flosshub.org/system/files/Survey%20on%20Firns%27%20Participation.pdf.

CAPRA, E., FRANCALANCI, C., AND SLAUGHTER, S.A. 2011. Is software green? Application development environments and energy efficiency in open source applications. *Information and Software Technology 54*, 60–71.

COOK, G., AND VAN HORN, J. 2011. *How dirty is your data? A Look at the Energy Choices That Power Cloud Computing*, Amsterdam.

DADA, A., REISCHACH, F. von, and STAAKE, T. 2008. Displaying dynamic carbon footprints of products on mobile phones. In *Advances in Pervasive Computing, Adjunct Proceedings of the 6th International Conference on Pervasive Computing*, 119–121.

DENG, W., LIU, F., JIN, H., LI, B., AND LI, D. 2013. Harnessing Renewable Energy in Cloud Datacenters: Opportunities and Challenges. *IEEE Network Magazine 28*, 1, 48–55.

DICK, M., KERN, E., DRANGMEISTER, J., NAUMANN, S., AND JOHANN, T. 2011. Measurement and Rating of Software-induced Energy Consumption of Desktop PCs and Servers. In *Innovations in sharing environmental observations and information. Proceedings of the 25th International Conference EnviroInfo October 5 - 7, 2011, Ispra, Italy*, W. PILLMANN, S. SCHADE AND P. SMITS, Eds. Shaker, Aachen, 290–299.

DICK, M., AND NAUMANN, S. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany*, K. GREVE AND A. B. CREMERS, Eds. Shaker, Aachen, 706–715.

ENERGIEINSTITUT DER WIRTSCHAFT 2010. *KMU-Initiative zur Energieeffizienzsteigerung. Begleitstudie: Kennwerte zur Energieeffizienz in KMU*. Endbericht, Wien.

EUROPEAN COMMISSION (Ed.) 2005. *The new SME definition. User guide and model declaration*. Enterprise and Industry Publications.

FEDERAL ENVIRONMENT AGENCY (Umweltbundesamt). 2011. *Prozessorientierte Basisdaten für Umweltmanagement-Instrumente (ProBas). Kraftwerksmix zur Stromerzeugung in Deutschland, Record: El-KW-Park-DE-2010*.

FUJITSU TECHNOLOGY SOLUTIONS 2011. *White Paper: Energy Consumption ESPRIMO E9900 E-Star5*.

FUJITSU TECHNOLOGY SOLUTIONS 2010. *White Paper: Life Cycle Assessment and Product Carbon Footprint. PRIMERGY TX 300 S5 and PRIMERGY RX 300 S5 Server*.

GALLI, A., WIEDMANN, T., ERCIN, E., KNOBLAUCH, D., EWING, B., AND GILJUM, S. 2012. Integrating Ecological, Carbon and Water footprint into a Footprint Family of indicators: Definition and role in tracking human pressure on the planet. In *Ecological Indicators*, 100–112.

GARTNER, INC. 2007. *Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO_2 Emissions*, Stamford.

GOMBINER, J. 2011. Carbon Footprinting the Internet. *Consilience-The Journal of Sustainable Development 5*, 1.

GONG, L., XIE, J., LI, X., AND DENG, B. 2013. Study on energy saving strategy and evaluation method of green cloud computing system. In *Industrial Electronics and Applications (ICIEA), 2013*. 483–488.

GRABOLLE, A., AND LOITZ, T. 2007. *Pendos CO2-Zähler. Die CO2-Tabelle für ein klimafreundliches Leben : die wichtigsten Zahlen, Fakten und Vergleiche zu Konsum, Strom, Heizen und Mobilität*. Pendo-Verlag, München, Zürich.

GRAU, A. 2009. *Commuters: Majority still going by car*. STATmagazin. Destatis - Statistisches Bundesamt Deutschland, Wiesbaden.

HINDLE, A. 2012. Green mining: A methodology of relating software change to power consumption. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, 78–87.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *Carbon footprint of products - Requirements and guidelines for quantification and communication*, 14067.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. 2014. *Carbon footprint of products—Requirements and guidelines for quantification and communication 13.020.40*, ISO/DIS 14067.2.

JING, S.-Y., ALI, S., SHE, K., AND ZHONG, Y. 2013. State-of-the-art research study for green cloud computing. *The Journal of Supercomputing 65*, 1, 445–468.

JOHANN, T., DICK, M., KERN, E., AND NAUMANN, S. 2012. How to Measure Energy-Efficiency of Software: Metrics and Measurement Results. In *Proceedings of the First International Workshop on Green and Sustainable Software (GREENS) 2012. held in conjunction with ICSE 2012, The International Conference on Software Engineering, June 2-9, Zurich, Switzerland*, IEEE, Ed. IEEE Computer Society, 51–54.

JONES LANG LASALLE 2009. *Büroflächenkennziffern – 2009. Studie zur Belegung von Büroflächen*. on.point, Frankfurt am Main.

JOSCHKO, P., WIDOK, A., AND PAGE, B. 2013. A Simulation Tool for Maintenance Processes of Offshore Wind Farms. In *Proceedings of the International Conference on Harbor, Maritime & Multimodal Logistic Modelling and Simulation 2013. organized within the 10th International Mediterranean and Latin American Modeling Multiconference, 25th-27th Septembre*, Athens (Greece).

KÄFER, G. 2009. *Green SE: Ideas for Including Energy Efficiency into your Software Projects. Technical Briefing (TB2)*. 31st International Conference on Software Engineering, Vancouver.

KAUR, M., KAUR, G., AND SINGH, P. 2013. A RADICAL ENERGY EFFICIENT FRAMEWORK FOR GREEN CLOUD. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*.

KFW. *KfW-CO2-Gebäudesanierungsprogramm. Technisches Merkblatt für das Maßnahmenpaket 4*. Programm-Nr. 130/132 Anlage B, Frankfurt am Main.

KRAMER, K.-L. 2012. *User experience in the age of sustainability. A practitioner's blueprint*. Morgan Kaufmann, Waltham, MA.

LAMI, G., FABBRINI, F., AND FUSANI MARIO. 2012. Software Sustainability from a Process-Centric Perspective. In *EuroSPI 2012, CCIS 301*, D. Winkler, R.V O'Connor and R. Messnarz, Eds. Springer, 97–108.

LAURENT, A., OLSEN, S.I., AND HAUSCHILD, M.Z. 2012. Limitations of Carbon Footprint as Indicator of Environmental Sustainability. *Environmental Science & Technology 46*, 7, 4100–4108.

LI, Z., HAN, T., LI, Y., AND LIU, Z. 2013. Risk Early Warning System Preventing Natural Disaster in Yunnan Power Grid. In *Informatics and Management Science V*. Springer, 69–76.

LIEBERT (Ed.) 2000. *High-Availability Power Systems. Part I: UPS Internal Topology*.

MAHAUX, M., HEYMANS, P., AND SAVAL, G. 2011. Discovering Sustainability Requirements: An Experience Report. In *Requirements Engineering: Foundation for Software Quality. 17th International Working*

Conference, REFSQ 2011, Essen, Germany, March 28-30, 2011, Proceedings, D. M. BERRY AND X. FRANCH, Eds. Springer, Berlin, Heidelberg, 19–33.

MAHMOUD, S.S., AND AHMAD, I. 2013. A Green Model for Sustainable Software Engineering 2013. *International Journal of Software Engineering and Its Applications 7*, 4, 55–74. http://www.sersc.org/journals/IJSEIA/vol7_no4_2013/5.pdf.

NAUMANN, S., DICK, M., KERN, E., AND JOHANN, T. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *SUSCOM 1*, 4, 294–304. doi:10.1016/j.suscom.2011.06.004.

OPITZ, N., KRUP, H., AND KOLBE, L.M. 2014. Green Business Process Management-A Definition and Research Framework. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, 3808–3817.

PCF PILOT PROJECT GERMANY. 2009. *Product Carbon Footprinting - The Right Way to Promote Low Carbon Products and Consumption Habits? Experiences, findings and recommendations from the Product Carbon Footprint Pilot Project Germany*. http://www.pcf-projekt.de/files/1241103260/lessons-learned_2009.pdf.

PRUULMANN-VENGERFELDT, P. 2006. *Information technology users and uses within the different layers of the information environment in Estonia*. Dissertationes de mediis et communicationibus, Universitatis Tartuensis 4. Tartu University Press, Tartu.

RAL GGMBH. 2012. *Basic Criteria for Award of the Environmental Label Energy-Conscious Data Centers*, RAL-UZ 161. Accessed 28 March 2014.

RAPP, B., AND BREMER, J. 2013. IT Solutions for EPI Management. In *Organizations' Environmental Performance Indicators. Measuring, Monitoring, and Management*, A. DADA, K. STANOEVSKA AND J. M. GÓMEZ, Eds. Springer, 19–31.

ROHJANS, S., DANEKAS, C., AND USLAR, M. 2012. Requirements for Smart Grid ICT-architectures. In *3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe). Berlin, Germany : October 14-17, 2012*. IEEE, Piscataway, N.J, 1–8.

SCHUBERT, S., KOSTIC, D., ZWAENEPOEL, W., AND SHIN, K.G. 2012. Profiling Software for Energy Consumption. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, 515–522.

SHENOY, S.S., AND EERATTA, R. 2011. Green software development model: An approach towards sustainable software development. In *India Conference (INDICON), 2011 Annual IEEE*, 1–6.

SPADARO, J.V., LANGLOIS, L., AND HAMILTON, B. 2000. Greenhouse gas emissions of electricity generation chains: Assessing the difference. *IAEA bulletin 42*, 2, 19–28.

STATISTIK DER BUNDESAGENTUR FÜR ARBEIT 2013. *Arbeitsmarkt in Zahlen. Betriebe und sozialversicherungspflichtige Beschäftigung, Deutschland, Stichtag 30. Juni 2012*. Arbeitsmarkt in Zahlen - Beschäftigungsstatistik, Nürnberg.

TAINA, J. 2010. How Green Is Your Software? In *Software Business. First International Conference, ICSOB 2010, Jyväskylä, Finland, June 21-23, 2010. Proceedings*, P. TYRVÄINEN, M. A. CUSUMANO AND S. JANSEN, Eds. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 151–162.

VICKERY, G., AND MICKOLEIT, A. 2013. Greener and Smarter: Information Technology can Improve the Environment in Many Ways. In *Broadband Networks, Smart Grids and Climate Change*, E. M. NOAM, L. M. PUPILLO AND J. J. KRANZ, Eds. Springer, 33–37.

WACKERNAGEL, M., AND YOUNT, J.D. 1998. The ecological footprint: an indicator of progress toward regional sustainability. *Environmental monitoring and assessment 51*, 1-2, 511–529.

WÄFLER, J., AND HEEGAARD, P.E. 2013. Interdependency Modeling in Smart Grid and the Influence of ICT on Dependability. In *Advances in Communication Networking*. Springer, 185–196.

WILLIAMS, D. 2013. *GHG Protocol Product Life Cycle Accounting and Reporting Standard ICT Sector Guidance. Chapter 7: Guide for assessing GHG emissions related to software*. http://www.ghgprotocol.org/files/ghgp/GHGP-ICT-Software-v2-9-26JAN2013.pdf. Accessed 6 February 2013.

Sustainable software products – towards assessment criteria for resource and energy efficiency

Eva Kern[a,b*], Lorenz M. Hilty[c,d,e], Achim Guldner[b], Yuliyan V. Maksimov[c,f], Andreas Filler[b,g], Jens Gröger[h], Stefan Naumann[b]

[a] Leuphana University Lueneburg, Universitätsallee 1, 21335 Lueneburg, Germany

[b] Institute for Software Systems in Business, Environment, and Administration, Trier University of Applied Sciences, Environmental Campus Birkenfeld, P.O. Box 1380, 55761 Birkenfeld, Germany

[c] Department of Informatics, University of Zurich, Zurich, Switzerland

[d] Technology and Society Lab, Empa Swiss Federal Laboratories for Materials Science and Technology, St. Gallen, Switzerland

[e] KTH Royal Institute of Technology, Stockholm, Sweden

[f] i4Ds Centre for Requirements Engineering, University of Applied Sciences Northwestern Switzerland (FHNW), Windisch, Switzerland

[g] Institute of Technology Management, University of St. Gallen, Dufourstrasse 40a, 9000 St. Gallen, Switzerland

[h] Sustainable Products & Material Flows Division, Oeko-Institut, Schicklerstraße 5-7, 10179 Berlin, Germany

Abstract. Many authors have proposed criteria to assess the "environmental friendliness" or "sustainability" of software products. However, a causal model that links observable properties of a software product to conditions of it being green or (more general) sustainable is still missing. Such a causal model is necessary because software products are intangible goods and, as such, only have indirect effects on the physical world. In particular, software products are not subject to any wear and tear, they can be copied without great effort, and generate no waste or emissions when being disposed of. Viewed in isolation, software seems to be a perfectly sustainable type of product. In real life, however, software products with the same or similar functionality can differ substantially in the burden they place on natural resources, especially if the sequence of released versions and resulting hardware obsolescence is taken into account. In this article, we present a model describing the causal chains from software products to their impacts on natural resources, including energy sources, from a life-cycle perspective. We focus on (i) the demands of software for hardware capacities (local, remote, and in the connecting network) and the resulting hardware energy demand, (ii) the expectations of users regarding such demands and how these affect hardware operating life, and (iii) the autonomy of users in managing

their software use with regard to resource efficiency. We propose a hierarchical set of criteria and indicators to assess these impacts. We demonstrate the application of this criteria set, including the definition of standard usage scenarios for chosen categories of software products. We further discuss the practicability of this type of assessment, its acceptability for several stakeholders and potential consequences for the eco-labelling of software products and sustainable software design.

Keywords. environmental criteria for software, green software, resource efficiency, sustainability indicators, model of software impacts, energy-aware software

1 Introduction

This article presents the results of a project on sustainable software design commissioned by the German Federal Environment Agency.[1] The project builds on the results of earlier projects [Naumann et al. 2011; Hilty et al. 2015].

The goal of the present project is to develop a method for evaluating the environmental impacts of software products and to provide recommendations to software engineers for developing software with low environmental impact. The evaluation method is intended to support both the procurement of software products with the consideration of environmental criteria and the development of resource-efficient software. In particular, the method is supposed to enable a comparison of two given software products with similar functionality, where the comparison will focus on the impacts of their use on natural resources. Based on the formulation of ambitious minimum standards, the method will help to define criteria for the awarding of an environmental or quality label to sustainable software products. The potential effect of exploiting the software functionality, such as the carbon emissions saved by using videoconferencing software to avoid flights (as demonstrated by Coroama et al. [Coroama et al. 2012]), which may be much larger, are not in the focus of this research. In the given example, we would distinguish among a set of software products instead, all providing videoconferencing functionality, by the amount of natural resources consumed per hour of using them.

Thus, the project makes a contribution to expanding the focus of "Green IT" beyond the hardware level to include the software level as the place where hardware requirements emerge and expand. Since software products are immaterial goods, it is a challenge to capture the indirect material and thus environmental impacts of these products in conceptual and methodological terms.

[1] "Sustainable software design – Development and application of criteria for resource-efficient software products with consideration of existing methods." UFOPLAN project no. 3715 37 601 0

A product's environmental impacts generally occur through the use of natural resources[2] during the life cycle of the product. We take this life-cycle perspective into account in relation to software products, as well (see Figure 1, upper part). We further consider that the hardware needed to operate a software product must be produced, supplied with electricity, and disposed of at the end of its useful life (Figure 1, middle part). Thus, every software product is responsible for a quantifiable fraction of the life cycle of all the hardware products required for its operation (programmable devices of any kind, peripheral devices, and storage media). During production, use and disposal, these hardware products demand a quantifiable part of natural resources.

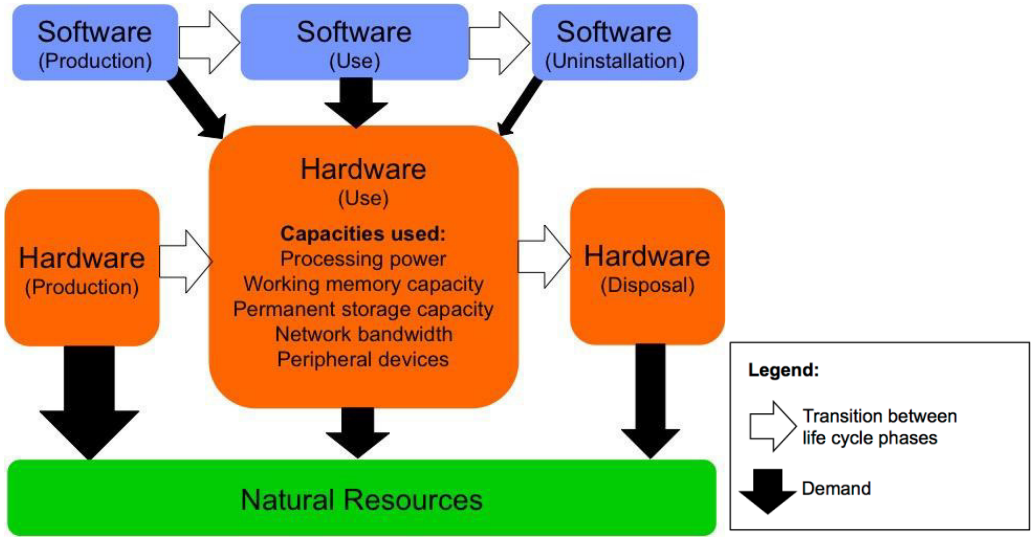

Figure 1 *Life cycles of hardware and software (horizontal dimension) and the resource demand induced by the life cycles (vertical dimension)*

Because it takes a life-cycle perspective, this approach can be expanded to include social aspects of sustainability, e.g. when producing the raw materials for the hardware (or producing the software) as well as the working conditions in hardware production and disposal; however, our focus is on the environmental aspects.

At the software level, we intentionally limit our perspective to the use phase when developing the criteria. Although the production (development) and disposal (uninstallation) of software also has its indirect environmental impacts, they are not considered in this project. One exception is that we take into account the resources that can be wasted due to a software product's incomplete uninstallability.

[2] In this article, we reserve the term "resource" for natural resources and mostly avoid the technical term "hardware resource" by describing hardware resources in terms of their capacities, i.e., quantifiable aspects of their performance such as computing power, storage capacity, or transmission bandwidth.

Focusing on the use phase of software products is justified for standard software that is installed and run in millions or billions of cases. Minor changes of software properties decided by the developers can have a huge impact in terms of resource demand when the software product is being used, just because of the high multiplication factor that has to be applied due to the large number of installations and executions.

The purpose of the set of criteria is to evaluate a software product on the basis of characteristics that are observable in its use phase, be it by the users themselves or by persons conducting professional tests. We excluded software production because assessing the process of software development seems less important to us than influencing it, in particular by making recommendations addressed to those responsible for software development.

The evaluation of widespread software products requires more than a snapshot, ideally an observation of the software product over longer periods of time which cover several versions. From this long-term perspective, questions concerning software-induced purchasing of new hardware become more relevant, for example.

Expressed in abstract terms, our analysis focuses on two essential flows caused by the use of a software product (see Figure 2):

- the flow of energy through the hardware running the software (electricity to waste heat),
- the flow of hardware through the organization using it (new hardware to electronic waste).

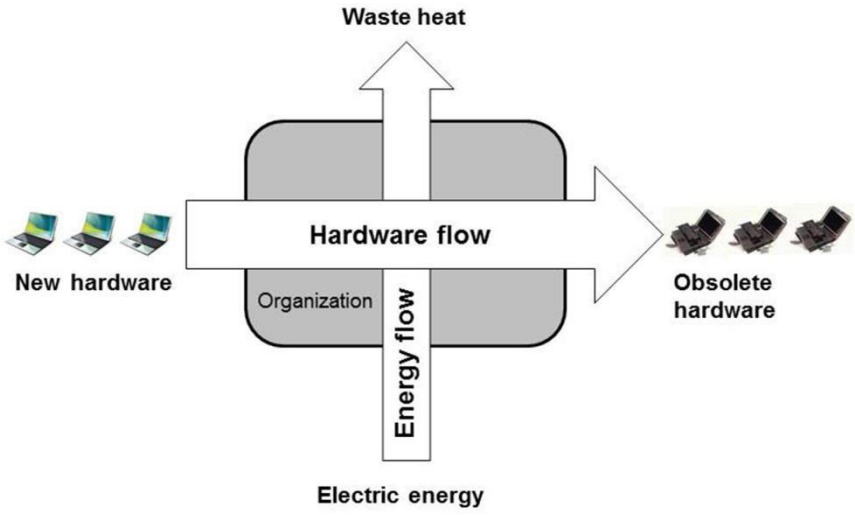

Figure 2 *The two main physical flows to be reduced by sustainable software*

The impact of both flows on natural resources can be determined by using standard life-cycle assessment (LCA) methods. Life cycle inventories for production and disposal of the most important hardware components exist for this purpose, and we take them as given without entering a detailed

discussion [Hischier et al. 2015; Wäger et al. 2015]. Energy flow can also be evaluated with LCA methods; the various methods for generating electricity have been examined sufficiently; that's why we take these results for granted as well [Turconi et al. 2013; Hertwich et al. 2015].

If a software product causes significantly lower hardware and energy flows than competing products with similar functionality, it can be considered "relatively sustainable."[3]

That is why it is sufficient to address the impact of software on the required hardware capacities. If one imagines a chain of impacts from software characteristics to natural resource use, we exclusively analyze the section of this causal chain from the software characteristics to the hardware products and their electricity consumption, because it is the only part of the causal chain that can vary among software products. In other words, a software product has usually no impact on the way hardware is manufactured or electricity is generated, but only on the amount of hardware and electricity consumed for its running. Exceptions may occur in cases of co-design of hardware and software or when an operating system and the hardware are optimized together. However, our focus is on standard application software. This does not exclude that the criteria or subsets of them can in principle be applied to a broader class of software products.

Additional hardware that is not directly involved in executing the programs of a software product, but indirectly used, such as the Internet nodes routing the traffic generated by the software, may not even be known; in such cases, using average impact factors like "Internet energy intensity" in kWh/GB [Coroama and Hilty 2014] is better than implicitly assuming a zero impact.

Thus, operational criteria are necessary to be able to assess the sustainability of software with reference to the hardware and energy flows it induces. Then these criteria can be applied, e.g., to inform people responsible for software development or software procurement – or to award an eco-label for software products [Kern et al. 2015].

The set of criteria proposed here focuses on environmental impacts resulting from the operation of a software product. As already mentioned, this does not rule out that the awarding of eco-labels also includes social criteria regarding the process of software development (e.g., compliance with ILO[4] standards when outsourcing programming work), the functionality of the software (e.g., accessibility, or exclusion of particular categories such as violent games), or other aspects. It seems important to us, however, to treat the impacts of software characteristics on natural resource consumption as a clearly

[3] The functionality of a software product, and thus its utility, will not be evaluated here. Our goal is restricted to estimating and evaluating the amount of resource use it induces. A given amount of useful work can be related to the amount of resource use induced to determine efficiency.

[4] International Labour Organization

defined object of research from the outset and not to confound it with other issues. Studies and criteria are available for many of these neighboring issues and can be used to complement our set of criteria [Johann and Maalej 2013; Al Hinai and Chitchyan 2015].

A tree of criteria was developed during the first phase of the project. The leaves of this tree are indicators serving to operationalize the parent criterion. An overview of the set of criteria is provided in Section 3. The full set of criteria is available in the supporting information to this article, which will be also published online. We will provide updates at: http://green-software-engineering.de/criteria-catalog, add a revision control on this website and invite for comments and feedback. In addition, we will keep the version to which this article directly refers as supplementary information constant and accessible. The set of criteria is intended to be used as a catalogue from which a selection can be made depending on the goal and scope of the assessment task. Direct comparisons of software products are of course only possible if the same selection is used in all cases investigated.

2 Criteria for software sustainability and related approaches in literature

Many fields of research are addressing interactions between ICT and the goal of sustainable development. Hilty and Aebischer provide a general framework for this type of research [Hilty and Aebischer 2015]. As a special case, research on software sustainability is focusing on the software part of ICT systems, looking for interdependencies between software engineering and sustainability issues. A first literature review in this context was done by Penzenstadler et al. in 2012 [Penzenstadler et al. 2012]. Based on their data, at that time "little research coverage on the different aspects of sustainability in software engineering" was found. This seemed to have changed two years later, when a similar study by the almost same authors came to the conclusion: "The topic of SE4S [Software Engineering for Sustainability] has received wide-spread attention in the software engineering community over the past few years." [Penzenstadler et al. 2014] Thus, the interest for the issue is widening.

Addressing a more specific topic, Calero et al. [Calero et al. 2013a] analyze publications dealing with software sustainability measurements. In addition to measurements, metrics are an issue in evaluating software products. Here, Bozzelli et al. [Bozzelli et al. 2013] describe and classify metrics regarding the so-called "greenness" of software while reviewing existing literature in this context. However, they do not define what is meant by "greenness of software". According to their results, the "research community is focusing on metrics strictly related to energy consumption and saving dimensions".

Nevertheless, many authors working on criteria and metrics for "green" or "sustainable" software have a broader understanding of the issue, e.g. addressing the "impacts on economy, society, human beings, and [the] environment that result from development, deployment, and usage of the software."

[Dick et al. 2010; Naumann et al. 2011] Several other definitions of "sustainable software" [Mahaux et al. 2011; Calero et al. 2013b; Penzenstadler 2013] discuss the issue from different perspectives. Summarizing, all of them address the protection of resources, among other issues. However, a standardized understanding of "green" or rather "sustainable" software is still missing. As set out in the introduction, we will focus on environmental sustainability.

2.1 Strategies on finding criteria for sustainable software

The following strategies on how to find and summarize criteria for sustainability or "greenness" of software products can be identified in literature:

i. Taking existing software quality criteria (such as maintainability) or quality models (such as ISO 25010) and interpreting them in the context of environmental sustainability,
ii. taking existing sustainability criteria (such as energy efficiency) and clustering them to categories (bottom-up approach), and
iii. taking an LCA approach and defining software sustainability criteria for software life cycle phases (top-down approach).

Table 1 summarizes literature that can be categorized into methods (i) to (iii). The list is not intended to be exhaustive.

Table 1 *Comparison of approaches on criteria for sustainable software*

Approach by	**Objectives**	**Outcomes**	**Criteria (Examples)**
Method (i)			
Albertao [2004] Albertao et al. [2010]	Assessing properties of software for environmental, economic and social aspects; Introducing a set of metrics to assess the sustainability of software products, demonstration how to use the metrics	Sustainability performance metrics and strategy how to improve follow-up releases by using the metrics	Modifiability, Reusability, Dependability, Usability, Efficiency, and Predictability
Calero et al. [2013b] Calero et al. [2015]	Extending the ISO 25010 quality model by including sustainability aspects; definition of "greenability"	Model for software sustainability that can be added to the ISO software product quality model	Energy efficiency, Resource Optimization, Capacity Optimization, Perdurability
Method (ii)			
Taina [2011]	Developing the criteria set "green software factors"	Framework for green quality factors: related to software development and execution	Feasibility (Carbon Footprint, Energy, Travel, …), Efficiency (CPU-intensity, Idleness, …), Sustainability (Reduction, Beauty, …)

Kern et al. [2013]	Summarizing existing approaches in a "quality model" for green and sustainable software	Quality model to classify green software and its engineering and exemplary corresponding metrics	Feasibility, Social Aspects, Portability, Efficiency, Reflectivity, Product Sustainability
Method (iii)			
Abenius [2009]	Evaluation of "Green IT", especially "Green Software", and pointing out possibilities to use software in a more energy-saving way	Examples of actions towards Green IT, mapped to software life cycle phases	Choice of Material, Reuse Refurbish Recycle, Production Logistics
Naumann et al. [2011]	Mapping potential effects of software to sustainable development	Life cycle model for software products including effects relevant to sustainability	Working Conditions, Manuals, Data Medium, Download Size, Accessibility, Hardware Requirements, Backup Size

2.2 Research Design

In order to create the set of criteria presented in this article, we used a procedure combining methods (i) to (iii): We first collected available extensions of quality models, findings of literature reviews (including scientific and practical publications), and additional ideas in the context of software sustainability (expert discussions). We then related the collected elements to each other and clustered them (bottom-up approach). This resulted in a structured collection of criteria which was reassessed for consolidating overlapping ideas. Then we mapped the criteria to software life cycle phases and to the sustainability aspects resulting from the causal model of the impact of software on natural resource use that was developed in parallel (Figure 3). Consequently, only a subset of the consolidated set of criteria has been followed up.

The resulting set of criteria will be presented in Section 3. The practical application of example criteria is demonstrated with existing software products in Section 4.

3 A set of criteria to assess the resource efficiency of software products

As a guideline to structure the criteria collected from the literature, we developed a causal model describing the principal mechanisms by which a software product can influence the demand for natural resources (inspired by [Som et al. 2009]). Each of these mechanisms is a direct path (representing a causal chain) from observable properties of the software product to the use of hardware capacities and,

finally, to the demand for natural resources that is induced by producing and running the hardware. This causal model is shown in Figure 3. We will first explain the model and then provide an overview of the set of criteria we formulated on its basis.

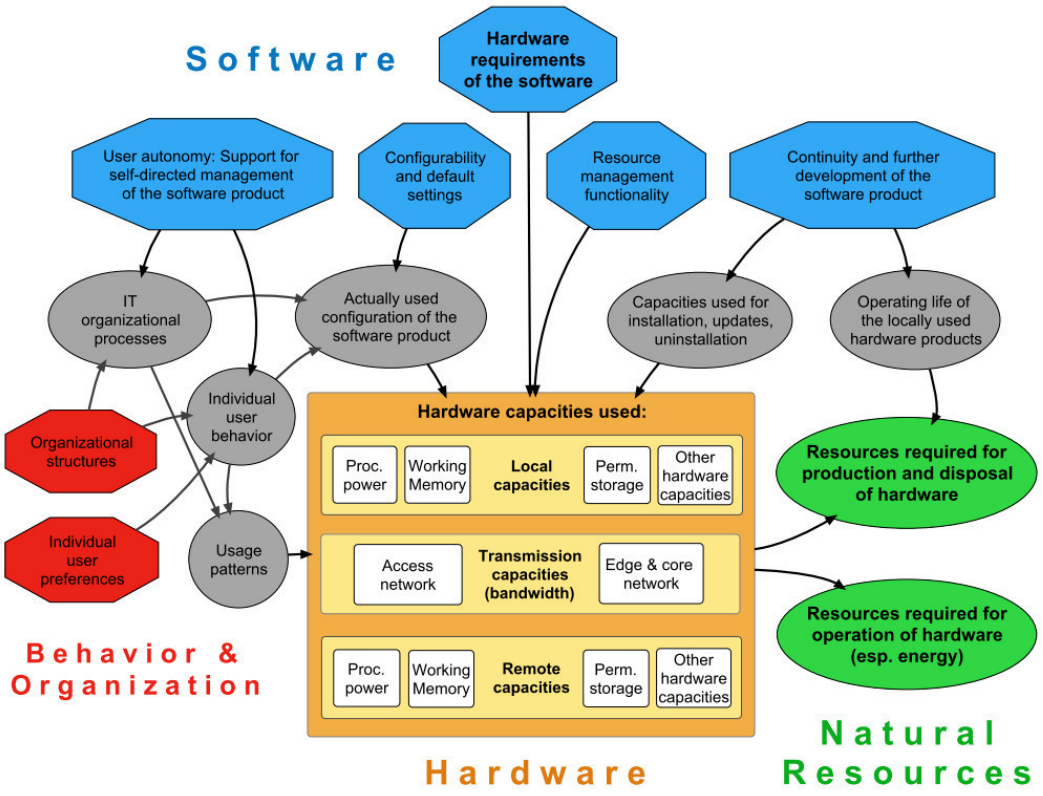

Figure 3 *The model describing the causal chains leading from software properties (blue) to the natural resources required for using the software (green). A link means "has impact on". The central node "Hardware capacities used" is structured into several types of local, transmission and remote hardware capacities procured and operated to run the software product.*

The observable properties of the software product are represented by the five (blue) nodes in the upper part of the graph. The most important property is the hardware requirements, i.e., the amount of hardware capacities the software product requires (both declared by the producer and actually measured, as partly shown in Section 4).

The amount of natural resources (including energy resources) demanded for running the software is represented by the two (green) oval nodes labelled "Resources required for production and disposal of hardware" and "Resources required for operation of hardware" (lower right part of the figure).

Each path leading from a software product property to the natural resource demand represents a possible way for software developers to influence the natural resource consumption that will be caused by their products. All paths happen to cross the node "Hardware capacities used" – except one.

This exceptional path starts at "Continuity and further development of the software product", which has a direct impact on the "Operating life of the locally used hardware products", which then affects the "Resources required for production and disposal of hardware". The idea behind this causal chain is that

the evolution of a software product has an influence on hardware obsolescence. If, for example, a new version is more demanding with regard to hardware capacities, it can shorten the useful life of the hardware equipment in use even when this is still fully functional, which then increases the amount of natural resources required to produce the hardware per unit of used hardware capacity*time.

All the other paths from software properties to natural resources cross the (orange) node "Hardware capacities used". Note that we assume that there are – in the general case – three spheres of hardware capacities involved in running application software:

- local capacities provided by end consumer devices,
- transmission capacities provided by network infrastructure, which includes hardware such as routers, switches and links (e.g., optical fibres), and
- remote capacities provided by servers.

Even if the software product under test runs only locally, it may require and access remote and transmission capacities which have to be assessed as well.

Another important feature of the model is that some of the causal chains reach the "Hardware capacities used" via (red) behavioral and organizational aspects of using the software product. This includes everything that may influence the "actually used configuration of the software product" or the "Usage patterns". This reflects that the actual configuration and the way of using the software product both influence the hardware capacity it requires. This is where users and the organizational structures in which they act come into play as influencers of the software-induced resource demand. Software developers cannot (and probably should not want to) determine user behavior and organizational structures, but they do define the decision space for organizations and individual users to optimize resource consumption when using the software. If this decision space is too constrained to allow users who intend to save resources to do so, this is less sustainable than providing such options. The grey nodes are intermediate nodes in the causal network.

In Table 2, we briefly explain the five software properties used in our model and provide examples for the criteria addressing them. Please note that the criteria numbers shown in brackets refer to Table 3 and to the full set of criteria that is provided as supporting information to this article.

Table 2 *Software properties (corresponding to the five top nodes of Figure 3) and example criteria. The numbers in brackets refer to the set of criteria provided as supporting information and described in Table 3.*

Software property	**Rationale**	**Example criterion**
Support for self-directed management of the software product (user autonomy)	If the organization or individual using the software product wants to save resources, the software should support this intention.	Uninstallability [3.2]
Configurability and default settings	The default settings for configuring the software product may have a	This aspect can be viewed as overarching, it affects

	substantial influence on the resources used.	several criteria by creating awareness for default settings, e.g., Resource Management [1.3].
Hardware requirements	The hardware required to run the software product have a crucial influence on the hardware capacities the user will procure and on the load and energy consumption when using the product.	Electricity consumption for a standard usage scenario and a standard configuration [1.2]
Resource management functionality	This is the capability of the software product to manage the hardware capacities (and thus the natural resources needed to provide them) in a way that avoids wasteful use.	Resource Management [1.3]
Continuity and further development of the software product	Backward compatibility of a software product mitigates the obsolescence effect that can be created by new versions.	Backward compatibility [2.1]

To provide an overview of the full set of criteria, Table 3 shows the levels one and two of the hierarchy. There are three main criteria, “Resource efficiency”, “Potential hardware operating life” and “User autonomy”. They have different numbers of sub-criteria. Some of the sub-criteria are further refined to a third level. The leaf criteria are operationalized by indicators which can be directly used to measure or qualitatively assess properties of the software. Third-level criteria and indicators are not shown in Table 3 to keep it short. Please refer to the supporting information for the full documentation.

The next section describes the application of four example criteria to demonstrate their practicability in the real word.

Table 3 *Levels one and two of the criteria tree.*
For the full description of the criteria set please refer to the supporting information to this article.

1 Resource efficiency

- 1.1 Hardware efficiency: Which hardware capacities must be available for operating the software product and what is the degree of capacity utilization during operation?
- 1.2 Energy efficiency: How much electricity does the hardware consume when the software product is used to execute a standard usage scenario?
- 1.3 Resource management: Does the software product have an energy management feature, and how effective is it when using the product in a standardized context?

2 Potential hardware operating life

- 2.1 Backward compatibility: Does the manufacturer of the software product guarantee that the current release can be executed on a reference system that is n years old?

2.2 Platform independence and portability: Can the software product be executed on different currently prevalent productive system environments (hardware and software), and can users switch between them without disadvantages?

2.3 Hardware sufficiency: Does the amount of hardware capacity used remain constant over time as the software product is developed further and additional functions are added?

3 User autonomy

3.1 Transparency and interoperability: Can users understand resource-relevant aspects of the software product with a reasonable amount of time and effort? Are they free to re-use data they produced with this software product with other software products?

3.2 Uninstallability: Can the software product be uninstalled easily, without leaving traces, and without avoidable disadvantages?

3.3 Maintenance functions: Does the software product provide easy-to-use functions permitting users to repair damage to data and programs?

3.4 Independence of outside resources: Can the software product be operated as independently as possible of resources not subject to the users' control?

3.5 Quality of product information: Does the information provided about the software product support its resource-efficient use?

4 Exemplary application of the criteria set

In order to show how the set of criteria can be used to compare software products with similar functionality in terms of resource efficiency, we will demonstrate the application of selected criteria.

The example criteria, for which we will demonstrate the operationalization here, were selected using the following requirements: at least one from each of the main criteria (level 1), potentially high relevance in terms of natural resource use, and different methodological challenges for applying the criteria. This led to the following selection:

- “Electricity consumption for a standard usage scenario and a standard configuration” [1.2],
- “Default settings supporting resource conservation” [1.3.3],
- “Backward compatibility” [2.1], and
- “Uninstallability of programs” [3.2.1]

For criteria with quantitatively measurable indicators (only the first one in our example), we use a measurement setup (following ISO/IEC 14756, as introduced in [Dirlewanger 2006], see Figure 4) to record the usage of the hardware capacities and energy consumption of a reference computer system to derive the hardware utilization and energy consumption induced by the software products. To demonstrate the applicability of the method, we used the measurement method of previous work, which used different measurement scenarios and focused on energy issues. The measurement setup is briefly described in Section 4.1.

The assessment of criteria for which no measurable indicators can be defined depends on observations of the software products' behavior, expert and user opinions, visual inspections, black-box tests and reviews of software manuals and other documents.

To evaluate the practicability of the criteria, we established a measurement procedure for each indicator. We then conducted case studies with 11 software products from the product groups "word processors", "web browsers", "content management systems", and "database systems". Thus, the four chosen software types represent three different software architecture patterns: local applications, applications with remote data processing, and server applications (see Table 4). The definition of the software architecture patterns is a result of the aforementioned prior research activities, a review of relevant literature, and expert interviews.

Table 4 *Selection of software products for the case studies*

#	**Product group**	**Software architecture pattern**	**Platform**	**Products and Licenses**
1	Word processors	Local application	Desktop/ Mobile	One proprietary and one open source word processor were selected.
2	Web browsers	Application with remote data processing	Desktop/ Mobile	One proprietary and two open source web browsers were selected
3	Content Management Systems	Application with remote data processing	Desktop/ Server	Three open source browsers were selected
4	Database systems	Server application	Server	One proprietary and two open source database systems were selected

Based on the decision to select software products representing the different software architecture patterns, we chose the specific products for the case studies. Table 4 shows the resulting selection of software product groups. In order to find the products that should be tested, the following aspects were considered:

- High installation or user count
- Long useful life
- Different user groups
- Different devices used to run the products
- Different operating systems used to run the products
- Different licenses

The specific selection is based on statistics on private and professional usage and market shares of the products. We chose from a large range of software product groups to test the applicability of the indicators over a large scope.

4.1 Example criterion "Electricity consumption for a standard usage scenario and a standard configuration"

To measure the energy consumption of software for the first sub-criterion of criterion 1.2, which is "Electricity consumption for a standard usage scenario and a standard configuration", we let the system execute the same task with a set of comparable software products and monitor the consumption behavior of the system under test (SUT) at the hardware level in a standardized test environment.

Figure 4 depicts an exemplary measurement setup, introduced in [Dick et al. 2011], that allows recording the utilization of hardware capacities induced by a software product and the resulting energy consumption. Previous work showed the comparison of the energy consumption of different configurations of a web content management system, using caching and compression technologies, and of different ways of using web browsers on two well-known tools, based on black box measurements [Dick et al. 2011]. In addition, we applied white box measurements to sorting applications and multi user web applications [Johann et al. 2012].

The software product is installed on the SUT, which can be a desktop computer or server. The workload generator then performs the tasks defined in the usage scenario (see below). The power supply of the SUT is monitored by a power meter. The SUT itself collects the data on the utilization of its hardware capacities. All data is aggregated in a centralized data storage and then analyzed.

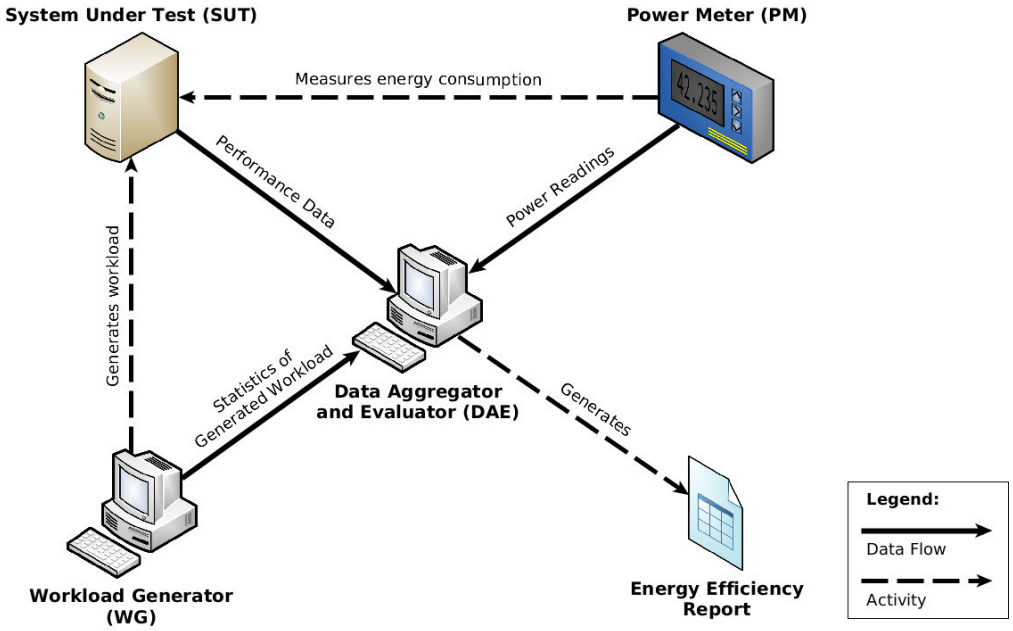

Figure 4 *Exemplary setup for measuring hardware utilization and energy consumption of a software system.*

In order to produce comparable results, we record several measurements: the baseline consumption of the operating system, the consumption of the operating system plus the software product in idle mode, and a standard usage scenario. To ensure a fair appraisal system, the standard usage scenario needs to be devised for each software product group in a way that it does not favor one product over the other. Thus, the scenarios that we describe here are merely suggestions that we used to prove the viability of the method. The creation of the scenarios, which are later used e.g. for awarding eco-labels, should be devised by the respective entity, like the certifier.

For a non-interactive product group (e.g. database systems), we propose to use or devise a benchmark that puts load on the system. This can be done in several steps (low, medium, and high load) to ensure an equitable comparison. For an interactive product group (e.g. word processors), we propose to create a standard usage scenario in a way that it emulates a user as realistically as possible. To do so, we propose the following steps: First, we analyze which tasks users typically carry out with the software product and which functions of the software are used most frequently in this process. Additionally, we consider expert opinions on functionalities which may induce high energy demand or high resource utilization. With this information, we define individual actions that then are scheduled in a flow chart. We test the scenario with each software product of the product group to ensure that the execution of the whole scenario is possible with each one. This way, when the energy consumption and hardware utilization of the SUT is measured while performing the scenario, the same useful work is done with each software product and we can compare the results for each one of the scenarios.

After the design of a standard usage scenario, we prepare the SUT. In order to reduce side effects from remnants of previous installations, we overwrite the whole hard drive with a predefined standardized disk image including the desired operating system, before installing the software product. In this system, all possible background processes – such as automatic updates, virus scanners, indexing- and backup processes – are deactivated. With this standardized configuration, we measure the baseline consumption of the operating system several times and average the results in order to ensure that the fewest possible number of other processes are interfering with the scenario measurements.

By means of a macro software, we repeat the measurement of the standard usage scenario several times in order to generate a representative sample. In the following, we present exemplary results of the measurements from the case studies. Table 5 provides some technical details.

Table 5 Details of the measurement procedure.

System Under Test (SUT)	We use two SUT, a desktop computer for local software (e.g. word processors) and a server for distributed software (e.g. content management systems).
Sampling rate	All data is collected with a sampling rate of one kilohertz. (1,000 data points/second)
Scenario length	The time interval of each scenario is set to 10 minutes.

System configuration	We use two standard configurations of hardware and operating system, one for each SUT (client and server). All software products of one product group are installed on the same reference system.
Synchronization	The analysis software synchronizes all measured data by means of time stamps.
Sample size	All measurements are repeated 30 times and then averaged. Assuming a normally distributed population [Dirlewanger 2006], and given the controlled test environment described in the text, 30 measurements are usually a sufficient sample size, as "the sampling distribution will tend to be normal regardless of the population distribution in samples of 30 or more" [Field 2009].

For each measurement, we record the power input and hardware utilization data, averaged per second. We store the measurement data together with the log data from the load generator in a database, in order to be able to analyze which action causes which resource consumption. By way of example, we present the results of the indicator "measured power input" (an indicator for criterion 1.2.1) for two word processors executing the same standard usage scenario (see Figure 5).

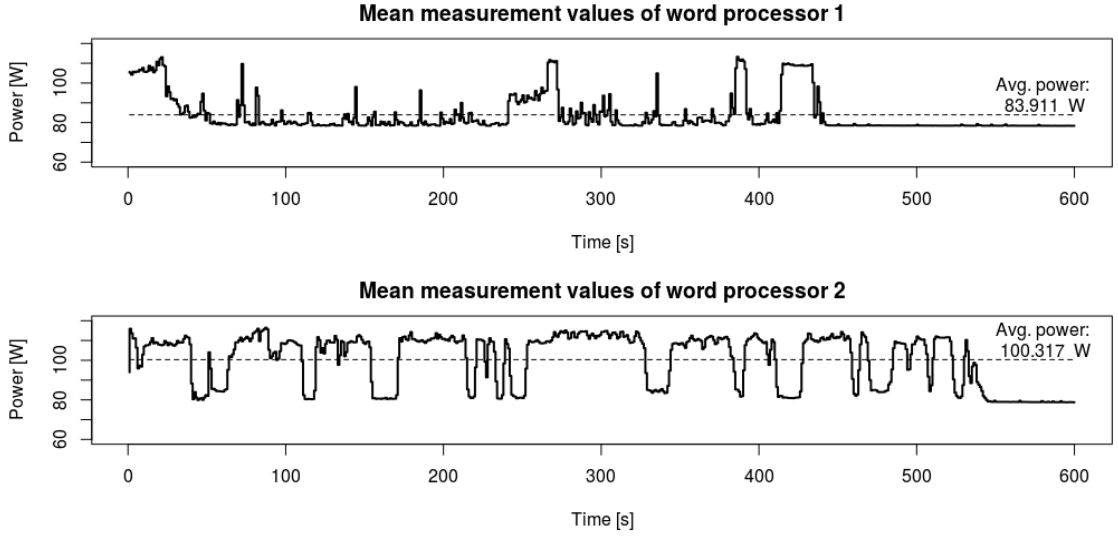

Figure 5 *Comparison of the power input of two word processors.*

The measured values are averaged per second over the 30 repetitions. It can be seen that the two word processors take a different amount of time to finish the same scenario within the 10-minutes interval.

From these measurements, we calculate the average consumed electrical energy [Wh] per standard scenario and decide whether the two candidates' mean values differ significantly via a t-test.

In this case, word processor 2 uses significantly more electrical energy (16.72 Wh) than word processor 1 (14.43 Wh) on average (confidence interval: 95%). Similarly, we also calculate and compare

the work that the software demands from the available hardware capacities, such as "processor work" (processor load integrated over time) to evaluate other criteria, in particular 1.1, "Hardware efficiency": Which hardware capacities must be available for operating the software product and what is the degree of capacity utilization during operation?

4.2 Further example criteria

All considerations about the evaluation of the criteria are made with the default settings of the software product to be assessed in order to locate the optimization potentials.

4.2.1 Example criterion "Default settings supporting resource conservation"

Criterion 1.3.3 (a sub-criterion of 1.3 Resource management) requires that "the default settings of the software product are selected in such a way that they also take the goal of resource conservation into account". This is assessed by means of observing the default settings of the software product and a reviewer's assessment. In this case, it is important to check settings both during and after the software installation and, if necessary, to measure the default settings with a separate scenario and compare the results with those of the other software products in the same group.

As mentioned in Table 2, this criterion is a special case because it is implicitly linked to other criteria and indicators addressing a broad variety of settings relevant for resource demand. Examples of such settings are sleep mode settings (indicator for 1.3.2, energy options (indicator for 1.3.1), data compression and transfer options.

As a starting point, the reviewer can look out for hardware-intensive modules identified in criterion 1.1.5 "Economical use of hardware through adaptability and support for users when adapting the software product". In some cases, it may also be necessary to rely on other assessment methods like reviewing manuals or visual inspections (e. g., in case of web browsers: which is the default page the browser opens at every start?).

For the case studies with software products from the "word processor" group, we found that for word processor 1, the default is to install all office tools (like word processor, spreadsheet program, presentation program, etc.) and some extensions (approximately 140 megabytes in size, including, e.g., dictionaries for the spell checker). Word processor 2 also includes the standard office tools. In addition, it installs several programs, plugins, and add-ons (approximately 1,000 megabytes in size).

4.2.2 Example criterion "Backward compatibility"

Criterion 2.1 requires that "the manufacturer of the software product guarantee[s] that the current release can be executed on a reference system that is n years old". The maximum number n is the result.

To apply this criterion, two indicators are evaluated:

a) Use the specification by the manufacturer (hardware, old operating systems, old frameworks), since no standard configurations have been defined for previous years.
b) When this criterion has been applied for at least one year, execute the standard usage scenario on earlier standard configurations as well. Can the standard usage scenario still be executed with the current release of the software product under the standard configurations from n years ago?

For indicator (a), we review the manufacturer's specification to identify how many years have passed since the most current version of the minimum requirements (hardware, operating systems, frameworks, etc.) to operate the software product was released. For example: If the minimum requirements are Windows 2000 and php 5 (released in 2006), this would yield n = 11 today (in 2017). For the case studies with software products from the "word processor" group, we found that for word processor 1, the current release can be executed on a system that is 8 years old. For word processor 2 we found n = 7. In the following iterations of the assessment of the software product, when a new version is available, the evaluation of indicator (b) can start by experimentally testing whether the standard usage scenario can be executed on earlier standard configurations as well. This may then provide more robust results.

4.2.3 Example criterion "Uninstallability of programs"

Criterion 3.2.1 requires that "the user receives sufficient support to uninstall the program without leaving traces". In order to test this criterion, we propose a black-box test that shows if after installing and uninstalling of the software product under study, the condition will be identical to that prior to the installation.

To achieve this, we first make a copy of the standardized disk image with exclusively the desired operating system. Then, we install the software product, perform the standard usage scenario and uninstall the software product again, following the instructions in the user manual (if available). We then create another disk image and compare it to the image we created before the installation. This way, we find all files and changes to files that remain after uninstalling. The reviewer then traces which files were created by the software product. Additionally, we search for remaining entries in the registry of Windows.

For the case studies with software products from the "word processor" group, we found that after uninstalling word processor 1, there were no related files left except user-generated documents. After uninstalling word processor 2 however, there were 14 related empty folders and 1 file remaining on the disk image and over 100 entries in the registry. The manufacturer also provides a manual for completely removing the registry entries. Indeed, this can only be accomplished with administrator rights and may be difficult for unexperienced users.

As exemplarily in Section 4 demonstrated, we verified all criteria during our case studies. In conclusion, our studies show that all criteria can be utilized with varying effort. As described, we chose these examples to be presented in the paper, because they are a cross section of the criteria set, potentially have a high impact in terms of resource consumption, and pose different methodological challenges. The general goal is not to establish a fixed methodology, but to show the viability of the criteria. The methods should be adapted by the entities that use the set of criteria e.g. for awarding eco-labels.

5 Potential use cases for the criteria set

In this section, we discuss the set of criteria and its potential future application from the perspective of different stakeholders.

5.1 Software user perspective

Software users are those who use software products without usually needing skills in developing or administrating software products. Thus, this group comprises everyone using a desktop PC, laptop, smartphone or similar end-user devices in private or professional contexts. A software user is not necessarily identical with the software purchaser (see Section 5.2).

From the perspective of software users, the criteria for sustainable software products provide information about the environmental impacts of the products. They inform about the idea how to characterize and evaluate the sustainability of a software product, and give hints how to use and configure software products to achieve higher energy and resource efficiency. Informing about these issues by exemplary demonstrations of this set of criteria (like the one presented in Section 4) might, hopefully, lead to more transparency and awareness for the topic.

Additionally, the set of criteria could be the basis for the development of a label for green software products (see Section 5.5). Such a label can be seen as an information medium supporting transparency in the relation between software usage and environmental impacts by presenting the information created by applying the set of criteria in a maximally aggregated form. The awareness for environmental impacts of software can support “greener” user behavior with regard to ICT products, especially software. Besides protecting the environment, the behavioral changes can also result in economic advantages for the users.

5.2 Software purchaser perspective

Software purchasers are those who care about searching for and buying new software products for their organizations. In private households, purchasers and users are often identical. In a company, the purchaser may be responsible for ordering vast amounts of software products. In any case, someone

must decide which software product is to be bought because there are competitive products with similar functionality.

If a purchaser is interested in sustainability issues, he or she can include our set of criteria or a selected part of it to evaluate candidate products or rely on test results provided by independent sources (NGO's, journalists) that have applied the criteria or by a public authority providing an eco-label.

Besides that, the set of criteria can inspire purchasers to include additional requirements for new software products in their calls for tenders. Such requirements can become part of procurement guidelines.

In the long run, our set of criteria can help companies reducing their CO2 footprints by purchasing sustainable software products. At the same time, there may be economic benefits by reducing hardware capacities and obsolescence.

5.3 Software administrator perspective

The software administrators do not only use the software products, but also care about configuration and related technical aspects. Thus, the set of criteria can be a source of inspiration and deeper understanding, a guideline and an argumentation aid for administrators to configure software products in an environment-friendly way. In large organizations with thousands of users, the impacts of an increased administrators' awareness for software sustainability issues can be huge both in environmental and in economic terms.

5.4 Software developer perspective

The group of software developers includes both individual developers and software companies.

For a software company, creating sustainable software and following marketing trends may be conflicting goals (e.g., creating customer lock-in effects or monitoring user behavior to sell this data might not work with sustainable software). However, both the individual developer as well as a future-oriented company, can use the idea of sustainable software to create a unique selling point. Considering our criteria in software development expands the spectrum of non-functional requirements for the software products. As soon as sustainability becomes a highly rated requirement for software purchasers, software manufacturers who are able to deliver sustainable products will have an advantage.

Recommendations for software developers that can be directly or indirectly extracted from the criteria may be transformed into a guideline for resource-efficient software development. The recommendations and the guideline can help to spread the ideas of caring about sustainability issues in software engineering.

5.5 Software certifier perspective

Software certifiers are people or organizations who are involved in developing and awarding eco-labels or sustainability labels for software products.

Software certifiers may use the set of criteria as a basis for awarding a label. In order to create guidelines on how to implement test procedures, the method of exemplary application (Section 4) could be formalized. To do so, our set of criteria provides a reliable basis.

Providing a label for sustainable software products extends the number of certified products in the context of responsible consumption. It extends the spectrum of software properties that can be certified, e.g., quality, security, and usability. As a consequence, the portfolio of software certifiers grows, strengthening their role in contributing to market transparency and supporting users in responsible behavior.

6 Discussion

The previous section already shows possible use cases for the presented set of criteria and assigns them to the different user perspectives. However, in order to be able to assess the criteria of sustainable software products in such a structured way as presented, we had to set priorities in (i) the scope, (ii) the application area, and (iii) product selection. This can be interpreted both as strength and as limitation.

As described in the introduction, we decided to focus on the usage phase of software. This reduces the complexity of gathering software products as immaterial goods and provides a starting point that is easier to handle. Nevertheless, we are aware of the connection between the different life cycle phases. Thus, we definitely speak out in favor of addressing the other phases and the connection between them in future work. Overall, the set of criteria presents a balanced selection of possible criteria since it is the result of literature reviews and working sessions of a team of seven researchers and practitioners with different backgrounds. Additionally, the selection has been evaluated by external experts.

Defining standard usage scenarios and taking this scenario as a basis for the measurements done (see Section 4.1), allows us to compare the energy consumption and hardware requirements of software products of the same product classification. However, as described, the standard usage scenarios are one viable approach to the assessment of the software products, especially for interactive product groups. Nevertheless, a certifier (as described in Section 5.5) who uses the criteria catalog must decide which scenarios are to be established within the community. To evaluate the exemplary criteria, we used a measurement setup including specific hardware and software tools (Figure 4). However, the portability of the measurement method is ensured since the results are repeatable by using comparable tools.

In order to be able to test the suggested criteria, we had to select software products the criteria could be applied to. We chose 11 products representing (a) the defined software architecture patterns and (b)

popular products of these classes. It turned out that there are some limitations in the application of the set of criteria that are caused by the software architecture. For example, local applications do not transfer data in the network. Thus, we do not need to estimate the energy consumed in the network for the data traffic. This (energy consumed in the network) is one of the indicators of the criterion 1.2 Energy efficiency (see Table 3). Overall, we are satisfied that the set of criteria can be applied to additional software products representing one of the software architecture patterns.

7 Conclusion and Outlook

In this article, we described how we developed a set of criteria for software sustainability and demonstrated the application of a subset of them. This research is one step towards awarding a sustainability label for software products by comparing products with similar functionality. If it turns out that one of the products causes less pressure on natural resources than others in its class, then it can be labelled "sustainable". The basic causal model and the criteria we developed go beyond energy demand at runtime by also covering the mechanisms that drive the increasing demand for hardware capacities, including software-induced obsolescence.

The criteria were developed with a combined approach including an extensive literature search, the selection and transfer of existing software quality criteria and the derivation of software-specific criteria from general sustainability indicators. The pilot application of core criteria addressing electricity consumption, default settings, backward compatibility and uninstallability of software products revealed that there are significant differences between products that may look quite similar at a first glance. We conclude that the criteria have a high relevance and offer practical benefit to any stakeholder who wants to distinguish similar software products with respect to their resource efficiency or environmental impact.

So far, the criteria are unweighted. The set of criteria provides no statement about what is more relevant and which minimum standards must be met to characterize sustainable software. When the criteria set should be further developed into an eco-label or procurement requirements, it will have to be expanded to a rating system. One can imagine that sustainability properties of software can finally be described, e.g., on a single scale from 0 to 100. This would make it easy to rank products according to their environmental friendliness. However, this step towards standardization should be done by experts of the certification field since it is a political decision leading to issues of environmental policy.

Next to moving forward in creating a standardized label for sustainable software products, future work contains an extension of the measurements, e.g. evaluating further software products and including mobile devices as SUT. A vision for the future is to integrate the measurements of software sustainability directly in the development process and when releasing new versions of software products. Thus, in

future, we will pay attention to providing recommendations for software engineers and an integration of the knowledge on sustainability characteristics for software products into teaching and education.

Acknowledgements

There have been many contributors to shape the set of criteria. The authors are thankful to each of them. We specifically would like to thank Marina Köhn, Dr. Hans-Jürgen Baumeister (both German Environment Agency), and Prof. Dr. Benno Schmidt from Bochum University of Applied Sciences.

This work was supported by the German Environment Agency under project number 3715 37 601 0.

References

ABENIUS, S. 2009. Green IT & Green Software - Time and Energy Savings Using Existing Tools. In *EnviroInfo 2009: Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools. proceedings of the 23rd International Conference Environmental Informatics - Informatics for environmental protection, sustainable development and risk management, September 09 - 11, 2009, HTW Berlin, University of Applied Sciences, Germany*, V. WOHLGEMUTH, B. PAGE AND K. VOIGT, Eds. Shaker, Aachen, 57–66.

AL HINAI, M., AND CHITCHYAN, R. 2015. Building social sustainability into software: Case of equality. In *Requirements Patterns (RePa), 2015 IEEE Fifth International Workshop on*, 32–38.

ALBERTAO, F. 2004. *Sustainable Software Engineering*. http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering#about. Accessed 14 April 2017.

ALBERTAO, F., XIAO, J., TIAN, C., LU, Y., ZHANG, K.Q., AND LIU, C. 2010. Measuring the Sustainability Performance of Software Projects. In *2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China*, IEEE Computer Society, Ed., 369–373.

BOZZELLI, P., GU, Q., AND LAGO, P. 2013. *A systematic literature review on green software metrics*. Technical Report: VU University Amsterdam. http://www. sis. uta. fi/ pt/TIEA5\ Thesis\ Course/Session\ 10 \ 2013\ 02\ 18/SLR \ GreenMetrics. pdf.

CALERO, C., BERTOA, M.F., AND ANGELES MORAGA, M. 2013a. A systematic literature review for software sustainability measures. In *2nd International Workshop on Green and Sustainable Software (GREENS)*, 46–53.

CALERO, C., MORAGA, M., AND BERTOA, M.F. 2013b. Towards a software product sustainability model. *arXiv preprint arXiv:1309.1640*.

CALERO, C., MORAGA, M.Á., BERTOA, M.F., AND DUBOC, L. 2015. Green Software and Software Quality. In *Green in Software Engineering*, C. CALERO AND M. PIATTINI, Eds. Springer, 231–260.

COROAMA, V.C., AND HILTY, L.M. 2014. Assessing Internet energy intensity A review of methods and results. *Environmental Impact Assessment Review 45*, 63–68. DOI: 10.1016/j.eiar.2013.12.004.

COROAMA, V.C., HILTY, L.M., AND BIRTEL, M. 2012. Effects of Internet-based multiple-site conferences on greenhouse gas emissions. *Telematics and Informatics 29*, 4, 362–374.

DICK, M., KERN, E., DRANGMEISTER, J., NAUMANN, S., AND JOHANN, T. 2011. Measurement and Rating of Software-induced Energy Consumption of Desktop PCs and Servers. In *Innovations in sharing environmental observations and information. Proceedings of the 25th International Conference EnviroInfo October 5 - 7, 2011, Ispra, Italy*, W. PILLMANN, S. SCHADE AND P. SMITS, Eds. Shaker, Aachen, 290–299.

DICK, M., NAUMANN, S., AND KUHN, N. 2010. A Model and Selected Instances of Green and Sustainable Software. In *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience. 9th IFIP TC 9 International Conference, HCC9 2010 and 1st IFIP TC 11 International Conference, CIP 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, J. BERLEUR, M. D. HERCHEUI AND L. M. HILTY, Eds. Springer, Berlin, Heidelberg, 248–259.

DIRLEWANGER, W. 2006. *Measurement and rating of computer systems performance and of software efficiency. An introduction to the ISO/IEC 14756 method and a guide to its application*. Kassel University Press, Kassel.

FIELD, A.P. 2009. *Discovering statistics using SPSS*. Sage, Los Angeles.

HERTWICH, E.G., GIBON, T., BOUMAN, E.A., ARVESEN, A., SUH, S., HEATH, G.A., BERGESEN, J.D., RAMIREZ, A., VEGA, M.I., AND SHI, L. 2015. Integrated life-cycle assessment of electricity-supply scenarios confirms global environmental benefit of low-carbon technologies. *Proceedings of the National Academy of Sciences 112*, 20, 6277–6282. http://www.pnas.org/content/112/20/6277.full.

HILTY, L., LOHMANN, W., BEHRENDT, S., EVERS-WÖLK, M., FICHTER, K., AND HINTEMANN, R. 2015. Green Software. Final report of the project: Establishing and exploiting potentials for environmental protection in information and communication technology (Green IT). Report commissioned by the Federal Environment Agency, Berlin, Förderkennzeichen 3710 95 302/3, 23.

HILTY, L.M., AND AEBISCHER, B. 2015. ICT for Sustainability: An Emerging Research Field. In *ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 3–36.

HISCHIER, R., COROAMA, V.C., SCHIEN, D., AND ACHACHLOUEI, M.A. 2015. Grey energy and environmental impacts of ICT hardware. In *ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 171–189.

JOHANN, T., DICK, M., KERN, E., AND NAUMANN, S. 2012. How to Measure Energy-Efficiency of Software: Metrics and Measurement Results. In *Proceedings of the First International Workshop on Green and Sustainable Software (GREENS) 2012. held in conjunction with ICSE 2012, The International Conference on Software Engineering, June 2-9, Zurich, Switzerland*, IEEE, Ed. IEEE Computer Society, 51–54.

JOHANN, T., AND MAALEJ, W. 2013. Position Paper: The Social Dimension of Sustainability in Requirements Engineering. In *Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems*.

KERN, E., DICK, M., NAUMANN, S., AND FILLER, A. 2015. Labelling Sustainable Software Products and Websites: Ideas, Approaches, and Challenges. In *Proceedings of EnviroInfo and ICT for Sustainability 2015. 29th International Conference on Informatics for Environmental Protection (EnviroInfo 2015) and the 3rd International Conference on ICT for Sustainability (ICT4S 2015)*. Copenhagen, September 7 - 9, 2015, V. K. JOHANNSEN, S. JENSEN, V. WOHLGEMUTH, C. PREIST AND E. ERIKSSON, Eds. Atlantis Press, Amsterdam, 82–91.

KERN, E., DICK, M., NAUMANN, S., GULDNER, A., AND JOHANN, T. 2013. Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In *ICT4S ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 87–94.

MAHAUX, M., HEYMANS, P., AND SAVAL, G. 2011. Discovering Sustainability Requirements: An Experience Report. In *Requirements Engineering: Foundation for Software Quality. 17th International Working Conference, REFSQ 2011, Essen, Germany, March 28-30, 2011*, Proceedings, D. M. BERRY AND X. FRANCH, Eds. Springer, Berlin, Heidelberg, 19–33.

NAUMANN, S., DICK, M., KERN, E., AND JOHANN, T. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *SUSCOM 1*, 4, 294–304. doi:10.1016/j.suscom.2011.06.004.

PENZENSTADLER, B. 2013. Towards a Definition of Sustainability in and for Software Engineering. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 1183–1185.

PENZENSTADLER, B., BAUER, V., CALERO, C., AND FRANCH, X. 2012. Sustainability in software engineering: A systematic literature review.

PENZENSTADLER, B., RATURI, A., RICHARDSON, D., CALERO, C., FEMMER, H., AND FRANCH, X. 2014. *Systematic Mapping Study on Software Engineering for Sustainability (SE4S) - Protocol and Results* ICS2 221. Institute for Software Research, University of California, Irvine, Irvine.

SOM, C., HILTY, L.M., AND KÖHLER, A.R. 2009. The precautionary principle as a framework for a sustainable information society. *Journal of Business Ethics 85*, 493–505. DOI 10.1007/s10551-009-0214-x.

TAINA, J. 2011. Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In *Green ICT: Trends and Challenges*, J.-C. LOPEZ-LOPEZ, G. SISSA AND L. NATVIG, Eds., 22–27.

TURCONI, R., BOLDRIN, A., AND ASTRUP, T. 2013. Life cycle assessment (LCA) of electricity generation technologies: overview, comparability and limitations. *Renewable and sustainable energy reviews 28*, 555–565. http://uni-obuda.hu/users/grollerg/LCA/hazidolgozathoz/lca-electricity%20generation%20technologies.pdf.

WÄGER, P.A., HISCHIER, R., AND WIDMER, R. 2015. The material basis of ICT. In *ICT Innovations for Sustainability*. *Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 209–221.

Sustainable software products – towards assessment criteria for resource and energy efficiency

– *Supplementary material* –

Set of criteria for sustainable software

Lorenz Hilty, Stefan Naumann, Yuliyan Maksimov, Eva Kern,
Andreas Filler, Achim Guldner, Jens Gröger

Version 01
Effective: 31.05.2017
We will provide updates of the set of criteria for sustainable software on:
https://green-software-engineering.de/en/kriterienkatalog-v01

This document presents results of the following research project:
UFOPLAN research project „Sustainable software design – Development and application of criteria for resource-efficient software products with consideration of existing methods.“ by order of the German Federal Environmental Agency, code number: 3715 37 601 0

Project contact persons

Dipl.-Ing. Jens Gröger	Oeko-Institut e.V. j.groeger@oeko.de
Prof. Dr. Stefan Naumann	Trier University of Applied Sciences, Environmental Campus Birkenfeld s.naumann@umwelt-campus.de
Prof. Dr. Lorenz Hilty	University of Zurich hilty@ifi.uzh.ch

***Suggested citation**: Suggested: Kern, Eva; Hilty, Lorenz M.; Guldner; Achim, Maksimov, Yuliyan V.; Filler, Andreas; Gröger, Jens; Naumann, Stefan.Sustainable software products – towards assessment criteria for resource and energy efficiency. Future Generation Computer Systems [accepted for publication], ISSN: 0167-739X*

Table of contents

1 Resource efficiency

To what extent are hardware capacities used, and therefore, to what extent are natural resources consumed indirectly, when a given function is performed?

This main criterion assumes that a given functionality can be fulfilled by a software product using different amounts of hardware capacities, which indirectly results in different amounts of natural resource consumption required for hardware provision, operation, and disposal.

The ideal is a software product that achieves a given functionality with minimum resource consumption, i.e., that maximizes resource efficiency (see glossary). Functionality is specified by standard usage scenarios (see glossary). The hardware capacities to be made available and those actually used as well as the energy consumed serve as approximations for estimating natural resource consumption.

1.1 Hardware efficiency

Which hardware capacities must be available for operating the software product and what is the degree of capacity utilization during operation?

Hardware capacities are measured in % of the corresponding capacity of a reference system[5]. They can be differentiated according to two dimensions: (Table 1). On one dimension, they are differentiated in local, network and remote capacities. Here, we further distinguish in recommended (1.1.1) and minimum (1.1.2) capacities as well as capacities required in idle mode (1.1.3) and during the execution of a standard usage scenario (1.1.4). On the other dimension, we differentiate according to the type of hardware capacity: processing power, working memory, permanent storage, bandwidth, and display resolution. The matrix is open to the addition of new columns in case new categories of hardware will become relevant in the future.

Table 1-1 Differentiation of hardware capacities in two dimensions. The numbers refer to the criteria explained in the following sections, the letters refer to the indicators.

		Processing power	Working memory	Permanent storage	Bandwidth	Display resolution
Local	recommended	1.1.1 a)	1.1.1 b)	1.1.1 c)	-	1.1.1 d)
	minimum	1.1.2 a)	1.1.2 b)	1.1.2 c)		1.1.2 d)
	idle	1.1.3 a)	1.1.3 b)	1.1.3 c)		
	standard usage	1.1.4 a)	1.1.4 b)	1.1.4 c)		
Network	recommended	-	-	-	1.1.1 e)	-
	minimum				1.1.2 e)	
	idle				1.1.3 d)	
	standard usage				1.1.4 d)	
Remote	recommended	1.1.1 f)	1.1.1 g)	1.1.1 h)	-	-
	minimum	1.1.2 f)	1.1.2 g)	1.1.2 h)		
	idle	1.1.3 e)	1.1.3 f)	1.1.3 g)		
	standard usage	1.1.4 e)	1.1.4 f)	1.1.4 g)		

[5] Application of the set of criteria requires that a reference system corresponding to current technical developments is determined periodically. The reference system serves to standardize indicators.

Each cell of the matrix in Table 1-1 shows the associated criterion (e.g., 1.1.1) with the corresponding indicator (e.g. a)) for operationalization. The criteria and indicators will be described in the following sections, which are numbered accordingly. Not all of the criteria 1.1.1 to 1.1.4 are applicable in all the matrix cells. For this reason, some of the cells remain empty.

Criteria 1.1.5 and 1.1.6 are used for the assessment of hardware efficiency as well. They can be assessed in general terms; they do not require differentiation according to this matrix and do not show up in Table 1-1 for this reason.

When these criteria are to be aggregated later, the principal problem arises that a trade-off between different hardware capacities (local vs. remote, processing power vs. working memory, processing power for data compression vs. bandwidth, etc.) must be made. If it were possible to evaluate the hardware capacities in the form of an ecological footprint, they could be weighted and aggregated in that regard. Assessing this footprint is not part of the work reported here; we refer the reader to existing life cycle inventories for ICT hardware and electric energy as a basis for aggregation.

Table 1-2 Basic definitions for the measurement of the criteria 1.1.3 and 1.1.4.

Identifier	Name	Definition	Comment
FL_i	full load	Upper limit of the capacity i in the reference system.	For processing power, the FL is 100%, for working memory the sum of the installed RAM, for network bandwidth the maximum transmission speed, etc.
BL_i	base load	Average load of the capacity i in the reference system when the software product under study is not installed	
IL_i	idle load	Average load of the capacity i in the reference system when the software product under study is installed, but idle.	Idle load includes base load.
NIL_i	net idle load	$NIL_i = IL_i - BL_i$	
t	time	Time needed to execute the standard usage scenario on the reference system.	Begins with the start of the standard usage scenario and ends when all required actions are executed, including follow-up processes (such as releasing memory or deleting temporary files).
GL_i	gross load	Load of the capacity i in the reference system while executing the standard usage scenario, measured as time-weighted average over t.	
NL_i	net load	$NL_i = GL_i - BL_i$	
AF_i	allocation factor	$AF_i = NL_i / (FL_i - BL_i)$	Allocation factor used to assign a share of the base load GA to the effective load EL (defined below).

AFI_i	allocation factor idle	$AFI_i = NIL_i / (FL_i – BL_i)$	Allocation factor used to assign a share of the base load GA to the effective load idle ELI (defined below).
EL_i	effective load	$EL_i = NL_i + AF_i * BL_i$	
EIL_i	effective load idle	$EIL_i = NIL_i + AFI_i * BL_i$	Used to calculate the indicators for hardware demand of criterion 1.1.3
HD_i	hardware demand	$HD_i = EL_i * t$	Used to calculate the indicators for hardware demand of criterion 1.1.4

For practical purposes, it is sufficient to calculate the allocation factors AF and AFI for criteria 1.1.3 and 1.1.4, in particular for processing power (indicators a. and e.) and working memory (indicators b. and f.). For all other indicators (c., d., g.) the allocation factors can be set to zero, i.e., it can be assumed that EL = NL and EIL = NIL for simplicity.

Figure 1-1 illustrates the process of measuring hardware capacity load by executing a standard usage scenario.

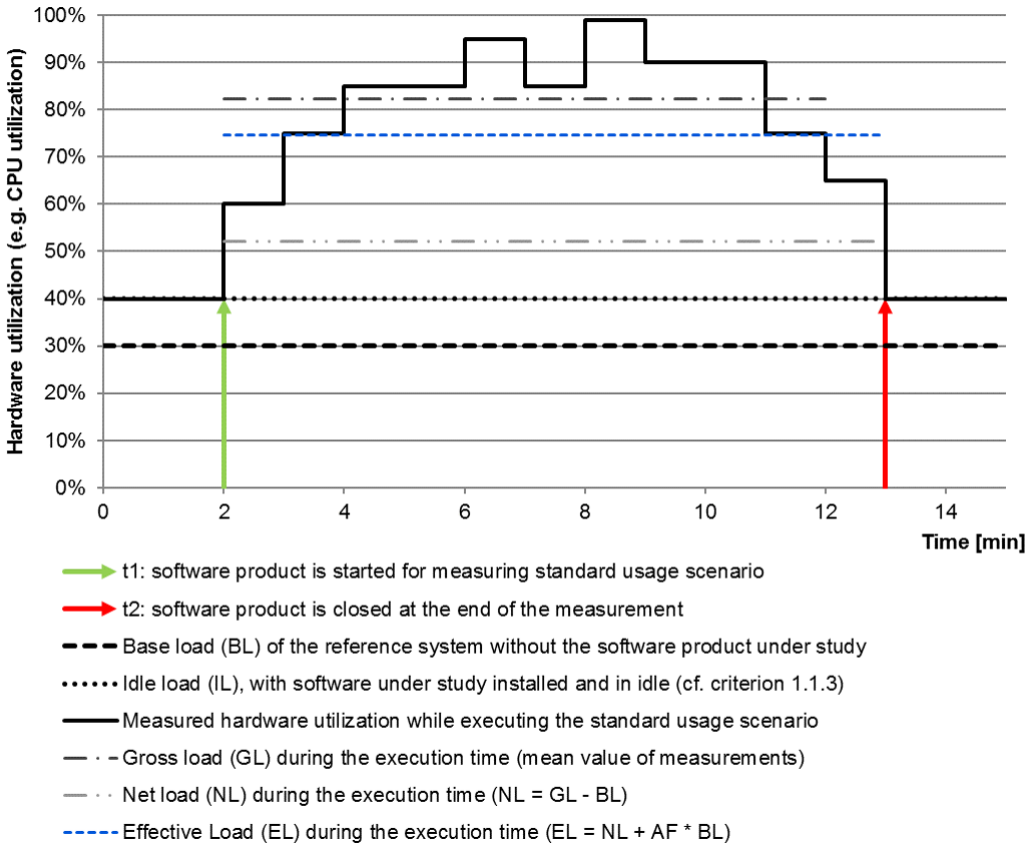

Figure 1-1 Exemplary measurement process of hardware capacity load

1.1.1 Recommended system requirements and resulting hardware requirements (including peripheral devices)

Which system requirements does the manufacturer recommend for operating the software product?

Indicators:

a) Recommended local processing power as specified by the manufacturer in % of the processing power of the reference system
b) Recommended local working memory as specified by the manufacturer in % of the working memory of the reference system
c) Recommended local permanent storage as specified by the manufacturer in % of the permanent storage of the reference system
d) Recommended display resolution as specified by the manufacturer in % of the display resolution of the reference system
e) Recommended network bandwidth as specified by the manufacturer in % of the network bandwidth of the reference system
f) Recommended server processing power as specified by the manufacturer in % of the processing power of the reference system
g) Recommended server working memory as specified by the manufacturer in % of the server working memory of the reference system
h) Recommended server permanent storage as specified by the manufacturer in % of the server permanent storage of the reference system

1.1.2 Minimum system requirements and resulting hardware requirements (including peripheral devices)

What are the minimum system requirements for operating the software product?

Indicators:

a) Minimum local processing power as specified by the manufacturer in % of the processing power of the reference system
b) Minimum local working memory as specified by the manufacturer in % of the working memory of the reference system
c) Minimum local permanent storage as specified by the manufacturer in % of the permanent storage of the reference system
d) Minimum display resolution as specified by the manufacturer in % of the display resolution of the reference system
e) Minimum network bandwidth as specified by the manufacturer in % of the network bandwidth of the reference system
f) Minimum server processing power as specified by the manufacturer in % of the processing power of the reference system
g) Minimum server working memory as specified by the manufacturer in % of the server working memory of the reference system
h) Minimum server permanent storage as specified by the manufacturer in % of the server permanent storage of the reference system

1.1.3 Hardware utilization in idle mode assuming a standard configuration

What is the level of utilization of the available hardware capacities by the software product in idle mode?

Indicators:

a) Measurement of average processor utilization in idle mode under the standard configuration
b) Measurement of average working memory utilization in idle mode under the standard configuration
c) Measurement of average permanent storage utilization in idle mode under the standard configuration
d) Measurement of average bandwidth utilization for network access in idle mode under the standard configuration
e) Measurement of average server processor utilization in idle mode under the standard configuration
f) Measurement of average server working memory utilization in idle mode under the standard configuration
g) Measurement of average server permanent storage utilization in idle mode under the standard configuration

Average processor load (indicators a. and e.) and average working memory load (indicators b. and f.) are calculated as effective idle load EIL (see Table 1-2).

1.1.4 Hardware utilization during normal use assuming a standard configuration and a standard usage scenario

What is the average utilization of the available hardware capacities during operation of the software product?[6]

It should be noted here that utilization of hardware capacities is understood as a variable integrated over time. If, for example, program A requires twice as much processing power, working memory, or bandwidth as program B to accomplish a given standard usage scenario, but makes the processor, memory, or bandwidth available again after half the period of time required by B, then according to this criterion, A is not less efficient than B. (This is not the case for criteria 1.1.1 to 1.1.3.) Thus, the use of acceleration technologies is not penalized by this criterion.

Indicators:

a) Measurement of average processor utilization when running the standard usage scenario under the standard configuration
b) Measurement of average working memory utilization when running the standard usage scenario under the standard configuration
c) Measurement of average permanent storage utilization when running the standard usage scenario under the standard configuration
d) Measurement of average bandwidth utilization for network access when running the standard usage scenario under the standard configuration
e) Measurement of average server processor utilization when running the standard usage scenario under the standard configuration

[6] Average capacity utilization determines which free hardware capacities can be used by other software products during operation of the software product.

f) Measurement of average server working memory utilization when running the standard usage scenario under the standard configuration
g) Measurement of average server permanent memory utilization when running the standard usage scenario under the standard configuration

Hardware demand for processor load (indicators a. and e.) and average working memory load (indicators b. and f.) are calculated as defined in Table 1-2.

1.1.5 Economical use of hardware through adaptability and support for users when adapting the software product

Does the software product use only those hardware capacities required for running the functions demanded by the individual user? Does the software product provide sufficient support when users adapt it to their needs?[7]

Indicators:

a) Does the software automatically minimize the required capacities and/or are there relevant options available during installation? (Scale: yes/no)
b) If users choose an option, can they change the decision for or against installation options at any later point in time? (Scale: yes/ no)
c) Black box test whether hardware-intensive modules can be disabled (Scale: can permanently be disabled/can temporarily be disabled/cannot be disabled)
d) Is it possible (without drawbacks) to disable peripheral devices that are not needed temporarily or permanently or to avoid providing them at all? (Scale: can be disabled temporarily and permanently/can be disabled only temporarily/cannot be disabled)
e) Will files used only for installing the product be deleted after installation?

1.1.6 Online delivery

Can the software product (including all programs, data, and documentation including manuals) be purchased, installed, and operated without transporting physical storage media (including paper) or other materials goods (including packaging)?

Indicators:

a) Can the software be delivered and updated online?
b) Is it supported that the user organization can store the software product and its updates on a local server, avoiding transferring them for every single user?

[7] No utilization of capacities by functionality temporarily or permanently not demanded by the user.

1.2 Energy efficiency

How much electricity does the hardware consume when the software product is used to execute a standard usage scenario?[8]

The consumption of electric energy is a consequence of the utilization of hardware capacities. How to measure hardware utilization has already been described in section 1.1.4 above. In parallel to those measurements, the electrical power demanded by the hardware should be measured (or estimated) as well, at least for the entirety of hardware used locally, for data transmission in the network or remotely, respectively.

Indicators:

a) Measurement of the energy consumed on the local device for running the standard usage scenario under the standard configuration
b) Estimation of the energy consumed in the network for the data traffic caused by running the standard usage scenario under the standard configuration (a current estimate of network energy intensity in kWh/GB from literature may be used, if necessary differentiated among types of access network)
c) Measurement of the energy consumed by servers for the remote processing and storage for running the standard usage scenario under the standard configuration (if measurement not possible, an estimate based on factors for average energy intensity of data center services from literature may be used)

The electric energy consumed is the integral of electric power over the time needed for the execution of the standard usage scenario. Departing from the specifications provided to measure hardware load (section 1.1.4), only net indicators will be used for the energy measurements (indicators a. and c.), i.e., only the quantity that exceeds the level of the electric base load. This is done to increase practicability (calculating an allocation factor for electricity may be difficult because a true upper limit for electric power is sometimes not known). It also adds to the clarity of the results of the energy measurements if base load energy is not included when comparing software products.

1.3 Resource management

To what extent does the software product contribute to efficient management of the resources it uses during operation?

Since the extent to which a given software product is used may vary, adaptive demand for hardware capacities that is supported by the software product contributes to resource conservation. Hardware capacities not in use can potentially be used by other processes or reduce their energy consumption. Both options contribute indirectly to natural resource conservation.

In contrast to criteria 1.1 and 1.2, this criterion refers to adapting the demand for hardware capacities at the program's runtime, in particular the transition to less energy-consuming modes, dependent on the current user requirements or the available hardware capacities or energy. In other words, while resource efficiency in the various modes was addressed by criteria 1.1 and 1.2, the focus here is on the ability to switch between modes depending on context.

[8] Use of electricity is a consequence of the use of hardware capacities already discussed in section 1.1. This implies that this criterion is redundant. However, the redundancy is desired since energy can be measured separately and not all sub criteria of hardware efficiency (1.1) are operationalizable.

1.3.1 Adaptation of hardware capacities used to current demand

Does the software product have the feature to release hardware capacities (and reduce energy consumption as a consequence) when it doesn't temporarily use these capacities?

Indicators:

a) Does the software product have different modes which have a measurable effect on energy consumption?
b) Does the software product dynamically change to a more energy saving mode when possible (e.g. sleep mode)?
c) In case the user has to make energy-relevant settings, are these settings concentrated in one place and easily understandable for the user?[9]

1.3.2 Adaptation of hardware capacities used to current supply

Is the software product able to dynamically adapt its demand for hardware capacities and energy when the supply is changing? (e.g., when the available bandwidth is decreasing or battery is low)

Indicators:

a) Does the software product switch to a more economical mode when less hardware capacity or energy is available, avoiding errors or loss of data? (no restrictions, slower execution, error during execution)
b) Is the full software functionality available in if the energy management of lower system layers or connected client systems is activated?[10]

1.3.3 Default settings supporting resource conservation

Are the default settings of the software product selected in such a way that they also take the goal of resource conservation into account?[11]

Indicators:

a) Reviewer's assessment whether the default settings of the software product are selected in such a way that they also take the goal of resource conservation into account

1.3.4 Feedback on use of hardware capacities and energy

Can the local and remote hardware capacities used by the software product and their resulting energy consumption be monitored, and are the displayed values correct?

Indicators:

a) Are the hardware capacities in use, data flow, and energy consumption displayed? (Scale: yes/to some extent/no)

---

[9] Examples: Background/sleep settings, animations, computing-intensive processes such as indexing etc., cache sizes, ability to select the time at which processes are executed to take advantage of ecologically more beneficial energy (demand shaping).

[10] In particular server-based software should avoid that activating the energy management on client side hampers the functionality. For example, no session information should be lost if the client computer enters sleep mode.

[11] Example: Default setting for printing: Double-sided printing if the printer has this capability?

b) Assessment by the reviewer whether the display is correct

2 Potential hardware operating life

To what extent are hardware replacement cycles decoupled from software replacement cycles?[12]

Software imposes requirements on the hardware on which it is executed. The faster these requirements increase as the software product is developed further, and the more specific they are, the more they limit the use of hardware products already in existence. If existing hardware products cannot be used, or can no longer be used, to execute the given software product, then this shortens the operating life of the hardware.

The ideal is a software product whose development dynamics permit operators to manage their hardware products independently of these dynamics, i.e., decouple hardware management from software management.

2.1 Backward compatibility

Does the manufacturer of the software product guarantee that the current release can be executed on a reference system that is *n* years old?[13]

Indicators:

a) Initially use the specification by the manufacturer (hardware, older operating systems, older frameworks), since no standard configurations have been defined for previous years.
b) When this criterion has been applied for a long enough time period, so that the standard usage scenario can also be executed on earlier standard configurations as well: Can the standard usage scenario still be executed with the current release of the software product on a configuration that was the standard configurations n years ago (n still needs to be specified)?

[12] Decoupling software and hardware replacement cycles amounts to long potential hardware operating life. Basic assumption: Every software product requires a system environment as the platform on which it is executed. The system environment is defined as the sum of the hardware and software components of the ICT system that are required for executing the software product. The software product itself can be part of the system environment of other software products. Example: A web browser requires an operating system, additional system software, and hardware as a system environment, and at the same time it constitutes the system environment for a web application. From the perspective of a given software product, the following question is crucial to understand its influence on hardware operating life: when the software product is replaced by a newer version, which requirements to the lowest level—the hardware—does this generate via the intermediate levels of the system environment?

[13] Thus, the software product can be executed on a standard hardware configuration that has already been in operation for *n* years.

2.2 Platform independence and portability

Can the software product be executed on different currently prevalent productive system environments (hardware and software), and can users switch between them without disadvantages?[14]

Indicators:

a) Manufacturer specifications (compatible with various operating systems, runtime environments).
b) Execute standard usage scenario on various currently prevalent productive system environments and check for portability of data and software settings.

2.3 Hardware sufficiency

Does the amount of hardware capacity used remain constant over time as the software product is developed further and additional functions are added?

This criterion rewards software manufacturers who make it easy for their customers to continue to use their existing hardware. It intentionally does not take into account whether functionality is expanded. Sufficiency means that the amount of resources required will *not* increase even if the utility they provide increases (which is possible, after all, because of increasing efficiency).

The ideal is a software product that fulfills more and more requirements from one version to the next, but nonetheless does not increase its hardware requirements.

This criterion can be applied only when products have already been assessed several times, i.e., when at least one previous result is available.

Indicators:

a) intertemporal comparisons with the following imaginable results:

1. "very good": To date, new versions have resulted in a decrease in the hardware capacities required.
2. "good": To date, new versions have resulted in no increase in the amount of hardware capacities required.
3. "sufficient": Although to date, new versions have increased the amount of hardware capacities required, the increases have not overcompensated for the efficiency improvements due to technical factors as exhibited by the succession of reference systems over time.
4. "insufficient": Because of new versions, the required hardware capacities have increased faster than technical efficiency.

[14] We recommend that this criterion should not be considered one of the minimum requirements because in principle, there could be very resource-efficient software that runs on just one platform. Nonetheless, platform independence is to be considered beneficial since it gives users more freedom when optimizing procurement of hardware and system software.

3 User autonomy

Does the manufacturer of the software product respect user autonomy in dealing with the purchased product?

This main criterion assumes that a relevant number of users is interested in using software in a resource-efficient way. If they can do so without functional disadvantages, they will try to work with a small amount of hardware capacity (which they generally pay for) and keep energy consumption low (which is also financially relevant or at least impacts the battery life of mobile devices). However, users can do so only if they are not forced to consume unnecessary amounts of resources and if they understand how they can avoid unnecessary resource consumption.

The ideal is a software product that respects the freedom of users to decide about utilizing hardware capacities (and thus indirectly about using resources) when using the product, as far as possible.

The following criteria are to be evaluated from the perspective of target groups that are not technical specialists; in other words, they will generally not be fulfilled simply by the fact that an expert can fulfill them. Criterion 3.1.2 is an exception in this regard.

3.1 Transparency and interoperability

Can users understand resource-relevant aspects of the software product with a reasonable amount of time and effort? Are they free to re-use data they produced with this software product with other software products?

3.1.1 Transparency of data formats and data portability

Is sufficient documentation provided for the data formats (file or data stream formats) used by the software product to enable interoperability? Do the data formats comply with open standards enabling further use of the data with another software product?[15]

To apply this criterion, it must first be defined which standards are considered open standards at the time of awarding a label.

Indicators:

a) Review of manuals and technical data sheets, comparison with known open standards
b) Check of compliance with known and open standards.

[15] This is decisive to prevent customer lock-in (dependence on the software product), which may force unnecessary resource consumption, both in the case of retaining an inefficient product and in the case of switching to a different product, which may require resources as well.

3.1.2 Transparency and interoperability of the programs

Are application programming interfaces (APIs) clearly documented, and are dissemination and further development of the program supported? Do the interfaces comply to open standards to enable interoperability?

Weighting of indicators may be highly dependent on context. The effects of open source code and licensing models on resource use cannot be assessed in terms of a general rule.

Indicators:

a) If APIs exist: Review of the documentation of the interfaces on the basis of the documentation of the software product and its APIs
b) Is the source code open?
c) Is the software released under a license that allows it to further develop it?

3.1.3 Continuity of the software product

Can the software product be used for longer periods of time without serious negatives (in particular IT security problems) occurring, and does the user have the option to avoid unnecessary updates?[16]

Indicators:

a) How long is the time period for which the supplier guarantees future support for the product, including security updates?
b) Does the manufacturer respond promptly when security gaps (vulnerabilities) become known?
c) Can the user influence the frequency of updates by configuring the software product and when doing so differentiate between security updates and other updates?
d) Is it possible to receive differential updates only?[17]

3.1.4 Transparency of task management

Does the software product inform users that it is automatically launching or running tasks in the background that are possibly not being used?

Indicators:

a) On the basis of the installation and the execution of standard usage patterns, test which processes are automatically launched by the software product and whether it informs users of this (Scale: informs users of all such processes/informs users of some such processes/does not inform users)
b) If the software product is automatically launched at system start ("autostart"): does it inform users that this is the case?

[16] A high frequency of updates causes resource consumption and makes it more difficult to maintain transparency. It is difficult to define the "necessity" of updates objectively; however, it makes at least sense to differentiate between security-relevant (and thus doubtless necessary) updates and other updates; this is addressed by indicator b).

[17] This avoids replacing the entire program, which can cause significant resource consumption if performed frequently.

c) If the user carries out an action that can be understood as ending the program, but at least one of the tasks remains active: does the software product inform the user that this is the case?

3.2 Uninstallability

Can the software product be uninstalled easily, without leaving traces, and without avoidable disadvantages?

3.2.1 Uninstallability of programs

Does the user receive sufficient support to uninstall the program without leaving traces?

Indicators:

a) Uninstallation of the software and comparison with the condition prior to installation, which must be identical.

3.2.2 Capability to erase data

Does the user receive sufficient support when erasing data generated during operation of the software product as desired?

This criterion is intended specially to avoid the case that compliance with high IT security standards following uninstallation of the software product can be guaranteed only by physically destroying hardware.

Indicators:

a) After erasing of the data explicitly stored by the user and comparison with the condition prior to installation, are the two states identical in relevant respects?
b) Does the software product provide transparency about the places where it stores data?
c) Is the user supported in erasing data stored on remote storage devices without leaving traces?

3.3 Maintenance functions

Does the software product provide easy-to-use functions permitting users to repair damage to data and programs?

3.3.1 Recoverability of data

Can the data be recovered in its last condition following an abnormal termination?

Indicators:

a) Does the manufacturer provide specifications and can they be validated by means of a test?
b) Can the user set the periodicity at which changes are automatically saved?

3.3.2 Self-recoverability

Can the installed instance of the software product be recovered following the occurrence of an inconsistent state?

Indicators:

a) Manufacturer specifications and review by means of a test

3.4 Independence of outside resources

Can the software product be operated as independently as possible of resources not subject to the users' control?

3.4.1 Offline capability

To what extent does the software product avoid forced connectivity that is not necessary for providing the functionality?[18]

Indicators:

a) Testing on the basis of the standard usage scenario (Scale: offline operation possible/possible with limitations/impossible)

3.5 Quality of product information

Does the information provided about the software product support its resource-efficient use?

3.5.1 Comprehensibility and manageability of product documentation, licensing conditions, and terms of use

Is all the information easy for users to understand?

Indicators:

a) Inspection by reviewers; test with actual users

[18] Examples of unnecessarily forced connectivity: establishing a connection to the license server, repeated download of fonts required.

3.5.2 Resource relevance of product information

Does the product information include everything that users need to minimize resource consumption by the software product in a structured form, and is the information correct?

The long-term goal is to develop standardized product descriptions for resource-relevant product information. As soon as a satisfactory standard exists in this regard, compliance with it can be included as an indicator.

Indicators:

a) Qualitative assessment of completeness and comprehensibility
b) Does the product information refer to the current version of the product?
c) Inspection whether the information is correct (information is conclusive / partially conclusive / non-conclusive)

Glossary

Energy efficiency: Generally, the amount of "useful work" divided by the amount of energy it requires. In the context of this document, "useful work" is operationalized as the successful execution of standard usage scenarios.

Hardware: The material goods required to run programs or to store or transport data.

Hardware capacity: Quantifiable characteristic of a hardware system which represents its performance limit on a given dimension of performance (e.g., working memory capacity, computing power, bandwidth).

Hardware system: Delimitable unit of hardware that performs defined functions.

Indicator: An empirically determinable quantity that provides insight into a matter that cannot be measured directly. The indicators proposed in this document have different levels of measurement. In some cases, researchers will have to settle for an ordinal scale (e.g., "insufficient", "sufficient", "good", "very good", or even merely "fulfilled", "not fulfilled") to avoid giving the false impression of non-existent precision arising from a cardinal scale.

Reference system: A hardware system that is defined as generally customary in terms of its most important capacities (e.g., working memory, processor performance) during a defined period of time (e.g., one year). The purpose of the reference system is to be able to express indicators such as "minimum local memory" in relation to a reference value (currently "customary" memory).

Resource: In the context of this document, a natural resource, in particular a raw material, a form of energy, or also the capacity of an environmental medium to absorb emissions. To differentiate natural resources from technical ones, especially hardware resources, the more precise term "hardware capacities" is used here for the latter. Since using hardware capacities always results in using natural resources, this distinction (which ultimately amounts to a definitionally difficult differentiation between the ecosphere and the technosphere) is not of decisive importance here.

Resource efficiency: Generally, the amount of "useful work" divided by the amount of resources it requires. In the context of this document, "useful work" is operationalized as the successful execution of standard usage scenarios.

Software: Programs and data in digital form.

Software product: A delimitable unit of programs and data for which a license is available.

Standard configuration: A set of conditions, defined as a reference, under which a given software product is run; it includes the parameter settings selected during installation or operation, the system software provided, potentially additional software products required for operation, as well as the reference system at the hardware level.

Standard usage scenario: A usage scenario that is used for testing a software product and is supposed to be as representative as possible for the customary use case.

Usage pattern: Abstracted form of a sequence of interactions with a given software product.

Usage scenario: Description of a usage pattern which is generally machine executable.

Bibliography

Abdullah, Rusli; Abdullah, Salfarina; Din, Jamilah; Tee, Mcxin; others (2015): A Systematic Literature Review of Green Software Development in Collaborative Knowledge Management Environment. In: International Journal of Advanced Computer Technology (IJACT) 9, S. 136.

Abdullah, Rusli; Abdullah, Salfarina; Tee, Mcxin (2014): Web-based knowledge management model for managing and sharing green knowledge of software development in community of practice. In: Software Engineering Conference (MySEC), 2014 8th Malaysian. IEEE, S. 210–215.

Afgan, Naim Hamdia (2010): Sustainability paradigm: intelligent energy system. In: Sustainability 2 (12), S. 3812–3830.

Afzal, Shehla; Saleem, M. Faisal; Jan, Fahad; Ahmad, Mudassar (2013): A Review on Green Software Development in a Cloud Environment Regarding Software Development Life Cycle:(SDLC) Perspective. In: International Journal of Computer Trends and Technology (IJCTT) 4 (9), S. 3054–3058.

Agarwal, Shalabh; Nath, Asoke; Chowdhury, Dipayan (2012): Sustainable Approaches and Good Practices in Green Software Engineering. In: IJRRCS 3 (1), S. 1425–1428. Available online scholarlyexchange.org, zuletzt geprüft am 15.03.2012.

Ahmad, Ruzita; Baharom, Fauziah; Hussain, Azham (2014): A Systematic Literature Review on Sustainability Studies in Software Engineering. In: Proceedings of KMICe. Knowledge Management International Conference (KMICe) 2014. Malaysia, 12 – 15 August 2014.

Albertao, Felipe (2004): Sustainable Software Engineering. Carnegie Mellon University Silicon Valley. Available online www.scribd.com, zuletzt aktualisiert am 04.09.2008, zuletzt geprüft am 30.11.2010.

Albertao, Felipe; Xiao, Jing; Tian, Chunhua; Lu, Yu; Zhang, Kun Qiu; Liu, Cheng (2010): Measuring the Sustainability Performance of Software Projects. In: IEEE Computer Society (Hg.): 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China. Technical Committee on Electronic Commerce (TCEC), S. 369–373. Available online doi.ieeecomputersociety.org, zuletzt geprüft am 04.03.2011.

Amsel, Nadine; Ibrahim, Zaid; Malik, Amir; Tomlinson, Bill (2011): Toward sustainable software engineering (NIER track). In: Proceedings of the 33rd International Conference on Software Engineering. ACM, S. 976–979.

Ardito, Luca; Morisio, Maurizio (2014): Green IT - Available data and guidelines for reducing energy consumption in IT systems. In: Sustainable Computing: Informatics and Systems 4 (1), S. 24–32.

RAL-UZ 161, 2012-07: Basic Criteria for Award of the Environmental Label Energy-Conscious Data Centers. Available online www.eco-institut.de

Berkhout, Frans; Hertin, Julia (2001): Impacts of Information and Communication Technologies on Environmental Sustainability: speculations and evidence. Report to the OECD. Hg. v. Organisation for Economic Co-operation and Development OECD. Brighton. Available online www.oecd.org, zuletzt geprüft am 02.03.2011.

Bouwers, Eric; van Deursen, Arie; Visser, Joost (2013): Evaluating usefulness of software metrics: an industrial experience report. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, S. 921–930.

Bozzelli, Paolo; Gu, Qing; Lago, Patricia (2013): A systematic literature review on green software metrics. Technical Report: VU University Amsterdam.

Calero, C.; Bertoa, M.F; Angeles Moraga, M. (2013a): A systematic literature review for software sustainability measures. In: Green and Sustainable Software (GREENS), 2013 2nd International Workshop on, S. 46–53.

Calero, Coral; Bertoa, Manuel F.; Moraga, Maria Ángeles (2013b): Sustainability and Quality: Icing on the Cake. In: RE4SuSy@RE. Citeseer.

Calero, Coral; Moraga, M.; Bertoa, Manuel F. (2013c): Towards a software product sustainability model. In: arXiv preprint arXiv:1309.1640.

Calero, Coral; Moraga, Maria Ángeles; Bertoa, Manuel F.; Duboc, Leticia (2015): Green Software and Software Quality. In: Coral Calero und Mario Piattini (Hg.): Green in Software Engineering: Springer, S. 231–260.

Capra, E.; Francalanci, C.; Slaughter, S. A. (2012): Measuring Application Software Energy Efficiency. In: IT Professional, S. 54–61.

Capra, Eugenio; Francalanci, Chiara; Slaughter, Sandra A. (2011): Is software green? Application development environments and energy efficiency in open source applications. In: Information and Software Technology 54, S. 60–71.

Dick, Markus; Naumann, Stefan (2010): Enhancing Software Engineering Processes towards Sustainable Software Product Design. In: Klaus Greve und Armin B. Cremers (Hg.): EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany. Aachen: Shaker, S. 706–715.

EPA ENERGY STAR (2014): ENERGY STAR Program Requirements Product Specification for Computers: Eligibility Criteria, Version 6.1. Environmental Protection Agency. Available online www.energystar.gov.

EPA Office of Air and Radiation, Climate Protection Partnerships Division (2015): National Awareness of ENERGY STAR for 2014. Analysis of CEE Household Survey. Hg. v. U.S. Environmental Protection Agency. Available online www.energystar.gov.

Erdmann, Lorenz; Hilty, Lorenz M.; Goodman, James; Arnfalk, Peter (2004): The Future Impact of ICTs on Environmental Sustainability. Technical Report EUR 21384 EN. Hg. v. Carlos Rodríguez Casal, Christine Van Wunnik, Luis Delgado Sancho, Jean Claude Burgelman und Paul Desruelle. European Commission; Joint Research Centre; IPTS - Institute for Prospective Technological Studies. Seville (Technical Report Series, EUR 21384 EN). Available online ftp.jrc.es, zuletzt geprüft am 26.07.2011.

Europäische Union (Hg.) (2011a): Beschluss der Kommission vom 6. Juni 2011 zur Festlegung der Umweltkriterien für die Vergabe des EU-Umweltzeichens für Notebooks. (Bekannt gegeben unter Aktenzeichen K(2011) 3736)Text von Bedeutung für den EWR. Available online eur-lex.europa.eu.

Europäische Union (2011b): Beschluss der Kommission vom 9. Juni 2011 zur Festlegung der Umweltkriterien für die Vergabe des EU-Umweltzeichens für Tischcomputer. (Bekannt gegeben unter Aktenzeichen K(2011) 3737)Text von Bedeutung für den EWR. Available online eur-lex.europa.eu.

Finkbeiner, Matthias; Schau, Erwin M.; Lehmann, Annekatrin; Traverso, Marzia (2010): Towards life cycle sustainability assessment. In: Sustainability 2 (10), S. 3309–3322. Available online www.mdpi.com.

Fujitsu Technology Solutions (Hg.) (2010): Green Label-Kategorien bei Fujitsu Technology Solutions. White Paper.

Fujitsu Technology Solutions (Hg.) (2012): Green Label Levels at Fujitsu Technology Solutions. White Paper. Available online globalsp.ts.fujitsu.com, zuletzt aktualisiert am 25.04.2012, zuletzt geprüft am 02.01.2013.

GeSI, Global e-Sustainability Initiative; The Climate Group (2008): SMART 2020: Enabling the low carbon economy in the information age.

Gröger, Jens; Köhn, Marina; Albers, Erik; Löhr, Patrik; Lohmann, Wolfgang; Naumann, Stefan (2015): Nachhaltige Software. Dokumentation des Fachgesprächs „Nachhaltige Software“ am 28.11.2014. Hg. v. Umweltbundesamt. Öko-Institut e.V. Dessau-Roßlau. Available online www.umweltbundesamt.de.

Gröger, Jens; Quack, Dietlinde; Grießhammer, Rainer; Gattermann, Marah (2013): TOP 100 - Umweltzeichen für klimarelevante Produkte: Freiburg. Available online www.ecodialog.de.

Held, Alexandra (2010): Entwicklung und Operationalisierung von Kriterien zur Bewertung der Nachhaltigkeit von Softwareprodukten. Abschlussarbeit zur Erlangung des akademischen Grades Master of Science eingereicht am Umwelt-Campus Birkenfeld. Masterarbeit. Fachhochschule Trier, Standort Umwelt-Campus

Birkenfeld, Hoppstädten-Weiersbach. ISS Institut für Softwaresysteme in Wirtschaft, Umwelt und Verwaltung.

Hilty, Lorenz; Lohmann, Wolfgang; Behrendt, Siegfried; Evers-Wölk, Michaela; Fichter, Klaus; Hintemann, Ralph (2015): Grüne Software. Schlussbericht zum Vorhaben: Ermittlung und Erschließung von Umweltschutzpotenzialen der Informations- und Kommunikationstechnik (Green IT). Studie im Auftrag des Umweltbundesamtes, Berlin, Förderkennzeichen 3710 95 302/3 (im Druck).

Horne, Ralph E. (2009): Limits to labels: The role of eco-labels in the assessment of product sustainability and routes to sustainable consumption. In: International Journal of Consumer Studies 33 (2), S. 175–182. Available online 19-659-fall-2011.wiki.uml.edu.

Kern, Eva; Dick, Markus; Naumann, Stefan; Guldner, Achim; Johann, Timo (2013): Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In: Lorenz M. Hilty, Bernard Aebischer, Göran Andersson und Wolfgang Lohmann (Hg.): ICT4S ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013. Zürich: ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, S. 87–94. Available online e-collection.library.ethz.ch.

Koçak, Sedef Akınlı; Calienes, Giovanna Gonzales; Alptekin, Gülfem Işıklar; Bener, Ayşe Başar (2013): Requirements Prioritization Framework for Developing Green and Sustainable Software using ANP-based Decision Making. In: EnviroInfo, S. 327–335.

Koçak, Sedef Akınlı; Alptekin, Gülfem Işıklar; Bener, Ayşe Başar (2014): Evaluation of Software Product Quality Attributes and Environmental Attributes using ANP Decision Framework. In: Proceedings of the Third International Workshop on Requirement Engineering for Sustainable Systems (pp. pp. 37-44). Karlskrona: Central Europe Workshop Proceedings. Available online ceur-ws.org

Lago, Patricia; Jansen, Toon; Jansen, Marten (2010): The service greenery-integrating sustainability in service oriented software. In: International Workshop on Software Research and Climate Change (WSRCC), co-located with ICSE, Bd. 2.

Lago, Patricia; Koçak, Sedef Akinli; Crnkovic, Ivica; Penzenstadler, Birgit (2015): Framing sustainability as a property of software quality. In: Communications of the ACM 58 (10), S. 70–78.

Lami, Giuseppe; Fabbrini, Fabrizio; Fusani Mario (2012): Software Sustainability from a Process-Centric Perspective. In: D. Winkler, R.V O'Connor und R. Messnarz (Hg.): EuroSPI 2012, CCIS 301: Springer, S. 97–108.

Mazijn, B.; Doom, R.; Peeters, H.; Vanhoutte, G.; Spillemaeckers, S.; Taverniers, L. et al. (2004): Ecological, Social and Economic Aspects of Integrated Product Policy. Integrated Product Assessment and the Development of the Label `Sustainable Development´ for Products. CP/20. SPSD II - Part I - Sustainable production and consumption patterns. Available online www.bernardmazijn.be.

Naumann, Stefan; Dick, Markus; Kern, Eva; Johann, Timo (2011): The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. In: SUSCOM 1 (4), S. 294–304. DOI: 10.1016/j.suscom.2011.06.004.

Penzenstadler, Birgit; Mahaux, Martin; Salinesi, Camille (2013): RE4SuSy: Requirements Engineering for Sustainable Systems. In: Journal of Systems and Software.

Prakash, Siddharth; Manhart, Andreas; Stratmann, Britta; Reintjes, Norbert (2008): Environmental product indicators and benchmarks in the context of environmental labels and declarations. Öko-Institut e.V.; Ökopol GmbH.

Schipper, Irene (2015): TCO Certified Smartphones versusFairphone. A comparison of sustainability criteria. Hg. v. GoodElectronics Network Südwind. Stichting Onderzoek Multinationale Ondernemingen (SOMO), Centre for Research on Multinational Cooperations, Netherlands. Amsterdam.

Schmidt, Benno (2014): Strategien für eine integrativ-nachhaltige Software-Entwicklung. Hochschule Bochum, Fachbereich Geodäsie. Bochum (14-02).

Schmidt, Benno; Wytzisk, Andreas; Plödereder, In E.; Grunske, L.; Schneider, E.; Ull, D.; others (2014): Software Engineering und Integrative Nachhaltigkeit. In: Erhard Plödereder, Lars Grunske, Eric Schneider und Dominic Ull (Hg.): INFORMATIK 2014. Big Data – Komplexität meistern. – Proceedings. 2. Workshop "Umweltinformatik zwischen Nachhaltigkeit und Wandel (UINW) / Environmental Informatics between Sustainability and Change"), 26.09.2014, im Rahmen der INFORMATIK 2014, 22.-26.09.2014 in Stuttgart. P-232: Lecture Notes in Informatics (LNI), S. 1935–1945.

Scholl, Gerd; Simshäuser, Ulla (2002): Machbarkeitsuntersuchung für Umweltzeichen-Analyse der Möglichkeiten zur Akzeptanzerhöhung des Umweltzeichens" Blauer Engel" für Haushaltsgroßgeräte (" Weiße Ware") bei potenziellen Zeichennehmern: Umweltbundesamt. Available online www.umweltbundesamt.de.

Stieß, Immanuel; Birzle-Harder, Barbara (2013): Der Blaue Engel – ein Klassiker mit Potenzial: eine empirische Studie zu Verbraucherakzeptanz und Marktdurchdringung des Umweltzeichens.

Sundblad, Yngve; Lind, Torbjörn; Rudling, Jan (2002): IT product requirements and certification from the users' perspective. In: Proceedings of WWDU 2002 Conference, S. 280–282. Available online http://cid.nada.kth.se/pdf/CID-176.pdf

International Standard ISO/IEC 25010:2011, 01.03.2011: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.

Taina, Juha (2011): Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In: CEPIS UPGRADE XII (4), S. 22–27. Available online www.cepis.org, zuletzt geprüft am 09.01.2012.

TCO Development (Hg.) (2012): TCO Certified Notebooks 4.0. Available online tcodevelopment.com.

Teufel, J.; Rubik, F.; Scholl, G.; Stratmann, B.; Graulich, K.; Manhart, A. (2009): Untersuchung zur möglichen Ausgestaltung und Marktimplementierung eines Nachhaltigkeitslabels zur Verbraucherinformation. In: Project report of the Öko-Institut e. V. in cooperation with the Institut für ökologische Wirtschaftsforschung (IÖW) GmbH. Freiburg: Öko-Institut e. V. Available online download.ble.de.

Umweltbundesamt (Hg.) (2013): Ökodesign-Richtlinie <Computer und Computerserver>. Available online www.umweltbundesamt.de.

Vergabegrundlage für Umweltzeichen RAL-UZ 100, 2014-06: Vergabegrundlage für Car-Sharing.

Vergabegrundlage für Umweltzeichen RAL-UZ 78a, 2014-11: Vergabegrundlage für Umweltzeichen - Computer.

Walch, Isabelle (2015): Standard ECMA-370. TED - The ECO Declaration. Hg. v. ecma INTERNATIONAL.

Waller, Lars (2015): TCO Certified Desktops 5.0 - Critera Document and Certification Process. TCO Development. Available online tcodevelopment.com, zuletzt aktualisiert am 11.11.2015.

Warschun, Mirko; Rühle, Jens (2008): Zwischen ÖkoLabels, grüner Logistik und fairem Handel Lebensmitteleinzelhandel auf der Suche nach Wegen zur Nachhaltigkeit.

Environmental issues of software & its labelling: an end user perspective

– Paper 3 –

Eva Kern[a,b*]

[a] Leuphana University Lueneburg, Scharnhorststrasse 1, 21335 Lueneburg, Germany

[b] Institute for Software Systems in Business, Environment, and Administration, Trier University of Applied Sciences, Environmental Campus Birkenfeld, P.O. Box 1380, 55761 Birkenfeld, Germany

Abstract. The energy consumption caused by Information and Communication Technologies (ICT) is increasing. Activities to counteract here are comprised by the term "Green IT". While there was a focus on the hardware side in research activities in the context of Green IT for a long time, research is increasingly addressing software issues as well. However, it seems as if the topic of environmental issues of software did not reach software users and developers, so far. The following paper presents a user survey addressing the awareness of and the interest in environmental issues of software. It turned out that the subject is new to the participants of the survey even if they are generally interested in environmental topics. Nevertheless, most of them can imagine taking environmental issues into account while searching for new software products. The aim is to include the findings in the development of an eco-label for software. Hence, the article will present corresponding recommendations that are based on the survey results.

Keywords. green software, eco-label, awareness, user survey, label development

1 Introduction

Possibilities to save energy and natural resources while developing, using, and maintaining ICT equipment have been discussed comprehensively in academia as well as in practice based around the term "Green IT". These discussions mainly focus on hardware aspects. However, "software characteristics determine which hardware capacities are made available and how much electric energy is used by end-user devices, networks, and data centers." [Hilty et al. 2015] In corresponding research communities, the connection between software and natural resources found its way into the

discussions. This trend is shown by a rising number of academic events as well as publications in the addressed research field (see Figure 1).

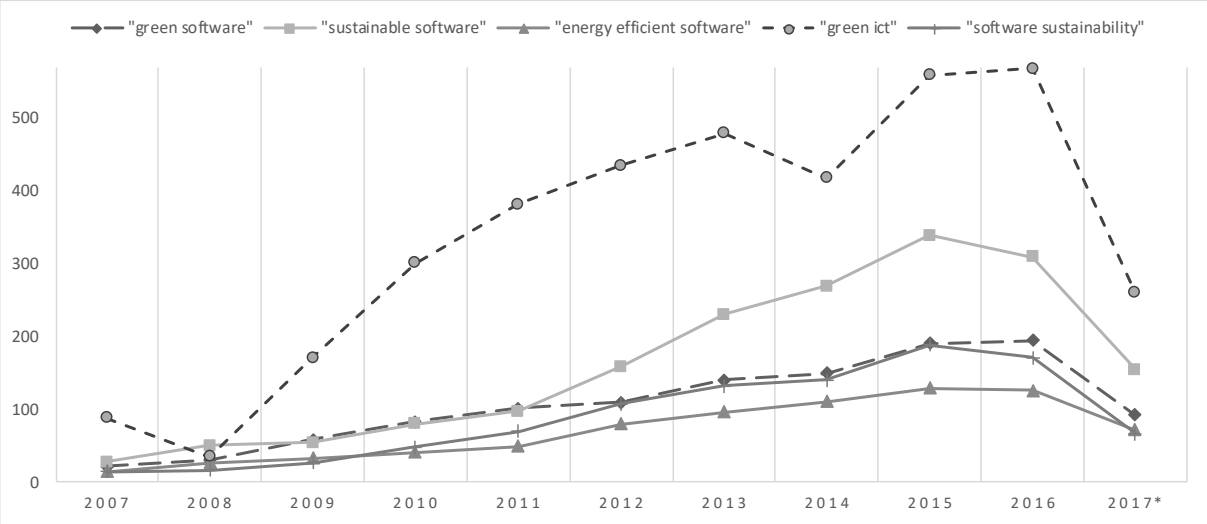

Figure 1 *Development of the number of papers published in the context of "green and sustainable software" from 2007 to the first half of the year 2017 based on Google Scholar results.*
The search on Google Scholar was conducted on July 6th 2017 and used the key words: "green software", "sustainable software", "energy efficient software", "green ict", and "software sustainability" (always including the quotation marks)

Certainly, it seems as if environmental issues of software did not reach a social awareness, so far [Torre et al. 2017]. Software's impacts on the environment or rather on a sustainable development are rarely investigated and even less known in non-academic contexts. To counteract here, the idea is to develop a product certification and thus providing information about environmental issues of software products.

An eco-label for software products can inform especially end users1 and purchasers of software about the environmental friendliness of the products they are using or searching for. Here, the focus is set on software users. Thus, the gap between research activities and society's awareness regarding environmental issues of software can be closed. In that way, the label can support transparency and might lead to a more environmental friendly user behavior about ICT products, especially software.

The main condition to develop such an eco-label is a set of awarding criteria. Besides the relevance of these criteria from an environmental point of view, the social interest in aspects to be awarded is important, as well [Grießhammer et al. 2007]. The latter point might influence the future acceptance of the resulting eco-label. Thus, I conducted a user survey regarding green software and its possible certification. The results of the survey will be presented in this article.

[1] Within this paper, the term "users" refers to those who are using software products and thus are the target group of the proposed eco-label. The idea is that this group takes the label into account while they are searching for new software products. Users stands in contrast to developers, administrators, IT managers, purchasers, distributors, and vendors. However, the members of the groups probably overlap.

The survey is not only to evaluate possible criteria for an eco-label for software products, but also to spread information about the connection between (using) software and the environment. By questioning the future target group of a label already during its development, the interests of the target group can influence the label development. Thus, it might be possible to find out those aspects that are especially relevant from a software user's point of view. Including them may lead to a higher acceptance of the resulting label.

The objective of the article is to provide an insight into awareness, interests, and ideas regarding an eco-label for software products among users. Thus, the presented results of the survey and derived recommendations help move the development of a certification for green software products one step forward. Thus, I approach the following research questions (RQ):

- RQ1: Which criteria for green software products arouse the interest of software users?
- RQ2: What are aspects influencing the user acceptance of an eco-label for software products?

The article is structured as follows: Section 2 presents information on the relevant key aspects of the green software research field. After providing this information regarding green software in general, eco-labelling and awareness of green computing, the following sections present the structure of the user survey (Section 3) and the findings (Section 4). Based on the results of the survey, initial recommendations for the development of an eco-label for software are outlined in Subsection 4.5. Then, I go into limitations of the survey and the idea of labelling green software products (Section 5). Finally, in Section 6, the findings are summarized and suggestions for future research activities in this context are presented.

2 Background

The related work for the article at hand considers three areas of "Software and Sustainability": In order to create a common basis of understanding, Section 2.1 starts with an introduction to the different existing definitions for the term "green (and sustainable) software". Since the article deals with eco-labelling of software products as one opportunity to address the awareness of green computing issues, the following review will additionally present information regarding these two aspects: Section 2.2 presents aspects of eco-labelling and Section 2.3 describes the awareness of green computing.

2.1 Green software: definition & characteristics

Depending on the selected focus of ICT sustainability, there are different definitions and terms used to describe the issues of software and their impacts on a sustainable development. Durdik et al. [2012] talk about "long-living software systems" that are "sustainable if they can be cost-effectively maintained and evolved over their complete life-cycle". Lami and Buglione [2012] go into the same direction when defining the sustainability of software products as "the capability of the software

product to meet current needs of required functionalities without compromising the ability to meet future needs". Here, the focus is set on the technical aspects of the system and the economic view of sustainability. Agarwal et al. [2012] also mention the longer life-aspect, but go one step further: "The term sustainable applies both the longer life and greener aspects of software." They point out the importance of the way a software product is used and consequent impacts towards sustainability. Amsel et al. [2011] place special emphasis on the users themselves when bringing in their needs: "Sustainable software engineering aims to create reliable, long-lasting software that meets the needs of users while reducing environmental impacts." Here, the environmental sustainability is explicitly referred. Additionally, pointing out the "needs of users" could be understood as bringing in social aspects. According to Johann and Maalej [2013], social sustainability of software must be taken stronger into account in the discussion about sustainable informatics. I agree that especially social aspects play an important part, e.g. with regards to usability and working conditions during the development phase. Thus, the work at hand is based on one of the first definitions in the field of sustainable software: "Green and Sustainable Software is software, whose direct and indirect negative impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software are minimal and/or which has a positive effect on sustainable development." [Dick et al. 2010; Naumann et al. 2011] However, the focus is set on environmental aspects.

Without exactly explaining the term, Bozzelli et al. [2013] provide metrics regarding the so called "greenness" of software. Having a software quality lifecycle in mind, Calero et al. [2015] coined the term "greenability" as the environmental sustainability of software. From their perspective "greenability is a 'how', because it is a way to improve a software product and must therefore be part of its quality." A wide-ranging examination of further terms and their corresponding definitions regarding green and sustainable software, ICT as well as systems can be found in Calero and Piattini [2015]. Whatever they are called, behind the terms and definitions lay specific characteristics of software products that are discussed in the respective research papers.

Researching the different papers regarding sustainable software quality criteria and characteristics, our research project called "Sustainable Software Design"[2] introduces more than 70 indicators[3]. Although the literature search conducted within the project was not based on the rules of a Systematic Literature Review (SLR) [Kitchenham and Charters 2007] or similar methods, we reviewed quite a representative number of studies that allows us to make considerations.

---

[2] http://green-software-engineering.de/en/project/ufoplan-ssd-2015.html The author of the contribution at hand was involved in the research project, the survey presented was not part of the project but of the doctoral research activities of the author.

[3] Further details and the resulting criteria catalog will be published online: http://green-software-engineering.de/en/kriterienkatalog

The findings include “Green by IT”-aspects as well as “Green (in) IT”. Similarly, software sustainability has two sides: “(1) the software code being sustainable, agnostic of purpose, or (2) the software purpose being to support sustainability goals, i.e. improving the sustainability of humankind on our planet.” [Penzenstadler et al. 2014a] For example, Taina [2011] proposes the criterion “reduction” that “defines how software supports its system in waste reduction”. This aspect counts among aspects how to support a sustainable development by using ICT or rather software (“green / sustainable by software”). While evaluating the energy efficiency [Naumann et al. 2011; Capra et al. 2012], hardware intensity [Hilty et al. 2015], obsolescence [Albertao 2004; Hilty et al. 2015], carbon footprint [Taina 2011; Kern et al. 2014], or perdurability [Calero et al. 2013b] of a software product, the sustainability of software products themselves is considered (“green / sustainable (in) software”). Additionally, the criteria can be divided into common criteria (i.e. standardized quality issues of software based on ISO/IEC 25000), direct criteria (i.e. referring to first-order effects), and indirect criteria (i.e. effects that are indirectly caused by energy and effects that have an indirect impact over the long run). Further categories for classifying criteria for sustainable software are: the software life cycle phases [Naumann et al. 2011] and sustainability aspects. Here, Penzenstadler and Femmer [2013] define five dimensions of sustainability: individual sustainability, social sustainability, economic sustainability, environmental sustainability, and technical sustainability. Some exemplary criteria for sustainable software are:

- Individual sustainability: accessibility [Albertao et al. 2010], usability [Albertao et al. 2010; Calero et al. 2013a], Safety and Security [Johann and Maalej 2015]
- Social sustainability: usability [Albertao et al. 2010; Calero et al. 2013a], organization’s sustainability [Kern et al. 2013]
- Economic sustainability: perdurability [Calero et al. 2013b], effectiveness [Calero et al. 2013a]
- Environmental sustainability: Carbon Footprint [Naumann et al. 2011; Taina 2011], reduction [Taina 2011]
- Technical sustainability: Testability [Calero et al. 2013b], Modifiability [Albertao et al. 2010; Calero et al. 2013b]

However, this mapping can be discussed from varied perspectives. Overall, the different classifications overlap.

Take, for example, the case of “energy consumption” (Figure 2): the criterion evaluates energy consumption that is consumed by the hardware caused by implementing software [Capra et al. 2011; Taina 2011; Hilty et al. 2015]. It goes into the “Green (in) Software”-category and is a direct criterion. Since the definition is talking about “implementing software”, having the life cycle of software products in mind, it belongs to the usage phase. Nevertheless, it could be also referred to the

development phase when measuring the energy that is consumed to develop a software product. The energy consumption of a software product is a criterion of environmental sustainability.

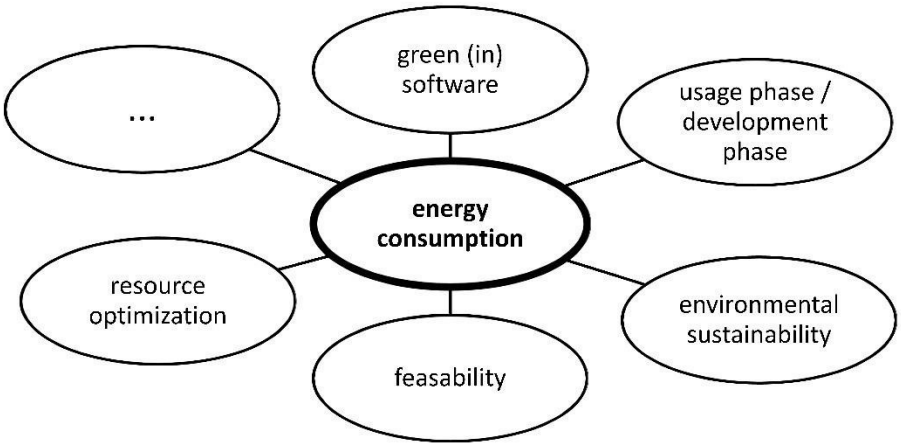

Figure 2 *Mapping of the criterion "energy consumption" to categories in the field of green and sustainable software.*

Summarizing, it depends on the perspective and context how to develop and structure criteria for sustainable software products. The different criteria found might use varying terms meaning the same issue and might be interpreted in divergent ways. However, the aim here is to give an insight into the current state-of-research of searching for criteria for green and sustainable products. In order to include them in software engineering, sustainability and its aspects have to be considered as a nonfunctional requirement [Penzenstadler et al. 2014b]. While Mahaux et al. [Mahaux et al. 2011] go into detail in asking how to define sustainability requirements for software products and activities supported by software, I concentrate on the end user perspectives on possible sustainability characteristics.

2.2 Eco-labelling: acceptance and awareness

The idea of creating eco-labels to inform about environmental issues of products is not new. This strategy might be especially interesting in contexts that are not directly brought into connection with environmental or rather sustainable aspects. Software is one of these cases as it is an immaterial good. Generally, software seems to be sustainable per se. However, it requires resources by being responsible for producing and running hardware products. [Capra et al. 2011; Taina 2011; Hilty et al. 2015] This connection between the life cycle of a software product and its effects on sustainable development, specifically on the environment, has to be pointed out in a stronger way – for example by creating a certification for environmental issues of software [Kern et al. 2015].

The idea of creating an eco-label for software products is to inform about environmental impacts of software. The information might lead to more eco-friendly behavior of software users while purchasing and using the products. However, even if "respondent's trust in eco-label and eco-brand has a positive effect on consumer's actual purchase behavior" [Rahbar and Abdul Wahid 2011], it is

not clear if label information influences the intention to purchase environmental friendly products [D'Souza et al. 2006]. The analysis of the environmental effectiveness of eco-labels is highly complex since it is (i) influenced by many other aspects and (ii) eco-labels themselves have multiple characteristics [Rotherham 2005]. Overall, one can find references pointing out the positive effects of eco-labels on consumer behavior [Rashid 2009] as well as studies showing that there are just minimal to no effects or that the effects are unclear [Pedersen and Neergaard 2006; Horne 2009; Rahbar and Abdul Wahid 2011]. In this context, Horne [2009] summarizes: "It is clear that eco-labels can affect consumer choice although it is less clear whether this leads to reduced environmental impacts". The Agenda 21 addresses the issue of labelling products and "identified eco-labelling as a way to encourage consumers to adopt more sustainable consumption patterns through the purchase of products that are more resource and energy efficient." [Horne 2009]

Generally, a label can be understood as "a tool that communicates expectations and requirements to whoever is interested" [Rotherham 2005]. Thus, they provide the possibility to create transparency and a process of awareness-raising in different contexts. Rotherham [2005] points out that "ecolabelling might be considered not as a tool in itself but as a communication tool working in concert with and empowering other public or private policy initiatives."

Overall, the debate and actions on climate change have "driven a renewed interest in eco-labels as a means to drive a widespread transition towards more sustainable lifestyles." [Horne 2009] However, this also leads to more and more eco-labels, including self-declared labels [Organisation for Economic Co-Operation and Development (OECD) 2008]. A result of this development is an overflow of the market and a mental overload of the consumers.

2.3 Awareness of green computing

Knowledge about green computing covers, according to the description of "environmental knowledge" by D'Souza et al. [2006], understanding the general environmental impacts of ICT and the eco-friendliness of the production process of the product. Environmental impacts in general or, to be more specific, energy issues of software products, should not only be an issue for those using software and caring about their ecological footprint [Pang et al. 2016; Jagroep et al. 2017]. Additionally, different studies list, among others, developers, legislation, community representatives, managers, activists, admins, industry, and government as stakeholders for Green IT [Herzog et al. 2015] or rather sustainable software engineering [Penzenstadler et al. 2013; Hindle 2016].

While the topic is more and more an issue in research communities, it is still a niche topic in the industry. Jagroep et al. [2017] point out that "industry is not yet able to adopt solutions provided by research." Penzenstadler et al. [2014a] state that about 80 % of reported evidence comes from academia. Here, in the software engineering community, the "topic of SE4S [software engineering for sustainability] has received wide-spread attention […] over the past few years" [Penzenstadler et al.

2014a]. This is confirmed by the results of different literature reviews [Penzenstadler 2012; Bozzelli et al. 2013; Procaccianti et al. 2014].

As stated, the awareness is an issue to reach a change in costumers purchasing behavior [Rashid 2009]. Thus, the question arises if especially practitioners are aware and/or knowledgeable about energy consumption and further green software issues. Hindle [2016] states that programmers are, in contrast to researchers, "not really" aware of those issues. This statement is confirmed by Grosskop and Visser [2013], Pang et al. [2016], and Torre et al. [2017]. Pang et al. [2016] affiliate the non-awareness to priorities in developing software: functionality matters most.

Additionally, the kind of software considered influences the awareness for energy efficiency and similar aspects: "energy usage requirements are more common for mobile and data center projects than traditional projects." [Manotas et al. 2016] Many programmers see energy consumption of mobile platforms as a decision factor (65 % of the persons surveyed by Pang et al. [2016]), but they do not see this requirement in case of traditional software (12 %, ib.).

Another reason for (mainly) not-addressing green issues while developing software is a lack of knowledge. The practitioners might care about energy but could be much more effective if they were more trained in creating efficient applications. [Manotas et al. 2016; Torre et al. 2017] According to Ferreira [2011], the general knowledge or even an idea about the role of software in the context of energy consumption is given: "98% of [the] respondents [of the Green Software Awareness Survey] believe that two pieces of software can have different energy consumption profiles for the same functionality." Overall, the results of the different studies show the need for training in green computing, green software engineering, and energy issues. According to Jagroep et al. [2017], "a willingness to address the SEC [software energy consumption] when sustainability goals are formulated and clear benefits can be identified".

Studies addressing student's awareness of green computing support this demand for education [Selyamani and Ahmad 2015]. The surveys show that there is some knowledge about green computing, but the transfer to practices as an everyday routine is missing [Dookhitram et al. 2012; Selyamani and Ahmad 2015]. Thus, based on this and the fact that these surveys concentrate on hardware aspects of green computing, it is assumed that if there is quite little knowledge on hardware issues, there will be even less knowledge on green software and its engineering. The results of the study addressing the presence of these issues in higher education curricula conducted by Torre et al. [2017] confirms this. On the other hand, this means that the fully-educated students starting to work in companies do not have the possibility to bring Green IT knowledge into the organizations, since they do not have it. According to Kogelman [2011], they will also not be trained in their companies to counteract this lack of knowledge. In case of IT decision makers, the reasons for that are: price, internal and/or political disagreement, efficiency, not offsetting costs, and lack of brands to convince them of the return on investment and promoting the importance of green issues [Dookhitram et al.

2012]. Some approaches on how to integrate Green IT issues into higher education curricula are presented e.g. by Lago [2014], Issa et al. [2014], and Torre et al. [2017].

However, researching environmental impacts and sustainability of ICT barely address end users [Selyamani and Ahmad 2015]. Thus, it is not surprising that "software users and clients [are] unaware of software energy consumption" [Pang et al. 2016]. As a consequence, they neither request energy efficiency nor complain about resource consumption of software products. [Ferreira 2011; Pang et al. 2016]

Looking at the studies, I draw the conclusion that the activities to spread the idea, knowledge and recommendations of sustainable software and its engineering, should be expanded. Next to academia, industry, and education, users of software should be integrated, as well [Lami et al. 2013]. Thus, the survey, presented in the following section, focusses on users' knowledge, awareness, and interests in Green IT and especially green software.

3 User survey: objectives and structure

In order to find out (i) if users are familiar with green software, Green IT labels, and corresponding aspects and (ii) how they could be informed about these issues, I conducted a user survey. The idea to spread information about environmental issues of software and thus raise the transparency in this field is to create an eco-label for software products. To do so, especially the awarding criteria are important [Kern et al. 2015].

3.1 Objectives of the survey "Green Software and its certification"

The main objective of the user survey was to find out the users' interest in environmental issues of software. The motivation is to draft recommendations on how to develop a label – including the view of those using the products in the process. From this perspective, the development of a certification of green software seems to be reasonable, if the interest of the target groups for the label (those who develop, buy, use, maintain, … software) is given. In my opinion, the group of end users, who are searching for software products and using them, is the most important one. In comparison to developers, purchasers, distributors, administrators, vendors, and IT managers, they are the biggest group and they are the ones who mainly decide on their own what product to use. It is assumed that if they are asked to bring in their interests into the label development process, the acceptance of the label might be higher. Here, acceptance means that the resulting eco-label for software products is considered while searching for products.

Since “demand and attitudes for green products is likely to be uneven across different market segments and cultures” [Rahbar and Abdul Wahid 2011], the main target group of the survey is:

- Software users,
- speaking German, and
- differ in age, gender, IT skills, knowledge regarding environmental topics, field of work / studies, etc.

Thus, the target was to spread the questionnaire as much as possible in Germany using mainly mailing lists, personal contacts (“snowballing method”), and social media.

3.2 Structure of the survey and used data

The questionnaire comprises three parts and includes mainly closed questions. Part 1 of the questionnaire addresses the demographic characteristics of the participants (see Section 4.1). The second and third part directly tackle the research questions pointed out in the introduction: evaluation of criteria for green software productions (RQ1, see Section 4.3) and aspects influencing the acceptance of an eco-label for software products (RQ2, see Section 4.4). In this article, the focus is laid on the second part of the survey.

While developing the survey, I used so called construct maps [Wilson 2004] for the definition of the responses of the polar questions. One example is given in Table 1. The options of response belong to the attitude towards environmental issues. It refers to the interest of the participant regarding environmental topics and environmental issues in their own life.

Table 1 *Construct map for the question "To which of the following statements do you agree mostly? (choose one answer)"*

Response	**Items**
Very positive attitude towards environmental issues	I am willing to pay more for organic products.
Positive attitude towards environmental issues	I catch up on environment topics regularly.
Rather positive attitude towards environmental issues	I go for green energy, but do not purchase it personally.
Rather negative attitude towards environmental issues	I think environment protection is just a fad.
Negative attitude towards environmental issues	I think organic products are a fraud.
Very negative attitude towards environmental issues	I think there are problems that are more important than environment protection.

In addition to the questions themselves, the survey included a short glossary of terms that might be new to the survey participants. To give some examples:

- Green software product[4]: "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." [Dick and Naumann 2010; Naumann et al. 2011]
- In simplified terms: software products with the lowest possible environmental impacts.
- Here, the terms "green software" and "green software product" are synonymous.
- Hardware resource[5]: A part of the computer hardware, e.g. processor, RAM, background memory, networking device.
- Natural resource[6]: A part of nature, e.g. renewable and not-renewable raw material, physical area, water / soil / air, flowing resources (e.g. geothermal energy, wind energy, solar energy), and biodiversity.

Providing this information should bring the knowledge of the participants to the same level regarding these terms. Overall, the glossary comprised seven terms (green software product, hardware, hardware resource, natural resource, product certification, resource, software). Nevertheless, the general knowledge of those participating in the survey differed. I am aware of the fact and welcome these differences and variety of participants. It is considered within the third part, e.g. by asking for the field of studies or work.

In the third part of the survey, the participants are asked to evaluate aspects of environmental issues of software, to find possible criteria for an eco-label for software products the users are interested in. The presented aspects are results from the "Sustainable Software Design" project. While the focus of the project is set on developing a methodology to better assess the environmental impacts of software, the focus of the survey is set on social interests in this context.

4 Sample and findings

In the following, the results of the survey conducted during 16th August and 5th October 2016 are presented. The survey addressed environmental issues of software and their certification. The overall aim was to bring a topic that is mainly based in research discussions to society. This requires the knowledge about society's interests. Thus, the data was collected via an anonymous online questionnaire, addressing primarily German software users by asking the questions in German. When

[4] http://green-software-engineering.de/en/referenzmodell.html

[5] https://www.umweltbundesamt.de/publikationen/gruene-software

[6] https://www.umweltbundesamt.de/publikationen/glossar-ressourcenschutz

closing the questionnaire, the sample comprised n = 854 participants. This dataset has been revised by excluding those that were not completed. A total of 712 questionnaires were completed and used for data analysis. The data was analyzed using both, descriptive methods and correlations between variables.

4.1 Participants of the survey

In order to show who participated in the survey, the participants are described by using their demographic data provided in the third part of the questionnaire: 55.8 % of the respondents are female (40.7 % male, 3.5 % not specified). About one third of the participants were between 20 and 39 years old (20 to 29 years: 39.3 %, 30 to 39 years: 29.4 %). About half of the answers were given by academics (48.3 % hold a master degree, diploma of a university or PhD). 52.5 % of the responders assign themselves to the field "natural sciences, geography, and informatics".

Most of the participants saw themselves primarily as software users (78.2 %, developer: 11.5 %, IT manager: 3.5 %, administrator: 2.9 %, purchaser 1.8 %, distributor: 0.3 %, vendor: 0.1 %, others: 1.5 %). This fits with the proposed target group. Regarding their personal IT skills, they evaluated themselves mainly as experienced persons (40.4 %) and intermediates (36.5 %), see Figure 3.

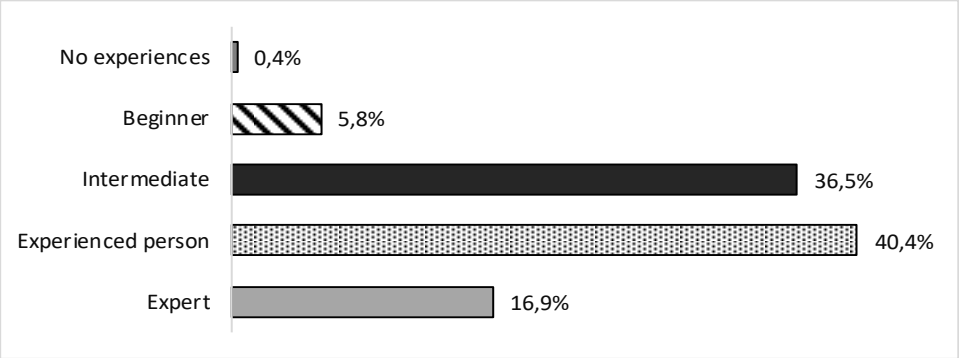

Figure 3 *Rating on the IT knowledge of the survey participants*

4.2 Awareness of existing eco-labels

When asked "Which one of the following test seals [TCO Certified, Blue Angel, Energy Star] do you know? (multiple choice)", nearly all of the responders claimed to know the Blue Angel (94.0 %, see Figure 4). However, faced with questions regarding details of the label, certified products, and its awarding criteria, the participants were quite uninformed. They answered many questions with "I don't know". Especially, the statements about the certifications seemed to be not known by the survey participants.

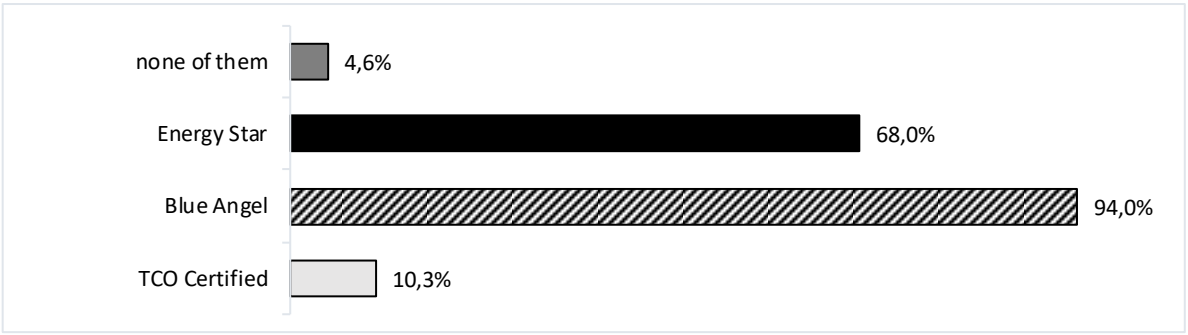

Figure 4 *Which one of the following test seals do you know? (Multiple choice)*

These findings comply with similar studies [Ramachandiran 2012; Selyamani and Ahmad 2015]. However, in our case, the products were generally known in many cases, especially when it comes to "prominent examples", like the ENERGY STAR for desktop PCs (83.9 % of those who know the label answer the question correctly) or TCO Certified for monitors (97.2 %) in contrast to Blue Angel for data centers (15.1 %). The lack of knowledge mainly exists in awarding criteria.

The uncertainty regarding labelling details could be interpreted as invitation to provide more information in this context. Similarly, it is a motivation for the development of an eco-label for software products to lay emphasis on information aspects. This is supported by the rating of information issues as "rather important" in the context of labelling green software products (Table 2).

Table 2 *Results of the evaluation of statements belonging to the information aspect*

Statement	**I agree**	**I rather agree**	**I rather do not agree**	**I do not agree**	**I do not know**
I consider criteria of product certifications.	12,5 %	**41,0 %**	35,7 %	9,6 %	1,3 %
I want to know the exact meaning of a certification before I take it into account.	35,8 %	**46,2 %**	13,8 %	1,1 %	3,1 %
It is important for me to be able to catch up on the certification in detail.	**56,7 %**	30,6 %	10,0 %	1,3 %	1,4 %

Please note: The answers have been interpreted against the assumption „the higher the agreement towards the statements, the more important the aspect is for the responder.

4.3 Evaluation of environmental issues of software

While there seems to be an interest in environmental topics and general knowledge regarding eco-labels in case of the surveyed persons (Section 4.1), the next step is to find out if these interests are also given regarding labelling green software products. The following paragraphs describe the results of the second part of the questionnaire.

4.3.1 Awareness and willingness

On the one hand, the results of the survey show that eco-labels are generally known by the responders (cf. Section 4.2). On the other hand, more than half of the survey participants agreed to the statement "I never took heed of the fact if there is a certification for environmental issues of software products."

(56.3 % “I agree”, 18.7 % “I rather agree”, 12.6 % “I rather do not agree”, 10.4 % “I do not agree”). These results comply with the results of the Green Software Awareness Study by Ferreira [2011]: “91% of respondents have never set or encountered any requirements regarding energy efficiency of software” and “Vast majority (85%) of respondents do not consider energy consumption when buying or developing software applications”. However, 35.8 % of the responders of the presented study disagreed and said that an environment certification for software is important while searching for new software products (30.6 % do rather not agree).

However, most of the participants said that they would not buy an unknown software product even if it is the only one that is certified as “being green” (63.3 %). This result is supported by the answers regarding the question “Can you imagine taking ‘grading of the environmental impact of the product’ as a criterion while searching for a software product? (choose one answer)”: 23.2 % of the participants of the survey say “Yes, I can imagine doing so.” 51.1 % can imagine doing so if the functionality of the product is the same between different vendors (Figure 5). Thus, I agree with the hypothesis by Ferreira [2011] that awareness for the topic is necessary: “Given the clear awareness, respondents are taking energy consumption into consideration when buying or developing software applications.” Similar to my results, mainly based on answers by software users, Pang et al. [2016] point out that also in case of programmers “the lack of attention to software energy consumption is an issue of priorities”.

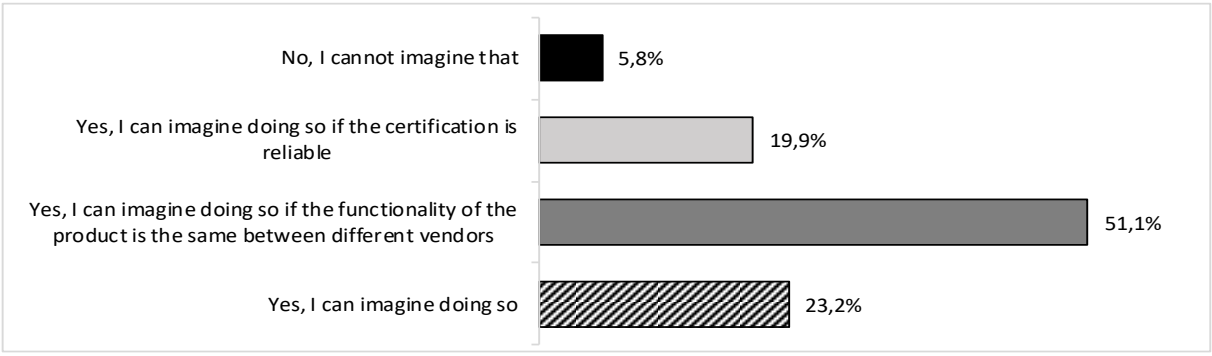

Figure 5 *Can you imagine taking ‘grading of the environmental impact of the product’ as a criterion while searching for a software product?*

Thus, linking the answers to this question with the roles of using software, the following results emerge: The biggest groups of my participants were users (78.2 % of the survey participants) and developers (11.5 %). Here, the distribution of the answers is quite similar. However, the tendency to disagree is slightly higher in case of the last-mentioned group (Figure 6).

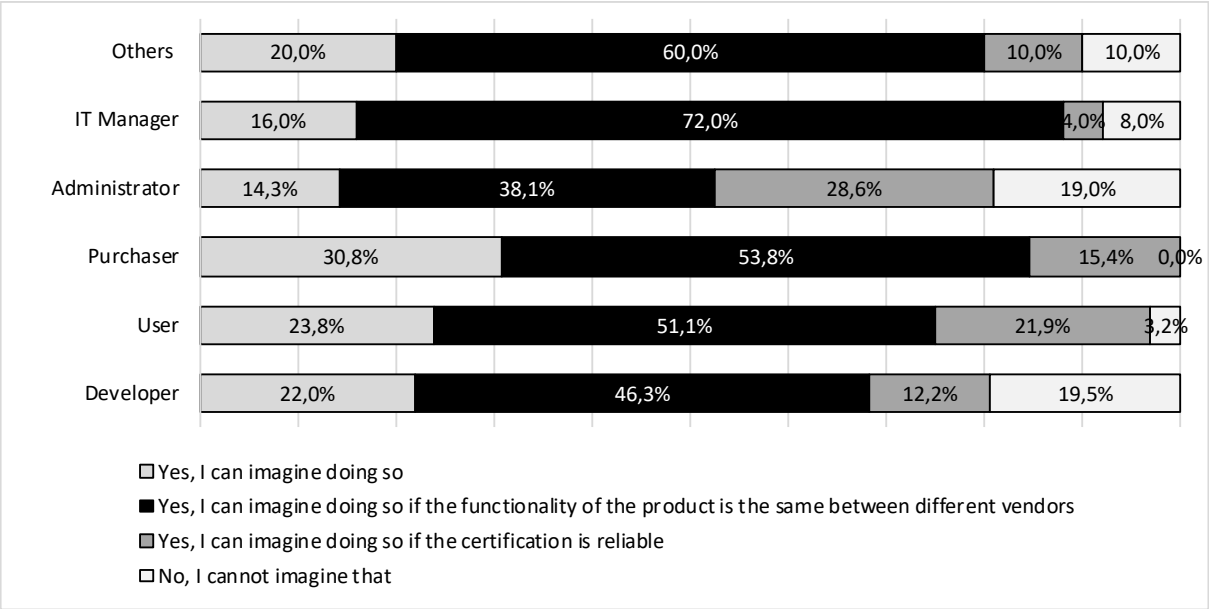

Figure 6 *Results of the question regarding "taking 'grading of the environmental impact of the product' as a criterion while searching for a software product" (see Figure 5) differentiated by role in using software. The diagram does not include "vendors" and "distributors" since there were too little mentions in this category.*

Including the IT skills of the participants in the interpretation of the survey results, the response bias is quite similar. Here, experienced persons (40.4 %) and intermediates (36.5 %) were the biggest groups. The relative frequencies of the answers of these two groups regarding the considered question were nearly the same (Figure 7).

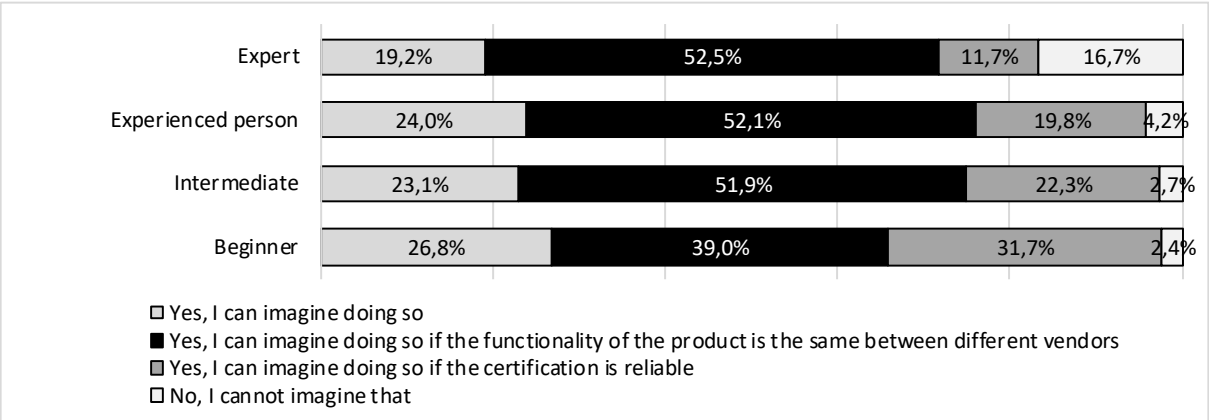

Figure 7 *Results of the question regarding "taking 'grading of the environmental impact of the product' as a criterion while searching for a software product" (see Figure 5) differentiated by IT knowledge of the participants (see Figure 3). The diagram does not include "persons with no experience" since there were too little mentions (3) in this category.*

Overall, while interpreting the survey results, one should keep in mind that the participants do have a high environmental awareness: 61.4 % of the responders were willing to pay more for organic products; 23.7 % caught up on environmental topics regularly. Additionally, 95.4 % of those completing the questionnaire, knew at least one of the Green IT labels presented (cf. Section 4.2).

Summarizing, the survey participants did not think about an eco-label for software products so far although they are generally interested in environmental issues and can imagine this kind of

certification. Thus, it seems as if there is no or just little awareness for environmental aspects of software so far.

4.3.2 Evaluation of possible criteria for green software products

One idea to create, or rather increase, the awareness for environmental impacts caused by software products, especially during their usage phase, is to develop an eco-label for software products. Within the project "Sustainable Software Design" we develop a method to assess and evaluate these impacts. The objective is to derive green software criteria, upon which an eco-label could be based. In this context, especially those aspects are relevant that have high impact on the environment and/or awake the public opinion.

In order to gather the latter and evaluate the criteria proposed in the project, the ideas for green software criteria were presented to the survey participants. They were asked for their opinion if the aspect "should be labelled", "should rather be labelled", "should rather not be labelled" or "should not be labelled". Furthermore, they had the option to say that they "do not know".

Table 3 lists the results of the responders' ratings of the proposed criteria for green software products. While especially efficiency aspects (77.5 % of the responders say, "energy efficiency" should be labelled, 62.5 % vote for "hardware efficiency") and platform independence seem to be important to be labelled from the survey participants' view, hardware sufficiency and quality of product information might be less important, but still relevant, for a certification.

Table 3 *Answers to the question "Which ones of these environmental issues of software should be labelled?", sorted by the numbers of mentions of "should be labelled"*

Environmental issue	**Should be labelled**
(1) Energy efficiency	77.5 %
(2) Hardware efficiency	62.5 %
(3) Platform independence	56.9 %
(4) Backward compatibility	52.7 %
(5) Uninstallability	48.5 %
(6) Independence of outside resources	46.6 %
(7) Portability	43.0 %
(8) Maintenance functions	40.4 %
(9) Transparency	40.0 %
(10) Hardware sufficiency	35.4 %
(11) Quality of product information	31.6 %

The overall results show that there seems to be an interest in the proposed aspects. The highest value for "should not be labelled" is 7.0 % – in case of uninstallability and 16.5 % for "should rather not be labelled" in case of quality of product information.

4.3.3 Additional ideas for criteria for green software products

In addition to the proposed criteria for an eco-label for software, the participants of the survey were interested in environmental and sustainable aspects regarding the producer of the product. 15 persons mentioned these in a free text field (of 84 comments in total). This includes, for example, the environment management of the producers, their infrastructure, development processes, and sustainable engagement.

Based on the survey results, an additional environmental impact of software that should be evaluated and certified is remote resources that are used by the software product. This includes, for example, the energy consumption of data centers, if the software communicates with the server as well as external processing power, memory space, updates, etc. Additionally, the participants want to know if this energy is renewable. They are also interested in the data traffic that is caused by using the software and its impacts onto the environment.

In addition to those, participants listed examples of topics regarding “updates” and “durability”, for example, the support of software during a fixed (required) time period, the planned number of updates or rather upgrade intervals.

Besides environmental sustainability aspects, six participants brought in the topic of “working conditions”. This refers to the whole supply chain: quarrying of resources, conditions during production, and involved sub companies (keyword “Corporate Social Responsibility”), treatment of humans and nature during the software production. Additionally, the survey responders suggested including the location where the software is produced primarily. Another topic that was referred to in the text field belongs to the distribution of software: download vs. data medium and the resulting resource consumption / energy efficiency / carbon footprint.

The participants of the survey were also interested in licensing aspects (keywords: “free software”, “Open Source”, “reselling”, “free data formats”, altogether seven comments) and data security (four comments).

Energy questions seemed to awaken a special interest of the participants, since they were, next to the proposed criterion “energy efficiency”, additionally mentioned in the text field. Aspects listed were: wear of the hardware by executing energy intensive software on it, possibility to adopt the energy modes of the operating system, and average energy consumption of the product.

The survey responders would like to have setting options to match the capacity of the product with its usage requirements. Depending on the context of use, the functionality of the software could be reduced to result in lower resource consumption. An example for this is to hide graphics. Additionally, they propose to provide product versions that provide less functions.

Another idea, in this direction, is the evaluation if additional programs that are undesired and needless to run the specific software are directly downloaded and installed with the software – with and without informing the user about it. In this context, products that are installed in "auto start mode" are mentioned especially.

Regarding software that is activated by default, the participants brought in the aspect of background activities of software. Mapped to environmental issues of software that might be certified, they propose to evaluate the resource consumption caused by these activities and the possibility to reduce or rather stop these background activities. Here, the evaluation should respect the benefit of the background activities.

Operability and usability are aspects that were mentioned by the survey participants and belong to social criteria (three mentions). In this context, the following questions were pointed out: Is there a possibility of "easy handling" of the software? How to rate the effort to learn how to use the software? Is the product intuitive and easy to use?

Additionally, there were some further ideas, just named once or twice (e.g. possibility to manage the permissions of the software, support like communities and documentation for beginners, and workflow: number of steps to fulfil specific tasks). They are not discussed in the paper but incorporated in future research activities.

4.4 Aspects influencing the acceptance of an eco-label for software

In order to provide recommendations to develop an eco-label for software products, the survey should come up with information regarding possible aspects influencing the acceptance of an eco-label for software. Thus, the survey participants were asked to rate specific statements belonging to the acceptance criteria, ranging from "I agree" to "I do not agree" (four options).

Table 4 *Results of the evaluation of statements belonging to the aspect "perceived ease of understanding"*

Statement	**I agree**	**I rather agree**	**I rather not agree**	**I do not agree**	**I do not know**
I would use additional information sources to understand the label	29.4 %	**40.0 %**	25.6 %	3.8 %	1.3 %
An eco-label for software products needs to be directly comprehensible	**71.3 %**	24.4 %	2.7 %	0.7 %	0.8 %
An eco-label needs to be self-explanatory	**53.4 %**	39.0 %	5.5 %	1.0 %	1.1 %

Please note: The answers have been interpreted against the assumption „the higher the agreement towards the statements, the more important the aspect is for the responder.

According to the survey results, providing information, the awareness of the label (for example for other product groups), perceived ease of understanding (see Table 4) and personal habits are rather important. Additionally, the perceived usefulness of the certification should be noticeable. This means

that these aspects should be integrated in the label development process. In contrast, the impact of the subjective norm on the acceptance of the label seems to be small. The image, i.e. the general reputation of the label, is ranked neither important nor irrelevant.

The next step is to analyze how the different acceptance criteria (subjective norm, attitude towards environmental topics and product labelling, habits, costs, information, awareness, image, perceived usefulness, perceived ease of understanding, intentional to use, usage behavior) relate to each other. Possible theses are presented in [Kern 2017].

4.5 Initial recommendations

The following section provides initial recommendations for the development of an eco-label for software, based on the results of the survey. The main recipients for these recommendations are those who are involved in the label development process.

Recommendation I: *Keep it simple and understandable.* 71.3 % of the survey participants agreed to the statement "A certification of green software needs to be understandable at a glance." 53.4 % state that a product certification should be self-explanatory. However, about 69.4 % of the participants are willing to use additional sources of information to understand the certification (29.4 % "agree", 40.0 % "rather agree"). 56.7 % lay emphasis on the possibility to read up about a certification. Summarizing, it seems to be helpful to keep the certification as simple as possible and to provide additional information for everyone who wants to know details. The priority should be set to ease of understanding of the certification itself, instead of overloading it with information.

Recommendation II: *Use the prominence of existing eco-labels to push the awareness of an eco-label for software.* More than half of the survey participants considered drawing on well-known product certifications instead of developing completely new ones reasonable (36.0 % "I agree", 33.8 % "I rather agree"). This could be helpful in case of trust (49.4 % of the participants rather agreed and 27.9 % agreed to "My trust in known certification is higher than in those that are less known"). Trust seems to be important, since nearly half of the participants mind solely product certifications that are credible (45.2 % "agree", 40.3 % "rather agree"). Additionally, by extending the kind of products relating to a certification, it is possible to support awareness of new certified products. Regarding the latter point, 41.3 % of the survey participants consented to "Product certifications should cover a number of product groups in order to gain awareness".

Recommendation III: *Keep the marked relevance of products in mind.* In addition to functionality of the software product, the fact whether or not a product is known plays a significant role: Most of the survey participants said that they prefer products they know, even if they are not labelled as "being green" (63.3 %). Indeed, the participants ranged between "heeding new certifications" and "relying on products and vendors I know": 41.4 % rather not agreed to the statement "While buying, I do not heed new certifications but rely on products and vendors I know".

However, 38.5 % rather agreed to it. The decision might be influenced by the satisfaction with familiar products ("My satisfaction with familiar products influences if I search for new products": 49.0 % of the survey participants agreed, 43.3 % rather agreed). Hence, to create awareness of the certification (as soon as it is available) and of environmental issues of software in general, it seems to be sensible to get the big players, e.g. market-leading software vendors, on board.

Recommendation IV: *Set the focus on publishing a certification that is understandable at first glance, instead of creating a complex information system.* The issue of environmental impacts caused by software products seems to be new for most of the end users. Thus, most of the participants did not pay attention to a corresponding certification so far (56.3 % "agree", see Section 4.3.1). Nevertheless 51.7 % of those who took part in the questionnaire saw the sense in certifying green software products. According to the results of the survey, publishing an eco-label for software might also help to raise the interest regarding Green IT topics (42.6 % "rather agree", 33.1 % "agree") and to support the buyer decision process (49.3 % "rather agree", 40.7 % "agree"). Trying to implement a certification for green software products as early as possible might lead to reducing the complexity of the first version of the certification. Thus, I propose to concentrate on environmental issues first (see Section 2.1), instead of trying to also include social and economic sustainability aspects. Here, energy efficiency, hardware efficiency, and platform independency seem to be good starting points to be elaborated on. These aspects raise the user's interests (cf. Table 3) and might be easily understood because they are known from other products.

5 Limits

The presented survey provides insights into the awareness, interest, and ideas of users regarding environmental issues of software. However, there are some limits of the study as well as of the idea of certifying software products.

The aim of the survey was to get a comprehensive idea of end user's awareness of and interest in green software and its labelling. However, the participants of the survey were mainly people holding an academic degree and already interested in environmental topics (see Section 4.1). As a result, the opinions obtained through the presented survey might be biased and differ from real population distribution. The results might be more positive in the study presented than in general surroundings. Furthermore, being answered predominantly by those who have a connection to academia strengthens the assumption that the topic is more present and raising higher interests in research-related, rather than social communities, as mentioned in the introduction. The diversity of participants could be higher to reach a representative status. Nevertheless, the impressions in general as well as the positive evaluation of the suggested criteria (see Section 4.3.3) can be understood as confirmation of being on the right track when developing an eco-label for software. The dominant presence in communities

holding an academic degree can be a motivation to bring the topic to a larger target group and reach an orientation to practical implementation.

Overall, the survey has been conducted in an early phase of the label development process. This can be rated positive in regards of integrating user opinions, ideas and interests timely. However, it can also make it harder for those who evaluate the initial ideas for the label, since the ideas are not fixed in any aspect. Definition, criteria, form of representation, target groups, and stakeholders [Kern et al. 2015] are just the main aspects that need to be discussed while developing a certification.

Generally, the effect of eco-labels on the behavior of everyone dealing with software is not clear. Since there is no certification so far, no one knows if it will lead to a more environmental friendly way of using software products. The validity of the label is still uncertain and the comparability of software products is a great point of discussions, since there are many factors influencing the evaluation of software (e.g. hardware settings, usage patterns). Additionally, companies and developers are free in their choice if they want to eco-label their software products. Even if the declaration would be mandatory, the users might not take the certification into consideration while searching for new software products. The results of the survey indicate that those who participated would take considering a label into account – however, the functionality of the product is number one when it comes to selection criteria for products in a category (cf. Figure 5).

However, a first step to spread the idea of environmental impacts of software was reached. While the survey already brought this idea into minds of some software users, a certification will address even more persons in this context. Studies show that eco-labels can have a positive influence of the usage behavior and resulting energy consumption [Horne 2009]. Thus, it might be also successfully in case of software.

6 Conclusion and suggestion for future research

Nowadays, researching environmental issues of software is getting more and more attention. However, the topic is primarily known in research areas excluding developers and users of software. To counteract this, the idea is to develop an eco-label for software products. The contribution presents a user survey, pointing out if environmental issues of software are known to those using software and which aspects stir their interest. Summarizing the findings of the survey, the participants did not think about an eco-label for software so far, although they are generally interested in environmental topics. However, they can imagine taking “environmental issues” into account when searing for new software products. If the functionality of the product is the same between different vendors, most of the participants would go with the eco-labelled product. The findings point out that there is (rather) no awareness for, but interest in, the environmental impacts of software.

Based on the findings, first recommendations are presented that should drive the development of an eco-label of software forward: (i) Keep it simple and understandable. (ii) Use the prominence of existing eco-labels to push the awareness of an eco-label for software. (iii) Keep the marked relevance of products in mind. (iv) Set the focus on publishing a certification that is understandable at first glance, instead of creating a complex information system.

The idea of labelling software is to create an awareness for environmental impacts of software. This awareness is a first step to a more environmental friendly way of using ICT and especially software. The overall vision is to reduce the impacts of ICT on the environment.

The next steps in developing a label for green software products includes the evaluation of the devised set of criteria for sustainable software. Finally, the set should be transferred into awarding criteria for an eco-label of software. Thus, there need to be corresponding valuation methods, which are missing so far. The label should be supplemented by guidelines for developers on how to develop sustainable software. These recommendations need to be gathered from literature, research results etc., and presented in a compatible way to software developers. Another possibility is to bring the idea of sustainable software development into universities and present a teaching concept for computer science students. Hence, the set of criteria – based on measurements and user interests – will be the basis for different activities to bring the topic from science to society.

References

AGARWAL, S., NATH, A., AND CHOWDHURY, D. 2012. Sustainable Approaches and Good Practices in Green Software Engineering. *IJRRCS 3*, 1, 1425–1428. http://scholarlyexchange.org/ojs/index.php/IJRRCS/article/view/9903/7030.

ALBERTAO, F. 2004. *Sustainable Software Engineering*. http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering#about. Accessed 14 April 2017.

ALBERTAO, F., XIAO, J., TIAN, C., LU, Y., ZHANG, K.Q., AND LIU, C. 2010. Measuring the Sustainability Performance of Software Projects. In *2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China*, IEEE Computer Society, Ed., 369–373.

AMSEL, N., IBRAHIM, Z., MALIK, A., AND TOMLINSON, B. 2011. Toward sustainable software engineering: NIER track. Software Engineering ICSE 2011. In *Proceedings of the 33rd International Conference on Software Engineering*, 976–979.

BOZZELLI, P., GU, Q., AND LAGO, P. 2013. *A systematic literature review on green software metrics*. Technical Report: VU University Amsterdam. http://www. sis. uta. fi/ pt/TIEA5\ Thesis\ Course/Session\ 10 \ 2013\ 02\ 18/SLR \ GreenMetrics. pdf.

CALERO, C., BERTOA, M.F., AND ANGELES MORAGA, M. 2013a. A systematic literature review for software sustainability measures. In *2nd International Workshop on Green and Sustainable Software (GREENS)*, 46–53.

CALERO, C., MORAGA, M., AND BERTOA, M.F. 2013b. Towards a software product sustainability model. *arXiv preprint arXiv:1309.1640*.

CALERO, C., MORAGA, M.Á., BERTOA, M.F., AND DUBOC, L. 2015. Green Software and Software Quality. In *Green in Software Engineering*, C. CALERO AND M. PIATTINI, Eds. Springer, 231–260.

CALERO, C., AND PIATTINI, M. 2015. Introduction to Green in Software Engineering. In *Green in Software Engineering*, C. CALERO AND M. PIATTINI, Eds. Springer, 3–27.

CAPRA, E., FRANCALANCI, C., AND SLAUGHTER, S.A. 2011. Is software green? Application development environments and energy efficiency in open source applications. *Information and Software Technology 54*, 60–71. http://ac.els-cdn.com/S0950584911001777/1-s2.0-S0950584911001777-main.pdf?_tid=a288bc86-f725-11e1-960e-00000aacb362&acdnat=1346827861_377674c8edd54a640c2d57df2bd4951b.

CAPRA, E., FRANCALANCI, C., AND SLAUGHTER, S.A. 2012. Measuring Application Software Energy Efficiency. *IT Professional*, 54–61.

D'SOUZA, C., TAGHIAN, M., AND LAMB, P. 2006. An empirical study on the influence of environmental labels on consumers. *Corporate Communications: An International Journal 11*, 2, 162–173.

DICK, M., AND NAUMANN, S. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany*, K. GREVE AND A. B. CREMERS, Eds. Shaker, Aachen, 706–715.

DICK, M., NAUMANN, S., AND KUHN, N. 2010. A Model and Selected Instances of Green and Sustainable Software. In *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience. 9th IFIP TC 9 International Conference, HCC9 2010 and 1st IFIP TC 11 International Conference, CIP 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, J. BERLEUR, M. D. HERCHEUI AND L. M. HILTY, Eds. Springer, Berlin, Heidelberg, 248–259.

DOOKHITRAM, K., NARSOO, J., SUNHALOO, M.S., SUKHOO, A., AND SOOBRON, M. 2012. Green computing: an awareness survey among university of technology, mauritius students. In *Conference Proceeding of International Conference on Higher Education and Economic Development, Mauritius. Available from http://tec. intnet. mu/pdf% 20downloads/confpaper/confpaper091224. pdf*.

DURDIK, Z., KLATT, B., KOZIOLEK, H., KROGMANN, K., STAMMEL, J., AND WEISS, R. 2012. Sustainability guidelines for long-living software systems. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, 517–526.

FERREIRA, M. 2011. *Green Software Awareness Survey. Results*. Presented at Report Workshop Green Software Architecture, Tuesday 7 June 2011, Amsterdam, Netherlands, Amsterdam.

GRIEßHAMMER, R., BUCHERT, M., GENSCH, C.-O., HOCHFELD, C., MANHART, A., REISCH, L., AND RÜDENAUER, I. 2007. *PROSA - Product Sustainability Assessment. Guideline*. http://www.prosa.org/fileadmin/user_upload/pdf/leitfaden_eng_final_310507.pdf.

GROSSKOP, K., AND VISSER, J. 2013. Identification of application-level energy optimizations. In *ICT4S ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 101–107.

HERZOG, C., LEFÉVRE, L., AND PIERSON, J.-M. 2015. Actors for Innovation in Green IT. In *ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 49–67.

HILTY, L., LOHMANN, W., BEHRENDT, S., EVERS-WÖLK, M., FICHTER, K., AND HINTEMANN, R. 2015. Green Software. Final report of the project: Establishing and exploiting potentials for environmental protection in information and communication technology (Green IT). Report commissioned by the Federal Environment Agency, Berlin, Förderkennzeichen 3710 95 302/3, 23.

HINDLE, A. 2016. Green software engineering: the curse of methodology. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 46–55.

HORNE, R.E. 2009. Limits to labels: The role of eco-labels in the assessment of product sustainability and routes to sustainable consumption. *International Journal of Consumer Studies 33*, 2, 175–182.

ISSA, T., ISSA, T., AND CHANG, V. 2014. Sustainability and green IT education: Practice for incorporating in the Australian higher education curriculum. *The International Journal of Sustainability Education 9*, 2, 19–30. https://espace.curtin.edu.au/bitstream/handle/20.500.11937/7873/194184_194184.pdf.

JAGROEP, E., BROEKMAN, J., VAN DER WERF, J.M.E., BRINKKEMPER, S., LAGO, P., BLOM, L., AND VAN VLIET, R. 2017. Awakening awareness on energy consumption in software engineering. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Society Track*, 76–85.

JOHANN, T., AND MAALEJ, W. 2013. Position Paper: The Social Dimension of Sustainability in Requirements Engineering. In *Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems*.

JOHANN, T., AND MAALEJ, W. 2015. Democratic mass participation of users in Requirements Engineering? In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, 256–261.

KERN, E. 2017. Communicating Environmental Issues of Software: Outline of an Acceptance Model. In *Advances and New Trends in Environmental Informatics: Stability, Continuity, Innovation*, V. WOHLGEMUTH, F. FUCHS-KITTOWSKI AND J. WITTMANN, Eds. Springer International Publishing, Cham, 335–345.

KERN, E., DICK, M., NAUMANN, S., AND FILLER, A. 2015. Labelling Sustainable Software Products and Websites: Ideas, Approaches, and Challenges. In *Proceedings of EnviroInfo and ICT for Sustainability 2015. 29th International Conference on Informatics for Environmental Protection (EnviroInfo 2015) and the 3rd International Conference on ICT for Sustainability (ICT4S 2015)*. Copenhagen, September 7 - 9, 2015, V. K. JOHANNSEN, S. JENSEN, V. WOHLGEMUTH, C. PREIST AND E. ERIKSSON, Eds. Atlantis Press, Amsterdam, 82–91.

KERN, E., DICK, M., NAUMANN, S., GULDNER, A., AND JOHANN, T. 2013. Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In *ICT4S ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 87–94.

KERN, E., DICK, M., NAUMANN, S., AND HILLER, T. 2014. Impacts of software and its engineering on the carbon footprint of ICT. *Environmental Impact Assessment Review 52*, 53–61. http://dx.doi.org/10.1016/j.eiar.2014.07.003.

KITCHENHAM, B., AND CHARTERS, C. 2007. Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. sn.

KOGELMAN, C.-A. 2011. CEPIS Green ICT Survey – Examining Green ICT Awareness in Organisations: Initial Findings. Carol-Ann Kogelman on behalf of the CEPIS Green ICT Task Force. *CEPIS UPGRADE XII*, 4, 6–10. http://www.cepis.org/upgrade/media/kogelman_2011_42.pdf.

LAGO, P. 2014. A master program on engineering energy-aware software. In *EnviroInfo 2014. Proceedings of the 28th International Conference on Informatics for Environmental Protection*, J. MARX GÓMEZ, M. SONNENSCHEIN, U. VOGEL, A. WINTER, B. RAPP AND N. GIESEN, Eds.

LAMI, G., AND BUGLIONE, L. 2012. Measuring Software Sustainability from a Process-Centric Perspective. In *Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2012 Joint Conference of the 22nd International Workshop on*, 53–59.

LAMI, G., FABBRINI, F., AND FUSANI, M. 2013. A Methodology to Derive Sustainability Indicators for Software Development Projects. In *Proceedings of the 2013 International Conference on Software and System Process*. ACM, New York, NY, USA, 70–77.

MAHAUX, M., HEYMANS, P., AND SAVAL, G. 2011. Discovering Sustainability Requirements: An Experience Report. In *Requirements Engineering: Foundation for Software Quality. 17th International Working Conference, REFSQ 2011, Essen, Germany, March 28-30, 2011*, Proceedings, D. M. BERRY AND X. FRANCH, Eds. Springer, Berlin, Heidelberg, 19–33.

MANOTAS, I., BIRD, C., ZHANG, R., SHEPHERD, D., JASPAN, C., SADOWSKI, C., POLLOCK, L., AND CLAUSE, J. 2016. An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of the 38th International Conference on Software Engineering Companion*, 237–248.

NAUMANN, S., DICK, M., KERN, E., AND JOHANN, T. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *SUSCOM 1*, 4, 294–304. doi:10.1016/j.suscom.2011.06.004.

ORGANISATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT (OECD) 2008. *Promoting Sustainable Consumption. Good Practices in OECD Countries*, Paris.

PANG, C., HINDLE, A., ADAMS, B., AND HASSAN, A.E. 2016. What do programmers know about software energy consumption? *IEEE Software 33*, 3, 83–89. https://www.researchgate.net/profile/Ahmed_E_Hassan/publication/282540010_What_do_programmers_know_about_software_energy_consumption/links/578fd90608ae4e917cff3919.pdf.

PEDERSEN, E.R., AND NEERGAARD, P. 2006. Caveat emptor-let the buyer beware! Environmental labelling and the limitations of 'green'consumerism. *Business Strategy and the Environment 15*, 1, 15–29. http://onlinelibrary.wiley.com/doi/10.1002/bse.434/abstract.

PENZENSTADLER, B. 2012. *Sustainability in Software Engineering: A Systematic Literature Review for Building up a Knowledge Base*. Accessed 10 July 2012.

PENZENSTADLER, B., AND FEMMER, H. 2013. A Generic Model for Sustainability with Process- and Product-specific Instances. In *Proceedings of the 2013 workshop on Green in/by software engineering*, S. MALAKUTI, Ed. ACM, [S.l.], 3–8.

PENZENSTADLER, B., FEMMER, H., AND RICHARDSON, D. 2013. Who is the advocate? stakeholders for sustainability. In *Green and Sustainable Software (GREENS). 2nd International Workshop on Green and Sustainable Software*, 70–77.

PENZENSTADLER, B., RATURI, A., RICHARDSON, D., CALERO, C., FEMMER, H., AND FRANCH, X. 2014a. *Systematic Mapping Study on Software Engineering for Sustainability (SE4S) - Protocol and Results* ICS2 221. Institute for Software Research, University of California, Irvine, Irvine.

PENZENSTADLER, B., RATURI, A., RICHARDSON, D., AND TOMLINSON, B. 2014b. Safety, security, now sustainability: the non-functional requirement for the 21st century.

PROCACCIANTI, G., LAGO, P., AND BEVINI, S. 2014. A systematic literature review on energy efficiency in cloud software architectures. *Sustainable Computing: Informatics and Systems*.

RAHBAR, E., AND ABDUL WAHID, N. 2011. Investigation of green marketing tools' effect on consumers' purchase behavior. *Business strategy series 12*, 2, 73–83.

RAMACHANDIRAN, C.R. 2012. Green ICT practices among tertiary students: A case study. In *Business, Engineering and Industrial Applications (ISBEIA), 2012 IEEE Symposium on*, 196–201.

RASHID, N.R.N.A. 2009. Awareness of eco-label in Malaysia's green marketing initiative. *International Journal of Business and Management 4*, 8, 132.

ROTHERHAM, T. 2005. The trade and environmental effects of ecolabels: Assessment and response. *United Nations Environment Programme*.

SELYAMANI, S., AND AHMAD, N. 2015. Green Computing: The Overview of Awareness, Practices and Responsibility Among Students in Higher Education Institutes. *Journal of Information Systems Research and Innovation*. https://seminar.utmspace.edu.my/jisri/download/Vol9_3/JISRI_dec15_P4_Asnita.pdf.

TAINA, J. 2011. Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In *Green ICT: Trends and Challenges*, J.-C. LOPEZ-LOPEZ, G. SISSA AND L. NATVIG, Eds., 22–27.

TORRE, D., PROCACCIANTI, G., FUCCI, D., LUTOVAC, S., AND SCANNIELLO, G. 2017. On the presence of green and sustainable software engineering in higher education curricula. In *Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials*, 54–60.

WILSON, M. 2004. *Constructing measures: An item response modeling approach*. Routledge.

Communicating Environmental Issues of Software: Outline of an Acceptance Model

– Paper 4 –

Eva Kern[a,b*]

[a] Leuphana University Lueneburg, Scharnhorststrasse 1, 21335 Lueneburg, Germany

[b] Institute for Software Systems in Business, Environment, and Administration, Trier University of Applied Sciences, Environmental Campus Birkenfeld, P.O. Box 1380, 55761 Birkenfeld, Germany

Abstract. During the last years, the research activities regarding software and its environmental impacts could find their way into the field of "Green IT". Thus, researchers became aware of the fact that software is one of the drivers of the energy consumption by ICT. However, the awareness for these aspects could be much higher – especially in the non-scientific area. On the one side, software developers should be aware of green strategies of software engineering. On the other side, those using the products need to be responded to the effects of using ICT products onto the environment. One idea to transfer the environmental effects of software into a social issue is to create an eco-label for software products. Next to defining criteria, such a label could be laid on, and methods to evaluate software products, it seems to be helpful to identify aspects influencing the acceptance of a certification for green software products. In this context, acceptance means taking the eco-label into account while searching for new software. Hence, the following paper aims at identifying those aspects by applying the Technology Acceptance Model (TAM 2) to the specific case of labelling green software products. We will present a first version of an acceptance model for a label for green software products. It is still work in-progress and needs to be evaluated as a next step. Generally, the aim is to create a tool that can be used to develop an eco-label for software products that will be strongly accepted.

1 Introduction

According to Stobbe et al. [2015], the energy demand of ICT reduced about 15 % in Germany from 2010 to 2015 and might continue to decrease in the next years. This trend is mainly caused by two aspects: (1) the technical improvement of the devices used in households and on workplaces and (2)

the regulation of product characteristics based on European Ecodesign Directive and European Ecolabel.

Regarding the relation between ICT and the environment, ICT can be generally understood as "part of the problem" and "as part of the solution." [Hilty and Aebischer 2015] That means, on the one hand, using ICT does have (negative) effects onto the environment, e.g. energy and resource consumption. These effects should be reduced ("Green IT"). On the other side, ICT can be used to support activities for the environment ("Green by IT"). Examples for both aspects can be found e.g. in [Hilty and Aebischer 2015] and [Naumann et al. 2011].

Current implemented activities mainly focus on the hardware side of ICT. Indeed, software issues do also have an environmental impact [Naumann et al. 2011]. To address environmental issues of software, the terms "Green Software" and "Sustainable Software" became established in research contexts. According to Dick and Naumann [2010], "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." Next to this definition, one can find a few more definitions in the current literature with slightly different foci. However, a standardized characterization for green software products and corresponding criteria are still missing. [Kern et al. 2015]

In order to enhance the above mentioned effects of the technical improvements and ecofriendly product designs of ICT, the idea is to transfer the principle of benchmarking and informing about environmental impacts to encourage more energy efficient hardware products to software products and increase the positive effects in that way. Within these activities different stakeholders, e.g. developers, administrators, ICT companies, and software users, should be integrated. [Penzenstadler et al. 2013; Kern et al. 2015]

This paper focusses on the end users of software. The question is how to draw attention to the topic on the user side. The idea is to create a label for green software products and, thus, inform about environmental issues of software. As far as we know, there is no eco-label for software products so far. We assume that the information process leads to a more environmental-friendly usage of ICT and especially software products. This could further reduce the negative environmental impacts in this area.

Generally, the effort of developing a label is more justifiable if the user's interest in the label is high and it is accepted by them. In this context, acceptance stands for taking the label into account while searching for new software products. In order to find out (1) which aspects influence the acceptance of a label for green software products and (2) how these factors are related to each other, we outline an acceptance model for a label for green software products (Labelling Green Software Products - LGSP). This model will be introduced in the following paper. It is still work in-progress.

That means, we will introduce a first version of the model. In order to assess it, the next step will be a survey evaluating the proposed factors. Overall, the aim is to create a tool that can be used during the develop process of an eco-label for software products; leading to a label that will be strongly accepted.

The paper is structured as follows: Section 2 presents current research activities in the context of environmental impacts and labelling of software as well as information about the acceptance of eco-labels. Based on that, the idea of the label and potential stakeholders in this context will be pointed out in Section 3. The description of the acceptance model, including the influencing aspects and related hypotheses, represents the main part of the paper (Section 4). Following to that, we will present a conclusion and an outlook how to go on in researching the acceptance and the development of a label for green software products in Section 5.

2 Related Work

Additionally to researching environmental impacts of hardware products current research activities are paying more and more attention to the software side. Thus, the number of publications dealing with criteria and metrics for sustainable or rather green software products is growing (e.g. [Albertao 2004; Taina 2011; Bozzelli et al. 2013; Kern et al. 2013; Calero et al. 2015]). This is just one aspect of sustainable software and its engineering. Further topics are, among others, procedure models, energy issues and measurements, quality aspects, sustainability issues in requirements engineering and general impacts of ICT on the environment. The idea of labelling green software products is, in comparison with these topics, relatively new [Kern et al. 2015]. Accordingly, many research questions arge to be answered. One of these questions is, next to the development of the label itself, the analysis of the potential acceptance and the effectiveness.

It is not yet possible to make a clear statement about the acceptance or rather the effectiveness of eco-labels. This is mainly due to missing data. [Rotherham 2005] Above, there can be neither a common proposition about the effects of eco-labels nor about the intention to buy labelled products. [Rahbar and Abdul Wahid 2011] Indeed, Loureiro et al. [2001] conclude, referring to different researchers, that “a change in labeling or information can change consumers’ perceptions and behavior”. The label seems to go towards influencing the preferences of the purchaser more than the choice of products. [Loureiro et al. 2001] Overall, the eco-labels can be seen as communication and information tools for consumers. In this context, Rotherham talks about “a tool that communicates expectations and requirements to whoever is interested”, calls it a “catalyst for change” and points out that it “can stimulate a process of environmental awareness raising in companies and the general public” [Rotherham 2005]. To do so, working as a complement to other initiatives seems to be sensible.

Apart from the lack of data, a general statement is challenging caused by the multiple characteristics of eco-labels and the varying contexts the labels are used in [Loureiro et al. 2001; Rotherham 2005; Horne 2009; Rahbar and Abdul Wahid 2011]. The effectiveness also depends on the kind of product [Loureiro et al. 2001]. Thus, giving generic recommendations for creating and using eco-labels is not possible. However, there are various types of studies addressing product labels to spread environmental issues. The foci of these studies differ: they deal with selected product categories (e.g. food products [Loureiro et al. 2001; Grunert et al. 2014], vehicles [Teisl et al. 2008]), specific labels (e.g. ENERGY STAR [EPA Office of Air and Radiation, Climate Protection Partnerships Division 2015], Blue Angel [Dirksen 1996], EU Energy Label [Heinzle and Wüstenhagen 2012]), influence of and consumer trust in eco-labels and product-labels in general [Thøgersen 2000; Borin et al. 2011; Atkinson and Rosenthal 2014; London Economics and Ipsos 2014] or green market and their effects in specific areas (e.g. Malaysia [Rahbar and Abdul Wahid 2011] or Europe [Allison and Carter 2000]).

Since there is no eco-label for software products so far, these research results refer to other products, contexts, and local regions. Nevertheless, they could be taken as inspiration while developing a green software product label. Besides, developing an eco-label is accompanied by coming up with new methods and generating new data. This new-generated knowledge might also contribute to an environmental awareness and positive effects.

3 Green software products: labelling and stakeholders

In the following, we will shortly describe the idea of the label and its potential stakeholders. This accounts for the statement by Rotherham [2005]: “It is time to shift the discussion away from ecolabels generally and towards their specific characteristics”.

1. The label in mind should award environmental friendly products, i.e. the underlying criteria evaluate environmental impacts of the product but do not include social or economic aspects.
2. It should award the product itself, i.e. without the development and distribution process, the surroundings, and the organization developing the product. The label refers to the whole life cycle of the product but does not include anything that is not part of the product. That means it values e.g. the algorithm and the modules of an application software but not the packing of it.
3. It should award “green in software” aspects. That means the product itself should be as environmental friendly as possible. Products that support environmental friendly processes or rather make them more environmental friendly (“green by software”) but are not green itself are not included.

In a general sense, we see developers, purchasers, administrators, and users as stakeholders regarding green software [Naumann et al. 2011]. Regarding an interest in environmental products

D'Souza et al. [2006] describe four types of consumers: environmentally green consumers, emerging green consumers, price sensitive green consumers and conventional consumers. Supplementary, Rotherham [2005] indicates that, next to the final consumer, retails and governments are also important purchasers. As a first step, our acceptance model (Section 4) generally takes every possible person interested in software products into account. If the proposed constructs, their connections to each other or the relevance of them differ depending on the stakeholder, their attitude towards green issues and / or the context (private or professional purchaser) will be analyzed in a next step by implementing an appropriate survey.

4 Acceptance Model "Labelling Green Software Products"

The proposed acceptance model regarding a certification of green software products (LGSP) is oriented towards the Technology Acceptance Model 2 (TAM 2) by Venkatesh and Davis [2000] Whereas the original TAM states that *Perceived Usefulness* and *Perceived Ease of Use* influences the *Attitude Toward Using* and thus the *Usage of a Technology*, TAM 2 is expanded by factors divided into *Social Influence* and *Cognitive Instrumental Process*. We choose this model since the field of application is environmental computer science: the idea of our model is to find a construct to analyze factors influencing the acceptance of an eco-label for software products and thus the acceptance of green software products in contrast to non-green software products. We assume that both, the attitude toward using those products and social factors, play a significant role in the decision process of buying green or non-green software products. The model can be seen as a tool being useful in the development of green software products and their certification. As a first step, we check TAM 2 for applicability regarding labelling green software products.

Figure 1 shows the proposed model. The theoretical constructs and the causal relationships of the model will be presented in the following sections. Additionally, we propose hypotheses about the relation of the constructs. These hypotheses need to be validated in a next step. They refer to the stakeholders described in Section 3 and are related to a specific label for green software products rather than general eco-labels. Finally, the proposed model will be compared to TAM 2.

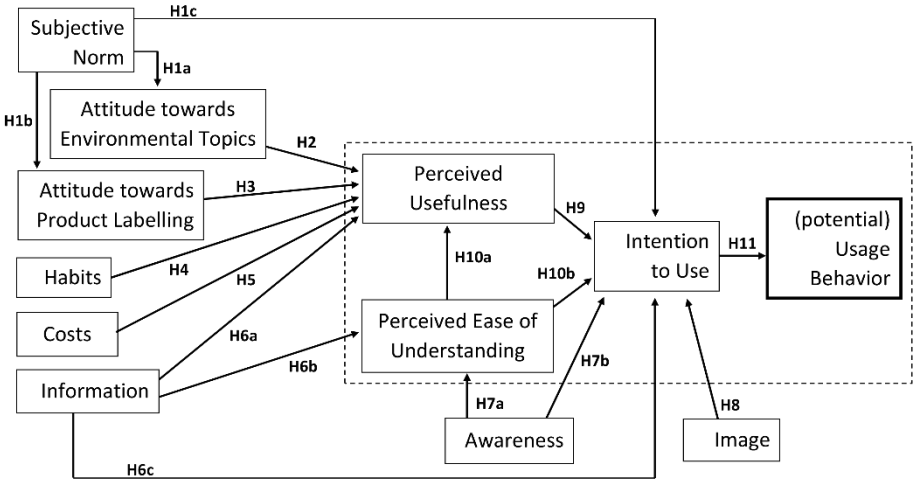

***Figure 1**. Outline of an acceptance model for a label for green software products (LGSP Acceptance Model)*

4.1 Constructs related to the Technology Acceptance Model

The core of the LGSP Acceptance Model comprises the well-known Technology Acceptance Model (TAM) by Davis Jr [1986]. Here, we use the construct names presented in [Venkatesh and Davis 2000] (e.g. "Intention to Use" instead of "Attitude towards Using" and "Usage Behavior" instead of "Actual System Use"). In this context, the construct *Usage Behavior* is understood as the potential use of a label for green software products. We call it "potential" since there is no label for these kinds of software products so far. Using a label means that the label is taken into account while searching for new software products. Precedent to that, there needs to be an *Intention to Use*. Regarding a label for green software products this construct stands for the general notion that an attention to a certification of environmental impacts caused by software makes sense. The *Intention to Use* is caused by the *Perceived Usefulness* and the *Perceived Ease of Understanding*. Here, the proposed model differs from TAM in calling the construct *Perceived Ease of Understanding* instead of talking about "Use". We see the importance in understanding the label including its statements or rather the idea behind the label. It seems to be decisive of paying attention to green issues of software.

H9 The higher the *Perceived Usefulness* that is produced by the certification, the higher the *Intention to Use*.

H10a The higher the *Perceived Ease of Understanding* of the certification, the higher the *Perceived Usefulness*.

H10b The higher the *Perceived Ease of Understanding* of the certification, the higher the *Intention to Use*.

H11 The higher the *Intention to Use*, the higher the potential *Usage Behavior*.

4.2 Constructs related to the person of interest

Next to the four constructs of the TAM, the proposed model contains aspects related to the person who should take the green software product label into account. According to D'Souza et al. [2006], having knowledge about environmental issues has the potential to increase the awareness and consumer's positive attitude towards green products. Thus, generating knowledge seems to be the first step to create an interest into environmental topics. It can result in a high involvement concerning environmental issues and favorable influence a person's eco-behavior. [D'Souza et al. 2006; Teisl et al. 2008; Rahbar and Abdul Wahid 2011] Hence, we assume that it has a positive effect onto the acceptance of the label if the person has a positive attitude towards environmental topics as well as towards product labelling in general. The construct *Attitude towards Environmental Topics* refers to the personal interest in environmental topics and the relevance of environmental questions for the personal life. It can be positive or negative. Similarly, *Attitude towards Product Labelling* considers the personal view on product certifications. The last one is to be seen in conjunction with trust that can be understood as one premise of environmentally friendly purchase decisions [Teisl et al. 2008]. Additionally to the individual believes, a person is guided by her own habits and trades off new against already-known aspects. D'Souza et al. [2006] point out that "Consumers appear to be somewhat less inclined (31.6 per cent) to consider known brands as being environmentally safe and seem to rely more on their own experience (66.5 per cent) in selecting environmentally safe products". This conduct is represented by *Habits*. Not only personal interests and practices may have an influence on the acceptance of a product label. Above, the social circumstances may play a role. Venkatesh and Davis [2000] talk about *Subjective Norm* and define it, according to [Fishbein and Ajzen 1975], as a "person's perception that most people who are important to him think he should or should not perform the behavior in question". The social environment of a person brings together personal experiences and leading options of the ones living with the person. Further explanations to the "faith in the co-behavior of others" [Teisl et al. 2008] can be found e.g. in [Berger and Corbin 1992; Gould and Golob 1998].

H1a The more positive the *Subjective Norm* is about environmental topics, the more positive is the personal *Attitude towards Environmental Topics*.

H1b The more positive the *Subjective Norm* is about product labelling, the more positive is the personal *Attitude towards Product Labelling*.

H1c The more the *Subjective Norm* tends to pay attention towards certifications, the higher the *Intention to Use*.

H2 The more positive the personal *Attitude towards Environmental Topics*, the higher the *Perceived Usefulness*.

H3 The more positive the *Attitude towards Product Labelling*, the higher the *Perceived Usefulness*.

H4 The more important individual *Habits* for the person, the lower the *Perceived Usefulness* of the certification.

4.3 Constructs related to the product itself

Additionally to the personal interests and preferences, the label itself matters in order to find out if a label for green software products is taken into account while searching for new products or not. First of all, we think that the awareness of the product label plays a role. The construct *Awareness* covers if the label itself – in other contexts or related to other products – is generally known. Whereas *Image* is related to the person of interested in the TAM 2 by Venkatesh and Davis [2000] the construct concerns the reputation of the label in the proposed LGSP model. Is this kind of label generally accepted in the society and well-reputed? Next to the standing, financial issues might have an influence onto the acceptance of a green software label. Certified products might be more expensive than non-certified ones. According to Loureiro et al. [2001] and Rotherham [2005], it is unclear if there is a higher willingness to pay for a product with an eco-label. The construct *Costs* brings this aspect into the proposed model. Furthermore, information material about the certification, criteria the rating is laid on, etc. might be relevant for the acceptance of the label. Above, the construct *Information* covers the relevance of the information content of the product certification itself. The aspect is related to the fact that, according to Horne [2009], consumers are looking for simple eco-labels providing clear information for decision making. However, one should be aware of oversimplifying the label in order to keep the label's credibility and effects [Teisl et al. 2008; Horne 2009].

H5 The higher the *Costs* for the certified software product, the higher the *Perceived Usefulness*.

H6a The more *Information* about the certification are available, the higher the *Perceived Usefulness*.

H6b The more *Information* about the certification are available, the higher the *Perceived Ease of Understanding*.

H6c The more *Information* about the certification are available, the higher the *Intention to Use*.

H7a The higher the certification *Awareness* in general, the higher the *Perceived Ease of Understanding*.

H7b The higher the certification *Awareness* in general, the higher the *Intention to Use*.

H8 The better the *Image* of the certification, the higher the *Intention to Use*.

4.4 Comparison to the Technology Acceptance Model 2

As already mentioned, the proposed model is based on the TAM 2, introduced by Venkatesh and Davis [2000]. In comparison to TAM 2, the constructs *Experience*, *Job Relevance*, *Output Quality*,

and *Voluntariness* were deleted. Concerning a product label instead of a technical system, we assume that those who are looking for (new) software products, are used to gather information, so that experiences in using different kinds of information sources are taken for granted. Similarly to that, we consider additional information to be generally sensible for the job to find new products and thus do not mention the *Job Relevance*. Taken a product label into account while choosing a product, does not lead to an output comparable to a technical system as addressed by TAM. Thus, we neither include the *Output Quality* nor *Result Demonstrability* in our model. Since it will not be possible to label every software product, the option to choose a green-labelled product is voluntary. Hence, the *Voluntariness* might only have an influence on the acceptance of a green software product label if every product would be certified (as green or non-green).

Although both models – TAM 2 and LGSP – contain the construct *Image*, the definition of it differs. Whereas Venkatesh et al. take the image of a person into account, our model refers the image to the product, here the label for green software products, itself. Nevertheless, there might be a connection between both views since it seems to be possible that the personal "status in one's social system" [Venkatesh and Davis 2000] benefits from using products labelled having a good image. Indeed, we will not analyze this connection here.

Venkatesh and Davis [2000] differentiate between social influence processes and cognitive instrument processes. According to that, it is possible to similarly class *Subjective Norm* in our model as social influenced process. The other constructs that are related to the person (*Attitude towards Environmental Topics* and *Product Labelling* as well as *Habits*) could be understood as more individual, but might also be influenced by social forces. Those constructs of LGSP that are related to the label itself might be seen as equivalences to the cognitive instrument determinants in TAM 2. However, we think, the differentiation between person- and product-related fits better in case of analyzing the acceptance of a label for green software products.

5 Conclusion and Outlook

Summarizing, the paper presents an outline of an acceptance model regarding a certification of green software products. The model is based on TAM 2 by Venkatesh and Davis [2000] and theoretically transferred to the new case of application. Whereas the model constructs related to the person should be especially taken into account while promoting the label for green software products, the constructs related to the product are important while developing the label. So far, there is no such kind of certification. Hence, the idea is to find out the aspects influencing the acceptance of the label and incorporate them into the labelling development process. For example, if the hypothesis "The higher the certification Awareness in general, the higher the Intention to Use" turns true, the development of a certification for green software should be based on well-known eco-labels. On the other hand, the

promotion of the label could e.g. focus on those who are already interested in environmental topics if the assumption that the acceptance of those who care about such issues is confirmed.

The next step is to evaluate the proposed acceptance model by conducting a user survey. Doing so, the hypotheses will be validated. Based on the results of the evaluation, the model will be adjusted and completed, if reasonable. Overall, the idea is to construe practical recommendations for labelling green software product from the resulting model (similarly to the given example above). In that way, the consumer should be integrated into the development process of an eco-label for software products. The aim is to design an information strategy including software users instead of simply presenting the results to them [Horne 2009]. Simultaneously to the label development, researching the environmental impacts of producing, using and deactivating software should go on since having accurate criterion for an eco-label is very important [D'Souza et al. 2006].

References

ALBERTAO, F. 2004. *Sustainable Software Engineering*. http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering#about. Accessed 14 April 2017.

ALLISON, C., AND CARTER, A. 2000. *Study on different types of Environmental Labelling (ISO Type II and IIN Labels): Proposal for an Environmental Labelling Strategy*. http://www.food-mac.com/Doc/200411/study_ecolabel_en.pdf. Accessed 16 January 2016.

ATKINSON, L., AND ROSENTHAL, S. 2014. Signaling the green sell: the influence of eco-label source, argument specificity, and product involvement on consumer trust. *Journal of Advertising 43*, 1, 33–45.

BERGER, I.E., AND CORBIN, R.M. 1992. Perceived consumer effectiveness and faith in others as moderators of environmentally responsible behaviors. *Journal of Public Policy & Marketing*, 79–89.

BORIN, N., CERF, D.C., AND KRISHNAN, R. 2011. Consumer effects of environmental impact in product labeling. *Journal of Consumer Marketing 28*, 1, 76–86.

BOZZELLI, P., GU, Q., AND LAGO, P. 2013. *A systematic literature review on green software metrics*. Technical Report: VU University Amsterdam. http://www. sis. uta. fi/ pt/TIEA5\ Thesis\ Course/Session\ 10 \ 2013\ 02\ 18/SLR \ GreenMetrics. pdf.

CALERO, C., MORAGA, M.Á., BERTOA, M.F., AND DUBOC, L. 2015. Green Software and Software Quality. In *Green in Software Engineering*, C. CALERO AND M. PIATTINI, Eds. Springer, 231–260.

D'SOUZA, C., TAGHIAN, M., AND LAMB, P. 2006. An empirical study on the influence of environmental labels on consumers. *Corporate Communications: An International Journal 11*, 2, 162–173.

DAVIS JR, F.D. 1986. *A technology acceptance model for empirically testing new end-user information systems: Theory and results*. PhD Thesis, Massachusetts Institute of Technology.

DICK, M., AND NAUMANN, S. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany*, K. GREVE AND A. B. CREMERS, Eds. Shaker, Aachen, 706–715.

DIRKSEN, T. 1996. HEWLETT-PACKARD's Experience with the German BLUE ANGEL Eco-label. In *Electronics and the Environment, 1996. ISEE-1996., Proceedings of the 1996 IEEE International Symposium on*, 302–304.

EPA OFFICE OF AIR AND RADIATION, CLIMATE PROTECTION PARTNERSHIPS DIVISION. 2015. *National Awareness of ENERGY STAR for 2014. Analysis of CEE Household Survey*. https://www.energystar.gov/sites/default/files/asset/document/National_Awareness_of_ENERGY_STAR_2014_v6_508_1.pdf.

FISHBEIN, M., AND AJZEN, I. 1975. Belief, attitude, intention, and behavior: An introduction to theory and research.

GOULD, J., AND GOLOB, T.F. 1998. Clean air forever? A longitudinal analysis of opinions about air pollution and electric vehicles. *Transportation Research Part D: Transport and Environment 3*, 3, 157–169.

GRUNERT, K.G., HIEKE, S., AND WILLS, J. 2014. Sustainability labels on food products: Consumer motivation, understanding and use. *Food Policy 44*, 177–189.

HEINZLE, S.L., AND WÜSTENHAGEN, R. 2012. Dynamic Adjustment of Eco-labeling Schemes and Consumer Choice-the Revision of the EU Energy Label as a Missed Opportunity? *Business Strategy and the Environment 21*, 1, 60–70.

HILTY, L.M., AND AEBISCHER, B. 2015. ICT for Sustainability: An Emerging Research Field. In *ICT Innovations for Sustainability*. *Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 3–36.

HORNE, R.E. 2009. Limits to labels: The role of eco-labels in the assessment of product sustainability and routes to sustainable consumption. *International Journal of Consumer Studies 33*, 2, 175–182.

KERN, E., DICK, M., NAUMANN, S., AND FILLER, A. 2015. Labelling Sustainable Software Products and Websites: Ideas, Approaches, and Challenges. In *Proceedings of EnviroInfo and ICT for Sustainability 2015*. *29th International Conference on Informatics for Environmental Protection (EnviroInfo 2015) and the 3rd International Conference on ICT for Sustainability (ICT4S 2015)*. Copenhagen, September 7 - 9, 2015, V. K. JOHANNSEN, S. JENSEN, V. WOHLGEMUTH, C. PREIST AND E. ERIKSSON, Eds. Atlantis Press, Amsterdam, 82–91.

KERN, E., DICK, M., NAUMANN, S., GULDNER, A., AND JOHANN, T. 2013. Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In *ICT4S ICT for Sustainability*. *Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 87–94.

London Economics. 2014. *Study on the impact of the energy label-and potential changes to it-on consumer understanding and on purchase decisions*, London.

LOUREIRO, M.L., MCCLUSKEY, J.J., AND MITTELHAMMER, R.C. 2001. Assessing consumer preferences for organic, eco-labeled, and regular apples. *Journal of agricultural and resource economics*, 404–416.

NAUMANN, S., DICK, M., KERN, E., AND JOHANN, T. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *SUSCOM 1*, 4, 294–304. doi:10.1016/j.suscom.2011.06.004.

PENZENSTADLER, B., FEMMER, H., AND RICHARDSON, D. 2013. Who is the advocate? stakeholders for sustainability. In *Green and Sustainable Software (GREENS)*. *2nd International Workshop on Green and Sustainable Software*, 70–77.

RAHBAR, E., AND ABDUL WAHID, N. 2011. Investigation of green marketing tools' effect on consumers' purchase behavior. *Business strategy series 12*, 2, 73–83.

ROTHERHAM, T. 2005. The trade and environmental effects of ecolabels: Assessment and response. *United Nations Environment Programme*.

STOBBE, L., PROSKE, M., ZEDEL, H., HINTEMANN, R., CLAUSEN, J., AND BEUCKER, S. 2015. *Entwicklung des IKT-bedingten Strombedarfs in Deutschland. Studie im Auftrag des Bundesministeriums für Wirtschaft und Energie Projekt-Nr. 29/14*. http://www.bmwi.de/BMWi/Redaktion/PDF/E/entwicklung-des-ikt-bedingten-strombedarfs-in-deutschland-abschlussbericht,property=pdf,bereich=bmwi2012,sprache=de,rwb=true.pdf.

TAINA, J. 2011. Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In *Green ICT: Trends and Challenges*, J.-C. LOPEZ-LOPEZ, G. SISSA AND L. NATVIG, Eds., 22–27.

TEISL, M.F., RUBIN, J., AND NOBLET, C.L. 2008. Non-dirty dancing? Interactions between eco-labels and consumers. *Journal of Economic Psychology 29*, 2, 140–159.

THØGERSEN, J. 2000. Psychological determinants of paying attention to eco-labels in purchase decisions: Model development and multinational validation. *Journal of Consumer Policy 23*, 3, 285–313. http://link.springer.com/article/10.1023/A:1007122319675#page-1.

VENKATESH, V., AND DAVIS, F.D. 2000. A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management science 46*, 2, 186–204.

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

– Paper 5 –

Eva Kern[a,b], Achim Guldner[b], Stefan Naumann[b]

[a] Leuphana University Lueneburg, Universitätsallee 1, 21335 Lueneburg, Germany

[b] Institute for Software Systems in Business, Environment, and Administration, Trier University of Applied Sciences, Environmental Campus Birkenfeld, P.O. Box 1380, 55761 Birkenfeld, Germany

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 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 [United Nations General Assembly 1987] 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 [van Heddeghem et al. 2014] and will consume even more in the future [Andrae and Edler 2015]. On the other hand, these technologies can support environmental processes and even make processes more sustainable in many different branches [Vickery and Mickoleit 2013]. 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" [Hilty et al. 2015]. 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)" [Ferreira 2011]. Referring to a sustainable development, Maevsky et al. [Maevsky et al. 2016] 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." [Maevsky et al. 2016]

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 implementation and into daily routines is missing – on the professional as well as on the private side [Dookhitram et al. 2012; Chitchyan et al. 2016; Pang et al. 2016].

Thus, after introducing the research field "green software engineering" (Section 2), we will present approaches on how to support especially software developers and users in considering environmental issues of software (Section 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

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

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 (Section 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" [Penzenstadler 2013b]. However, we do not aim at presenting a comprehensive literature overview with the following section. This can be found e.g. in [Penzenstadler et al. 2014; Anwar and Pfahl 2017].

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" [Penzenstadler 2013a]. While the so-called "absolute definition" refers to approaches that reduce the environmental impact of a process by using software, Taina [2011] 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 [Taina 2011]: (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. [Naumann et al. 2011; Penzenstadler et al. 2014; Kharchenko and Illiashenko 2016]). In addition to environmental, Lago et al. [2015] 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. [Naumann et al. 2011; Taina 2011; Calero et al. 2013b; Kern et al. 2013b]). While Taina [2011] distinguishes between feasibility, (carbon footprint, energy, travel, …), efficiency (CPU intensity, idleness, …), and sustainability (reduction, beauty, …), Calero et al. [2013b] 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. [2014] 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 [Calero et al. 2013a; Betz and Caporale 2014] 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" [Calero et al. 2013a]. 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." [Ahmad et al. 2014]. Thus, "sustainability of a software product [can be defined] as the capacity of developing a software product in a sustainable manner" [Calero et al. 2013b].

Summarizing, the commonalities in the understanding of the term "(green and) sustainable software" are [Kern et al. 2015a]: "(1) environmental and resource protection as well as (2) supporting sustainable development including the three pillars of sustainability, but setting priorities." None of the definitions stands in total contrast to the other ones but each of them represents a specific

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

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 [Penzenstadler 2013b]. 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." [Naumann et al. 2011]

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. [Kharchenko and Illiashenko 2016] and [Calero and Piattini 2015].

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 (Figure 1), presented in [Naumann et al. 2011], 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 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 [Schütte 2013], 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.

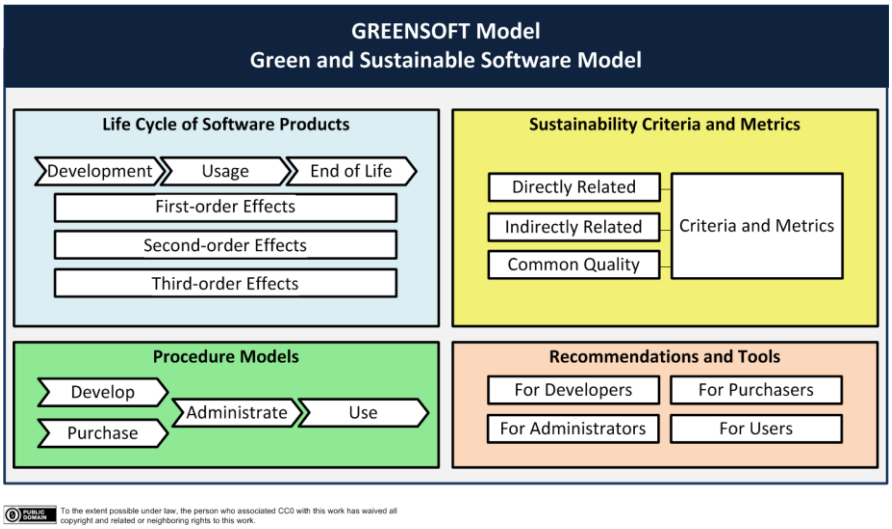

***Figure 1**. The GREENSOFT Model, a reference model for green and sustainable software engineering [Research Project "GREENSOFT" 2014]*

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. [2001], 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. [2013] 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 [Kern et al. 2013b]. Both models [Calero and Bertoa 2013; Kern et al. 2013b] 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 O. Gordieiev et al. [Gordieiev et al. 2015]. 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” [Gordieiev et al. 2015].

Lami et al. [2012] 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 software processes. Thus, referring to Figure 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 [Kern et al. 2015b].

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 sustainable

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 [Tischner et al. 2000]. 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 CO2 emissions. Additionally, also social aspects find their way into this consideration, e.g. working conditions [Dick and Naumann 2010]. 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 usage 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]". [Naumann et al. 2011] Thus, the question „Who are the stakeholders in context of a life cycle of (sustainable) software products?" arises. While Penzenstadler et al. [2013] identify stakeholders, who are important for successfully implementing sustainability issues, Herzog et al. [2015] 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." [Jagroep et al. 2017]. Maevsky et al. [Maevsky et al. 2016] and Chemeris

et al. [Chemeris et al. 2017] 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. [Chemeris et al. 2017] 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 [Chitchyan et al. 2016; Pang et al. 2016; Groher and Weinreich 2017]. Sustainability of software is, according to Groher et al. [2017], 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." [Groher and Weinreich 2017] If practitioners have knowledge in these contexts, they do just minimally take them into account during software development [Pang et al. 2016], and just connect them to specific kinds of software products (e.g. mobile development [Pang et al. 2016]). Generally, the transfer of knowledge to practice is missing [Chitchyan et al. 2016; Manotas et al. 2016]. 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" [Pang et al. 2016].

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" [Kern 2018]. 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.

***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	1. Demographic characteristics of the participants 2. Evaluation of criteria for green software products 3. 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

Similarly to software engineers, users might have some ideas on green computing [Dookhitram et al. 2012; Selyamani and Ahmad 2015], energy [Ferreira 2011], or rather sustainability issues of software [Torre et al. 2017], but awareness, information, and education are missing [Ferreira 2011; Dookhitram et al. 2012; Selyamani and Ahmad 2015; Pang et al. 2016; Torre et al. 2017]. Ferreira [2011] provides the hypothesis "Given the clear awareness, respondents are taking energy consumption into consideration when buying or developing software applications". In this context, Selyamani et al. [2015] 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" [Shenoy and Eeratta 2011; Mahmoud and Ahmad 2013], 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. [2017], 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" [Manotas et al. 2016].

Mahmoud et al. [Mahmoud and Ahmad 2013] 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. [Shenoy and Eeratta 2011] 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. [2012], Becker et al. [2016], 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. [2017]. 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 [Manotas et al. 2016]. 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” [Kern et al. 2014]. Hence, in order to meet the requirements of practitioners [Groher and Weinreich 2017], 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 [Dirlewanger 2006]. 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.

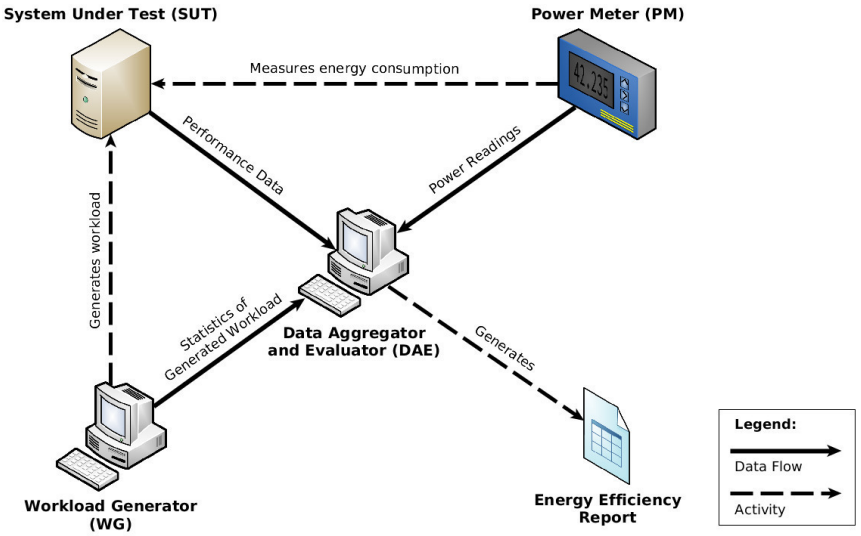

Figure 2. Setup for measuring power usage and hardware utilization of software (based on [Research Project "GREENSOFT" 2014])

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, 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 [Drangmeister et al. 2013; Kern et al. 2013a].

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 minutes), 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, Figure 3 shows the resulting graph of the energy measurement.

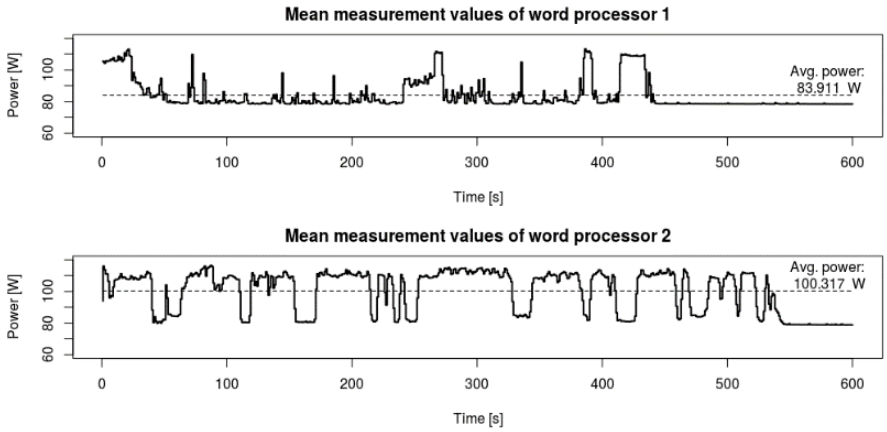

Figure 3 *Comparison of word processors: per second average of power consumption*

From the measured average power consumption, we calculated the consumed electrical energy [Wh]. Here, it is obvious that word processor 2 consumed more 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 (Figure 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." [Maevsky et al. 2017] 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 & 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 [United Nation Environment Programme 2003; Lago and Jansen 2010]. Referring to the GREENSOFT Model, depicted in Figure 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.

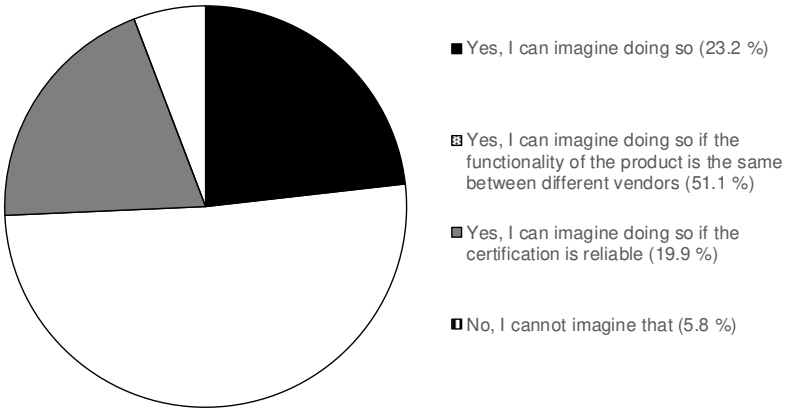

***Figure 4**. Results of the question "Can you imagine taking 'grading of the environmental impact of the product' as a criterion while searching for a software product?"*

However, as far as we know, there is no standardized label or any similar form of characterization for green software products, so far. According to our survey we conducted in 2016, the interest in such an eco-label is given [Kern 2018]. The potential of such a label is also present: in the USA, energy savings comparable to the energy consumption of ten million households have been achieved by

purchasing and using of products certified by the ENERGY STAR in 2002 [United Nation Environment Programme 2003].

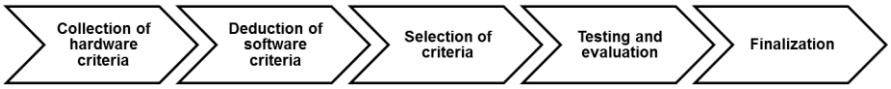

Figure 5 *Procedure approach for developing sustainability criteria for software*

In order to develop a label for green and sustainable software products, specific criteria, upon which a label could be based, are required [Kern et al. 2015a]. Figure 5Figure 1 depicts the procedure to create such a set of criteria.

Since there are Green IT labels referring to hardware products (e.g. ENERGY STAR, Blue Angel, TCO Certified), we analyzed the corresponding awarding guidelines, in order to find possibilities to transfer them to software products (Figure 5, Step 1: "Collection of hardware criteria" and Step 2: "Deduction of software criteria"). Additionally, we reviewed existing literature on criteria for sustainable software products and added further ideas for criteria. We focused on environmental aspects and the usage phase of a software product. Thus, the frame for the eco-label in mind can be summed up as follows:

1. The label should be awarded to environmentally friendly products, i.e. the underlying criteria should evaluate environmental impacts of the product but not include social or economic aspects.
2. It should be awarded to the product itself, i.e. without the development and distribution process, the surroundings, and the developing organization.
3. It should be awarded because of "green in software" aspects. This means the product itself should be as environmentally friendly as possible. Products that support environmentally friendly processes, or rather make them more environmentally friendly ("green by software"), but are not green themselves should not be included.

Starting with these three principles, we adjusted the criteria collection (Figure 5, Step 3: "Selection of criteria"). Additionally, as proposed by Mazijn et al. [Mazijn et al. 2004], we evaluated the resulting criteria collection by their relevance (Is the aspect relevant for software products?), measurability (Is it possible to measure the aspect through currently existing methods?), feasibility (Is it possible to provide information on the fulfillment of the aspect for software products?), and discrimination (Is the aspect different to other aspects?).

The selection process resulted in the following list of possible criteria for green and sustainable software products:

- Backward compatibility
- Energy efficiency

- Hardware efficiency
- Hardware sufficiency
- Independence of outside resources
- Maintenance functions
- Platform independence
- Portability
- Quality of product information
- Transparency
- Uninstallability

The next step (Figure 5, Step 4) was to evaluate these criteria ideas. This was done by (1) evaluating the social interest in the proposed criteria and (2) proof of applicability. The social evaluation was done by our user survey, addressing users of software products [Kern 2018]. As a result, especially the efficiency aspects (77.5 % say "energy efficiency should be labelled", 62.5 % say "hardware efficiency should be labelled") and platform independence (56.9 %) raise the user's interests. The result goes well together with the statement by Kharchenko et al. [Kharchenko and Illiashenko 2016]: "Energy saving and energy efficiency are the main characteristics of green computing."

In order to prove the relevance and verifiability of the criteria ideas, we evaluated the software products by measurements as described in Section 3.1. Besides black-box measurements (e.g. in case of resource and energy consumption), we used existing product information, research, and visual inspection for the evaluation of the criteria. The analysis of the tests includes a calculation of indicators of the proposed criteria, an evaluation of the selected information, a comparison within the product group, and a statistical presentation of the results.

After the evaluation of the criteria collection, we structured, described, and operationalized the criteria. The result of the criteria development process is a set of criteria that is a basis for awarding an eco-label for software products[3].

4 Conclusion and Outlook

Our contribution describes several approaches and procedure models, how green software can be classified and developed, and how different stakeholders like developers, purchasers, and end users can be supported. In doing so, we showed that the increasing demand of energy by information and communication technology (ICT) is caused by hardware, but software and algorithms induce this

[3] http://green-software-engineering.de/criteria-catalog

consumption. Consequently, and looking at the complete life cycle of a software product, all phases from development, usage, and also disposal have to be considered. Besides several methodological and practical procedures, a deeper awareness for the role of software regarding energy and resource consumption is necessary, as well. In our contribution, we presented two exemplary concepts on how to increase this awareness for 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 environmental issues of software products and thus create more transparency in this context. To be more concrete, we presented and evaluated criteria and indicators, upon which a label could be based.

Future work will test further software products and also modules regarding their "greenness". The overall goal is that these developed criteria are applied on a regular basis in software development processes like other non-functional requirements. This will reduce the energy and resource footprint of ICT and helps additionally to make software more efficient in several cases.

References

AHMAD, R., BAHAROM, F., AND HUSSAIN, A. 2014. A Systematic Literature Review on Sustainability Studies in Software Engineering. In *Proceedings of KMICe*.

ANDRAE, A.S.G., AND EDLER, T. 2015. On global electricity usage of communication technology: trends to 2030. *Challenges 6*, 1, 117–157.

ANWAR, H., AND PFAHL, D. 2017. Towards greener software engineering using software analytics: A systematic mapping. In *Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on*, 157–166.

BECKER, C., BETZ, S., CHITCHYAN, R., DUBOC, L., EASTERBROOK, S., PENZENSTADLER, B., SEYFF, N., AND VENTERS, C. 2016. Requirements: The key to sustainability. *IEEE Software 33*, 1, 56–65.

BERKHOUT, F., AND HERTIN, J. 2001. *Impacts of Information and Communication Technologies on Environmental Sustainability: speculations and evidence. Report to the OECD*. http://www.oecd.org/dataoecd/4/6/1897156.pdf. Accessed 2 March 2011.

BETZ, S., AND CAPORALE, T. 2014. Sustainable Software System Engineering. In *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on*, 612–619.

CALERO, C., AND BERTOA, M.F. 2013. 25010+ S: A software quality model with sustainable characteristics. Sustainability as an element of software quality. *Proceeding of the GIBSE*.

CALERO, C., BERTOA, M.F., AND ANGELES MORAGA, M. 2013a. A systematic literature review for software sustainability measures. In *2nd International Workshop on Green and Sustainable Software (GREENS)*, 46–53.

CALERO, C., BERTOA, M.F., AND MORAGA, M.Á. 2013b. Sustainability and Quality: Icing on the Cake. In *RE4SuSy@RE*.

CALERO, C., AND PIATTINI, M. 2015. Introduction to Green in Software Engineering. In *Green in Software Engineering*, C. CALERO AND M. PIATTINI, Eds. Springer, 3–27.

CHEMERIS, A., LAZORENKO, D., AND SUSHKO, S. 2017. Influence of Software Optimization on Energy Consumption of Embedded Systems. In *Green IT Engineering: Components, Networks and Systems Implementation*, V. KHARCHENKO, Y. KONDRATENKO AND J. KACPRZYK, Eds. Springer, 111–134.

CHITCHYAN, R., BECKER, C., BETZ, S., DUBOC, L., PENZENSTADLER, B., SEYFF, N., AND VENTERS, C.C. 2016. Sustainability design in requirements engineering: state of practice. In *Proceedings of the 38th International Conference on Software Engineering Companion*, 533–542.

DICK, M., AND NAUMANN, S. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany*, K. GREVE AND A. B. CREMERS, Eds. Shaker, Aachen, 706–715.

DIRLEWANGER, W. 2006. *Measurement and rating of computer systems performance and of software efficiency. An introduction to the ISO/IEC 14756 method and a guide to its application*. Kassel University Press, Kassel.

DOOKHITRAM, K., NARSOO, J., SUNHALOO, M.S., SUKHOO, A., AND SOOBRON, M. 2012. Green computing: an awareness survey among university of technology, mauritius students. In *Conference Proceeding of International Conference on Higher Education and Economic Development, Mauritius. Available from http://tec. intnet. mu/pdf% 20downloads/confpaper/confpaper091224. pdf*.

DRANGMEISTER, J., KERN, E., DICK, M., NAUMANN, S., SPARMANN, G., AND GULDNER, A. 2013. Greening Software with Continuous Energy Efficiency Measurement. In *Workshop Umweltinformatik zwischen Nachhaltigkeit und Wandel, Koblenz 2013*, 940–951.

FERREIRA, M. 2011. *Green Software Awareness Survey. Results*. Presented at Report Workshop Green Software Architecture, Tuesday 7 June 2011, Amsterdam, Netherlands, Amsterdam.

GORDIEIEV, O., KHARCHENKO, V., AND FUSANI, M. 2015. Software Quality Standards and Models Evolution. Greenness and Reliability Issues. In *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications*, 38–55.

Governing Concil of the United Nations Environment Programme. 2003. *Background Paper for the Ministerial Level Consultations: Promoting sustainable consumption and production patterns*, Nairobi.

GROHER, I., AND WEINREICH, R. 2017. An Interview Study on Sustainability Concerns in Software Development Projects. In *Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on*, 350–358.

HERZOG, C., LEFÉVRE, L., AND PIERSON, J.-M. 2015. Actors for Innovation in Green IT. In *ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 49–67.

HILTY, L., LOHMANN, W., BEHRENDT, S., EVERS-WÖLK, M., FICHTER, K., AND HINTEMANN, R. 2015. Green Software. Final report of the project: Establishing and exploiting potentials for environmental protection in information and communication technology (Green IT). Report commissioned by the Federal Environment Agency, Berlin, Förderkennzeichen 3710 95 302/3, 23.

JAGROEP, E., BROEKMAN, J., VAN DER WERF, J.M.E., BRINKKEMPER, S., LAGO, P., BLOM, L., AND VAN VLIET, R. 2017. Awakening awareness on energy consumption in software engineering. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Society Track*, 76–85.

KERN, E. 2018. Green Computing, Green Software, and Its Characteristics. Awareness, Rating, Challenges. In *From Science to Society*, B. OTJACQUES, P. HITZELBERGER, S. NAUMANN AND V. WOHLGEMUTH, Eds. Springer, 263–273.

KERN, E., DICK, M., DRANGMEISTER, J., HILLER, T., NAUMANN, S., AND GULDNER, A. 2013a. Integrating Aspects of Carbon Footprints and Continuous Energy Efficiency Measurements into Green and Sustainable Software Engineering. In *EnviroInfo 2013 - Environmental Informatics and Renewable Energies. 27th International Conference on Informatics for Environmental Protection. Proceedings of the 27th EnviroInfo 2013 Conference, Hamburg, Germany, September 2-4, 2013*, B. PAGE, A. FLEISCHER, J. GÖBEL AND V. WOHLGEMUTH, Eds. Shaker Verlag, Aachen, 300–308.

KERN, E., DICK, M., NAUMANN, S., AND FILLER, A. 2015a. Labelling Sustainable Software Products and Websites: Ideas, Approaches, and Challenges. In *Proceedings of EnviroInfo and ICT for Sustainability 2015. 29th International Conference on Informatics for Environmental Protection (EnviroInfo 2015) and the 3rd*

International Conference on ICT for Sustainability (ICT4S 2015). Copenhagen, September 7 - 9, 2015, V. K. JOHANNSEN, S. JENSEN, V. WOHLGEMUTH, C. PREIST AND E. ERIKSSON, Eds. Atlantis Press, Amsterdam, 82–91.

KERN, E., DICK, M., NAUMANN, S., GULDNER, A., AND JOHANN, T. 2013b. Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In *ICT4S ICT for Sustainability*. *Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 87–94.

KERN, E., DICK, M., NAUMANN, S., AND HILLER, T. 2014. Impacts of software and its engineering on the carbon footprint of ICT. *Environmental Impact Assessment Review 52*, 53–61. http://dx.doi.org/10.1016/j.eiar.2014.07.003.

KERN, E., NAUMANN, S., AND DICK, M. 2015b. Processes for Green and Sustainable Software Engineering. In *Green in Software Engineering*, C. CALERO AND M. PIATTINI, Eds. Springer.

KHARCHENKO, V., AND ILLIASHENKO, O. 2016. Concepts of Green IT Engineering: Taxonomy, Principles and Implementation. In *Green IT Engineering: Concepts, Models, Complex Systems Architectures*, V. KHARCHENKO, Y. KONDRATENKO AND J. KACPRZYK, Eds. Springer, 3–19.

LAGO, P., AND JANSEN, T. 2010. Creating environmental awareness in service oriented software engineering. In *International Conference on Service-Oriented Computing*, 181–186.

LAGO, P., KOÇAK, S.A., CRNKOVIC, I., AND PENZENSTADLER, B. 2015. Framing sustainability as a property of software quality. *Communications of the ACM 58*, 10, 70–78.

LAMI, G., FABBRINI, F., AND FUSANI MARIO. 2012. Software Sustainability from a Process-Centric Perspective. In *EuroSPI 2012, CCIS 301*, D. Winkler, R.V O'Connor and R. Messnarz, Eds. Springer, 97–108.

MAEVSKY, D.A., MAEVSKAYA, E.J., AND STETSUYK, E.D. 2016. Evaluating the RAM Energy Consumption at the Stage of Software Development. In *Green IT Engineering: Concepts, Models, Complex Systems Architectures*, V. KHARCHENKO, Y. KONDRATENKO AND J. KACPRZYK, Eds. Springer, 101-121.

MAEVSKY, D.A., MAEVSKAYA, E.J., STETSUYK, E.D., AND SHAPA, L.N. 2017. Malicious Software Effect on the Mobile Devices Power Consumption. In *Green IT Engineering: Components, Networks and Systems Implementation*, V. KHARCHENKO, Y. KONDRATENKO AND J. KACPRZYK, Eds. Springer, 155–172.

MAHAUX, M., AND CANON, C. 2012. Integrating the Complexity of Sustainability in Requirements Engineering Engineering. In *18th International Working Conference on Requirements Engineering: Foundation for Software Quality*. *Prodeedings of the Workshops RE4SuSy, REEW, CreaRE, RePriCo, IWSPM and the Conference Related Empirical Study, Empirical Fair and Doctoral Symposium*, R. B. SVENSSON, D. BERRY, M. DANEVA, J. DÖRR, S. A. FRICKER, A. HERRMANN, G. HERZWURM, M. KAUPPINGEN, N. H. MADHAJI, M. MAHAUX, B. PAECH, B. PENZENSTADLER, W. PIETSCH, C. SALINESI, K. SCHNEIDER, N. SEYFF AND I. VAN DE WEERD, Eds., 28–32.

MAHMOUD, S.S., AND AHMAD, I. 2013. A Green Model for Sustainable Software Engineering 2013. *International Journal of Software Engineering and Its Applications 7*, 4, 55–74. http://www.sersc.org/journals/IJSEIA/vol7_no4_2013/5.pdf.

MANOTAS, I., BIRD, C., ZHANG, R., SHEPHERD, D., JASPAN, C., SADOWSKI, C., POLLOCK, L., AND CLAUSE, J. 2016. An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of the 38th International Conference on Software Engineering Companion*, 237–248.

MAZIJN, B., DOOM, R., PEETERS, H., VANHOUTTE, G., SPILLEMAECKERS, S., TAVERNIERS, L., LAVRYSEN, L., AND VAN DUQUE RIVERA, J. 2004. *Ecological, Social and Economic Aspects of Integrated Product Policy. Integrated Product Assessment and the Development of the Label 'Sustainable Development´ for Products.* CP/20. SPSD II - Part I - Sustainable production and consumption patterns.

NAUMANN, S., DICK, M., KERN, E., AND JOHANN, T. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *SUSCOM 1*, 4, 294–304. doi:10.1016/j.suscom.2011.06.004.

PANG, C., HINDLE, A., ADAMS, B., AND HASSAN, A.E. 2016. What do programmers know about software energy consumption? *IEEE Software 33*, 3, 83–89.

PENZENSTADLER, B. 2013a. Towards a Definition of Sustainability in and for Software Engineering. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 1183–1185.

PENZENSTADLER, B. 2013b. What does Sustainability mean in and for Software Engineering? In *ICT4S ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich.

PENZENSTADLER, B., FEMMER, H., AND RICHARDSON, D. 2013. Who is the advocate? stakeholders for sustainability. In *Green and Sustainable Software (GREENS). 2nd International Workshop on Green and Sustainable Software*, 70–77.

PENZENSTADLER, B., RATURI, A., RICHARDSON, D., CALERO, C., FEMMER, H., AND FRANCH, X. 2014. *Systematic Mapping Study on Software Engineering for Sustainability (SE4S) - Protocol and Results* ICS2 221. Institute for Software Research, University of California, Irvine, Irvine.

RESEARCH PROJECT "GREENSOFT". 2014. *Website: Research Project "Green Software Engineering" - Downloads*. http://www.green-software-engineering.de/en/downloads.html.

SCHÜTTE, R. 2013. *Grundsätze ordnungsmäßiger Referenzmodellierung. Konstruktion konfigurations-und anpassungsorientierter Modelle* 233. Springer-Verlag.

SELYAMANI, S., AND AHMAD, N. 2015. Green Computing: The Overview of Awareness, Practices and Responsibility Among Students in Higher Education Institutes. *Journal of Information Systems Research and Innovation*. https://seminar.utmspace.edu.my/jisri/download/Vol9_3/JISRI_dec15_P4_Asnita.pdf.

SHENOY, S.S., AND EERATTA, R. 2011. Green software development model: An approach towards sustainable software development. In *India Conference (INDICON), 2011 Annual IEEE*, 1–6.

TAINA, J. 2011. Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In *Green ICT: Trends and Challenges*, J.-C. LOPEZ-LOPEZ, G. SISSA AND L. NATVIG, Eds., 22–27.

TISCHNER, U., DIETZ, B., MAßELTER, S., SCHMINCKE, E., PRÖSLER, M., RUBIK, F., AND HIRSCHL, B. 2000. *How to do EcoDesign? A guide for environmentally and economically sound design*. Verlag form, Frankfurt am Main.

TORRE, D., PROCACCIANTI, G., FUCCI, D., LUTOVAC, S., AND SCANNIELLO, G. 2017. On the presence of green and sustainable software engineering in higher education curricula. In *Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials*, 54–60.

United Nations General Assembly. 1987. *Report of the World Commission on Environment and Development. Our common future*. UN document no. A/42/427 English, New York.

VAN HEDDEGHEM, W., LAMBERT, S., LANNOO, B., COLLE, D., PICKAVET, M., AND DEMEESTER, P. 2014. Trends in worldwide ICT electricity consumption from 2007 to 2012. *Computer Communications 50*, 64–76. https://biblio.ugent.be/publication/5782416/file/5782424.pdf.

VICKERY, G., AND MICKOLEIT, A. 2013. Greener and Smarter: Information Technology can Improve the Environment in Many Ways. In *Broadband Networks, Smart Grids and Climate Change*, E. M. NOAM, L. M. PUPILLO AND J. J. KRANZ, Eds. Springer, 33–37.

Labelling Sustainable Software Products and Websites: Ideas, Approaches, and Challenges

– Paper A –

Eva Kern[a,b], Markus Dick[c], Stefan Naumann[b], Andreas Filler[b]

[a] Leuphana University Lueneburg, Lueneburg, Germany, mail@nachhaltige-medien.de

[b] Environmental Campus Birkenfeld, Birkenfeld, Germany, {s.naumann|a.filler}@umwelt-campus.de

[c] sustainablesoftwareblog@gmail.com

Abstract. The awareness for software as an important player regarding the energy consumption caused by ICT steadily increased in the past years. The impact of software on the energy consumption is also more and more accepted by the research community under the umbrella of sustainability in general. Nevertheless, the end user is still only slightly or not addressed in the research activities regarding the whole energy consumption of software over its complete lifecycle. Also other stakeholders, e.g. administrators, designers, developers etc., are not in the focus of creating awareness for the aforementioned topics. In this contribution, we therefore focus on ideas, approaches, and challenges in developing a general-purpose labelling process for green and sustainable software products and websites. At first we provide a literature roundup, followed by the elaboration of requirements for the creation of a sustainability label for software products in general based on already existing and new approaches. On a first attempt, we furthermore concentrate on a labelling process for sustainable as well as green websites and sum up with a discussion followed by an outlook on our future work.

Keywords. green software; web engineering; eco-label

1 Introduction and Motivation

The research activities in the area of green and sustainable software and its engineering are steadily increasing. The topic resulted in many contributions, journals, conferences, and workshops. Two major focus areas in the aforementioned research field are based on the questions how ICT can encourage sustainability (“sustainable by ICT”) and how ICT itself can be more sustainable

("sustainable in ICT"). In this context software is also an issue. It is, next to hardware, substantially responsible for the energy consumption caused by ICT as well as the positive impact of software in the context of sustainability in general. Hence, focusing on software aspects can be expressed by the terms "sustainable by software" and "sustainable in software". We will dwell on corresponding definitions in section 3.1.

For us the field of "sustainable by software" also covers the field of "green by software" since green spots on the environmental pillar of sustainability. Hence, when we talk about sustainable by software, green by software is also included in that. It is the same for green or rather sustainable (in) software. If one will concentrate just on the environmental part, the term "green" explicitly points that out.

However, especially the attention of the energy consumption of software could be much higher in order to identify and implement additional potentials. In particular, the end user is only slightly or not addressed within these research activities. Indeed, its influence is high and should not be underestimated. "Up to 90 % of the energy used by ICT hardware can be attributed to the application software running on it" [GeSI, Global e-Sustainability Initiative et al. 2013]. This statement of the GHG Protocol is only one example showing the importance of the usage phase and therefore the user in the context of the energy consumption of software and the overall ICT. Above, e.g. Zhang et al. [Zhang et al. 2014] deal with the role of users regarding energy consumption focusing on mobile devices.

Although the end user is in the focus of our work addressing all of the stakeholders, namely administrators, designers, developers, authors, and end users, makes sense to round up these activities. Here, possibilities how to create awareness upon the impact of software onto a sustainable development needs to be found. In our paper, we will focus on a possibility to address users and developers.

One possibility to include users and developers in the activities around sustainable software is the development of a label for this kind of software. There could be a certification of software products in the sense of quality labels and based on standardized criteria. A certification body like the environment agency or another similar control board could award the sustainable software product. In that way information on sustainability relevant aspects are spread and the transparency for sustainability aspects of ICT can be supported. This approach is successfully proved for other product categories, e.g. household appliances, organic products, and consumables. Within the ISO 14000 schema the International Organization for Standardization deals with environmental labels and declarations (ISO 14020 to 14025). Hence, a standardized certification process for sustainable software products should be laid on these declarations. In our paper we will not go into details of these documents because we focus on comparing existing approaches in sustainable software engineering. The standardization of the whole process is a vision of the future.

Our paper discusses requirements and existing as well as new approaches regarding a label for sustainable software products in order to point out the state of the research and next challenges in this context. The findings are mostly based on literature reviews, but also include new ideas. In addition to the general addressing of the challenges for labelling sustainable software products, the possibilities for a label for green websites are considered. Here, green web engineering [Dick et al. 2010] is seen as one specific part of sustainable software engineering. It bridges the field of web engineering and Green IT, especially focusing on the energy consumption of the web.

The paper is structured as follows: Section 2 looks at the related work and research activities regarding sustainable software. Section 3 goes into issues being relevant in creating a sustainability label, shows up existing approaches, and compares them to be able to come up with new approaches. Section 4 and 5 concentrate on web engineering by presenting an approach for sustainable websites (Section 4) and afterwards having a closer look at green websites (Section 5). Section 6 states open questions and points of discussion. At last, Section 7 presents our conclusions including findings of our work and a short outlook for future work. Overall, the idea is to present the state of research as basis for further developments and to launch a further dealing with the topic, e.g. applying corresponding case studies.

2 Related Work

Based on the definition of sustainable software [Dick and Naumann 2010] and its advancements (e.g. Calero et al. 2013a; Penzenstadler et al. 2014; Schmidt et al. 2014), the research results comprise models and approaches specifying the topic of sustainable software development and analyzing the energy and resource consumption of software [Naumann et al. 2011; Lami et al. 2012; Calero et al. 2013b; Mahmoud and Ahmad 2013]. Related to the software product itself there are approaches for criteria and metrics (e.g. Calero et al. 2013a; Betz and Caporale 2014; Penzenstadler et al. 2014). The evaluation and implementation of the criteria and metrics in practical projects is missing so far. Additionally, energy consumption measurements are an issue in the current research area. Here, for example, approaches of measuring the energy consumption of databases [Capra et al. 2010], virtual machines [Marcu and Tudor 2011], and mobile applications [Willnecker et al. 2014] exist. Another issue in energy efficient software development is the programming style and its relation to the resulting energy consumption (e.g. Noureddine et al. 2012; Sahin et al. 2012). A lack of current research approaches is the missing connection to real software usage. Thus, a next step is to develop usage scenarios in order to find suitable metrics and estimate the induced resource consumption over the whole life cycle of software products. This can result in classification for sustainable software products.

Beside the different research activities and because of the relevance of the usage phase for the resulting energy consumption induced by software [GeSI, Global e-Sustainability Initiative et al.

2013] the end user should be involved into the activities of sustainable software. One possibility to do so is to visualize the energy consumption or rather the environmental impact. Here, some prototypes are already published: the browser plugin "(Green) Power Indicator" shows if the server a website is hosted on is operated with renewable energies [Naumann et al. 2008]. Wilke et al. [Wilke et al. 2012] present a marking system for the energy consumption of smartphone apps. Further green web initiatives exist online, e.g. co2stats1, Greenanalytics2, Greenfox3.

3 Relevant Aspects and Approaches Regarding Sustainable Software

3.1 Definition

At first, in order to create a common understanding of the term "sustainable software product", a clarification and distinction will be elaborated in the following. So far, one can find a few versions of definitions in the literature that are mainly representing the results of projects with slightly different foci [Dick and Naumann 2010; Calero et al. 2013a; Penzenstadler 2013; Penzenstadler et al. 2014; Schmidt et al. 2014]. At next, we will evaluate these definitions and bring them together to build a basis for a future standardization and potential labelling for sustainable software products.

Dick et al. [Dick and Naumann 2010] published the first definitions for sustainable software and sustainable software engineering in 2010. Hence, the following definitions provide the basis for later published definitions: "***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.*" [Dick and Naumann 2010] and "***Sustainable Software Engineering*** *is the art of defining and developing software products in a way so that the negative and positive impacts on sustainability that result and/or are expected to result from the software product over its whole lifecycle are continuously assessed, documented, and optimized.*" [Dick and Naumann 2010]

Regarding the software product, Calero et al. [Calero et al. 2013a], Penzenstadler et al. [Penzenstadler et al. 2014], and Schmidt et al. [Schmidt et al. 2014] come out with additional definitions that include but slightly change the basic aspects of Dick et al. [Dick and Naumann 2010]. Whereas Dick et al. name explicitly the impacts on economy, Schmidt et al. drop this and point out the conservation of the nature and social fairness by laying emphasis on the "integrative" sustainability. Above that, they replace the "and/or" by "and" to underline the importance of long-term positive effects of the software product. The "by" aspect, that is to be seen as an addition in the definitions of

[1] http://www.co2stats.com/faq.php
[2] http://greenanalytics.ca/
[3] http://greencodelab.fr/en/content/greenfox-firefox-addon-to-measure-your-browser-consumption

Dick et al. and Schmidt et al., is focalized by Penzenstadler et al. They come out with "Software Engineering for Sustainability (SE4S)" and talk about the process that is supported by sustainable software. They highlight the purpose of the software product to support sustainability and reduce the characterization of the software itself to energy efficiency in the definition. However, Calero et al. see "sustainable" itself as a characterization for software products. They postulate to add sustainability as an additional non-functional requirement for software products. Hilty et al. [Hilty et al. 2015] also follow this approach. Hence, sustainability can be seen as a quality aspect that improves a software product.

All authors agree on the fact that sustainable software engineering is a precondition to develop a sustainable software product. Next to the definition by Dick et al., Calero et al. published a definition for sustainable software development. This definition gives priority to a software development "in which resource use aims to meet product software needs" [Calero et al. 2013a] and ecological sustainability. In contrast to Dick et al., they do not mention life cycle assessment aspects at all.

Summarizing, the different definitions have the following aspects in common: (1) environmental and resource protection as well as (2) supporting sustainable development including the three pillars of sustainability, but setting priorities. Overall, it is negligible on which one of the sustainability models the definition is based on since the overall aim is the same. A holistic view is important.

3.2 Identification of Criteria

If there is a common understanding for sustainable software products, the next step is to identify criteria. Here, the state of research is similar to the definition part. Many contributions deal with possible criteria, that can be brought together to find a common basis [Albertao et al. 2010; Capra et al. 2011; Taina 2011; Agarwal et al. 2012; Calero et al. 2013a; Calero et al. 2013b; Kern et al. 2013; Penzenstadler et al. 2014; Abdullah et al. 2015]. Indeed, an evaluation of the proposed criteria in the field is missing. Following, we will give an overview of the current status in theoretical elaborations and show how the approaches can fit together. By showing possibilities how to bring the criteria together and analyzing overlaps and specifics we will show up possibilities for structuring the criteria. The review will be laid on the criteria presented in the Quality Model for Green and Sustainable Software [Kern et al. 2013], complemented by results of newer publications. Hence, we will not describe the criteria in detail but reference corresponding publications instead. The aim is to create a basis for drawing up evaluations of identified criteria. The collection does not list must-have criteria for sustainable software products but presents possibilities that need to be further developed.

In order to structure criteria for sustainable software products, mainly the following approaches can be found: (1) structuring based on the three pillars of sustainability, (2) mapping the criteria onto the life cycle of software products [Albertao et al. 2010; Naumann et al. 2011] and/or onto (3) first-, second- and third-order effects [Naumann et al. 2011], and (4) dividing them into categories like (a)

common quality criteria, directly related and indirectly related criteria [Kern et al. 2013], (b) process and product aspects [Penzenstadler 2013] or (c) resource-oriented and well-being oriented sustainability indicators [Betz and Caporale 2014]. In many cases the approaches present criteria completed by sub criteria or sub characteristics [Taina 2011; Calero et al. 2013a; Kern et al. 2013]. Beyond that, the criteria are connected with either "sustainable in ICT" or "sustainable by ICT" [Naumann et al. 2011; Penzenstadler 2013]. They have either a positive or a negative impact on sustainable development.

	Life cycle phase the criteria is to be considered	
Main Criteria	**Development Phase**	**Usage Phase**
Efficiency defines how software behaves when it comes to saving resources and avoiding waste.	• Energy Efficiency • Number of Methods • Framework Entropy • Resource Optimization • Memory Usage	• Peripheral Intensity • Memory Usage • Hardware Utilization • CPU Intensity • Hardware Management • Runtime Efficiency • Idleness • Energy Efficiency • Performance (Efficiency) • Functionality • Reflectivity
Resource-oriented Feasibility considers how environmental aspects of a sustainable development are supported.	• Energy Consumption • Carbon Footprint • Waste • Travel • Infrastructure • Energy Type	• Energy Consumption • Energy Type • Carbon Footprint • Effectiveness
Well-being oriented Feasibility considers how social aspects of a sustainable development are supported.	• Supportability • Organizations Surroundings • Accessibility • Satisfaction	• Usability • Accessibility • Sociality • Satisfaction • Dependability • Beauty • Supportability • Freedom of Risk
Perdurability can be defined as the degree to which a software product can be modified, adapted and reused in order to perform specified functions under specified conditions for a long period of time.	• Reusability • Functional Suitability • Reduction • Modularity • Testability • Analysability • Modifiability	• Obsolescence • Functional Types • Reliability • Adaptability • Fit for Purpose • Reduction • Maintainability • Context Coverage • Portability • Maturity • Availability • Fault tolerance • Recoverability • Installability • Replaceability

Figure 1 *Draft of identified criteria for sustainable software products mapping to Efficiency, Feasibility & Perdurability*

For the purpose of finding criteria for sustainable software products on which a label can be based on, a summary of the specific criteria that can be found in the given literature is necessary. Summing up the different categorizations for sustainability criteria we come to three keywords that cover all of the known approaches: *Efficiency*, *Feasibility*, and *Perdurability*. In order to specify these, *Feasibility* is separated into *resource-oriented Feasibility* focusing on environmental impacts and *well-being oriented Feasibility* focusing on social impacts. These main criteria for sustainable software have to be specified in greater detail by dedicating suitable criteria. By rating the sub criteria the degree of fulfillment of the main criteria can be analyzed.

In Figure 1 we mapped the found criteria to the proposed main criteria and separated them by the development and the usage phase. The results of analyzing approaches for sustainability criteria are

not summed up or ordered to show the existing criteria flood. Moreover, the presented collection is not intended to be exhaustive.

It is obvious that the complexity to sort them is quite high and it is a big challenge to come to a standardization in this context. Hence, in order to specify the main criteria by finding corresponding sub criteria it is important to be sure about the aim and the focus of the label. The specification might also depend on the kind of software that should be labelled. As soon as these questions are answered and thus the general set-up for a label is defined, an award guideline has to be developed based on the criteria.

3.3 Form of Representation

By analyzing currently existing eco-labels we found a lot of representations: (1) standardized labels including required by law vs. voluntary, (2) differentiated vs. specific vs. multidimensional, and (3) many designs for labels for (a) products, (b) producers, and (c) branches. Moving forward in creating more and more labels leads to an information explosion users cannot handle. Hence, not only the definition and criteria need to be clear but also the form of presenting these product characterizations. In order to create a recognition value the following approaches for forms of representation of the sustainability of software products will be described with international examples of eco-labels.

1. Quality classes similar to the Green Power Indicator[4]: One has to define general quality criteria that have to be fulfilled by all of the sustainable software products. Additionally to that, the product can be ranked higher if it meets more criteria than the basic ones. This grading can be presented by different colors, e.g. green, yellow, and red.
2. Presenting a statement for the labelling similar to the Blue Angel[5]: Here, the environmentally friendly aspect of the specific product, meaning the protective goal, is directly shown on the label, e.g. “protects humans and the environment”.
3. Neutral label without presenting any grading or statement similar to the Energy Star[6] or the EU Ecolabel[7]: The label helps to identify sustainable software without giving any information about criteria.
4. Declaration of the average induced CO2 emissions similar to the Carbon Footprint Label of UK Carbon Trust[8]: They differ between the Reduction CO2 Label communicating the measurement of the carbon footprint and the commitment to reduce it on the one hand and the CO2 Measured Label communicating just the first aspect. The consideration can also be expanded by all of the greenhouse gases resulting in a GHG footprint. Above, the footprint of

[4] http://www.green-software-engineering.de/en/software/powerindicator.html
[5] https://www.blauer-engel.de/en/our-label-environment
[6] http://www.energystar.gov/
[7] http://ec.europa.eu/environment/ecolabel/
[8] http://www.carbontrust.com/client-services/footprinting/footprint-certification/carbon-footprint-label

the whole software development project or the footprint of the software product itself can be calculated. This form of representation might be a way to achieve transparency and awareness for sustainable software products, but it covers only one specific aspect of sustainability.

5. Declaration of the average induced energy consumption similar to the EnerGuide Label[9] of the Government of Canada: Instead of presenting the CO2 emissions it labels and rates the energy consumption or energy efficiency of specific products. Indeed, this does not cover the whole aspects of sustainability either.
6. Combination of different forms of representation similar to the EU Energy Label[10]: It is a uniform label in all EU28 members that states and presents a rating of the energy efficiency of the product by classes mapping to colored arrows. Additionally, the energy class, the supplier's name or trademark and model identifier as well as the annual energy consumption are given. Pictograms highlight specific characteristics depending on the product that is labelled.

3.4 Target Groups

In order to analyze potentials for labelling sustainable software products, one has also to look after the target groups. Who can benefit from a label?

- *Developers* are interested in labels for libraries and tools like integrated development environments. They are also general interested in the criteria in order to consider them during the development.
- *Administrators* are interested in labels, which help them to distinguish between different software products with similar functionality. They also want to know differences, e.g. in energy efficiency of frameworks, runtime environments, and operating systems.
- *End Users* normally use a software product only on one computer / tablet / smartphone. They are interested in labels, which compare standard software like browsers, office software, etc. A label for custom software is not so relevant. They also might be interested in the question, how "green" a service or a website is.
- *Ecological activists* are interested in criteria for Green ICT and how a label might improve sustainable development.
- *ICT companies* are also interested in the criteria and want to know, how they can improve their products. They also want to increase their sales by labelling software.

These ideas for target groups can be interpreted as small starting points for a broad target group analysis being implemented after the general set-up for the label is available.

[9] http://www.nrcan.gc.ca/energy/products/energuide/12523

[10] http://ec.europa.eu/energy/en/topics/energy-efficiency/energy-efficient-products

3.5 Stakeholders

Whereas the target groups comprise those who should be addressed in order to arouse their interests for the qualification, stakeholders are the ones without whose support the implementation of a qualification for sustainable software products would be pretty much impossible. Hence, both groups may overlap but follow different interests. Obviously, the individual persons in both groups are diverse by motivation, interests, personality, and initiative. Whereas the stakeholders are already interested in the specific field in general, the awareness of target groups for appropriate activities still needs to be created in most of the cases. However, it is important to consider stakeholders as well as target groups in developing a label for sustainable software products. The first one especially in order to push the development and the distribution of the label (and all the processes in there), the second one especially to bring awareness to it.

Regarding the identification of stakeholders Penzenstadtler et al. [Penzenstadler et al. 2013] present four approaches of identifying stakeholders for sustainability in software engineering. Herzog et al. [Herzog et al. 2015] identified actors involved in Green IT innovations as they state that "actors can affect the whole life cycle of IT" [Herzog et al. 2015]. In order to find out the stakeholders in our context (labelling sustainable software products) both findings can be combined: the generic list of sustainability stakeholders by Penzenstadtler et al. [Penzenstadler et al. 2013] is based on five different dimensions of sustainability (individual, social, environmental, economic, and technical), whereas the approach by Herzog et al. only goes into the three pillar theory of sustainability as we did. Hence the list can be adapted and mapped to the development of a certification program for software. The models of innovation presented by Herzog et al. should be integrated as well. Thus, ideas how to move on and push the labelling activities forward might be a result. Another important aspect is that the identification of stakeholders needs to be evaluated by mapping concrete case studies to the theoretical findings in order to make the consideration as complete as possible.

Just to come up with some ideas and clarify the stakeholder aspect in the context of labelling sustainable software product: First of all, we see the end user as the one who will gain information about sustainable software and its dimensions by presenting those aspects by a certification. Above, the company who is developing sustainable software product can make a name for herself being sustainable and can grant relief to their Corporate Social Responsibility. Indeed there are also those who are part of the software and sustainability activities and interpret a sustainability label as possibility to support and bring awareness to these activities.

Overall we go with the final statement by Penzenstadtler et al. [Penzenstadler et al. 2013]: "We are positive that successfully identifying the stakeholders for sustainability will help ensure that this objective receives the deserved attention." In our opinion this also holds true for the labelling process.

4 Consideration on Sustainable Websites

After considering the comprehensive field of software, we will focus on websites in the following Sections 4 and 5 in order to reduce the complexity and present how the first step of breaking down the reference criteria to more concrete ones. Since web engineering can be seen as one specific part of software engineering, we will first reduce the domain by looking at websites instead of software products in general in the following section. The break down is not unmitigated and does not present a final solution.

The basis of a sustainable website (product) is web engineering (process of developing the product) that pays attention to sustainability aspects. Hence, before going into details, the question if the website itself or the whole web engineering process including the product website should be labelled needs to be clarified. In a first step of labelling sustainable websites, we will focus on the product. Here, it is possible to evaluate the website as a black box. The next step could be a comprehensive evaluation of the whole process including aspects like the energy consumption of the infrastructure used for developing the website and business trips, belonging to the development of a website and commuters of the development team.

Following the structure of Section 3, we will first present a definition for sustainable websites, followed by a suggestion for corresponding criteria and possible forms of representation.

4.1 Definition

A definition for sustainable websites should meet the identified aspects of sustainable software discussed in Section 2: environmental and resource protection as well as sustainable developments. Hence, the following definition based on [Dick and Naumann 2010] (see Section 5) is a first approach: *A **sustainable website** is a website that is optimized in terms of a sustainable development over its whole life cycle. Negative direct and indirect impacts on sustainable development are reduced, positive impacts on sustainable development are supported. The impacts should be monitored over all life cycle phases.*

The definition is a first step to create transparency by being a basis to build on and it might help to create awareness of the different stakeholders including administrators, designers, developers, authors, and end users [Dick et al. 2010]. All of them are an issue in the context of web engineering and should be addressed.

4.2 Criteria

Criteria on how to characterize a sustainable website help to label a sustainable website and to realize the definition presented above. In order to get a first impression of how one can decide if a website is sustainable, we sum up some criteria inspired by the criteria collection presented in Figure 1 and the

cited literature. It is a first approach and needs to be further refined. The criteria for sustainable websites can be the basis for labelling websites. Overall, the different aspects can be classified as ecological, economic, and social criteria, based on the three pillars of sustainability. Some of the characteristics belong to more than one sustainability aspect. In order to find a connection to the criteria for sustainable software products and their engineering, we will match the criteria to Efficiency, Feasibility, and Perdurability.

The following set of criteria helps to evaluate the degree of fulfillment regarding the Efficiency, Feasibility, and Perdurability of a website. The following list is by no means complete and not intended to be so.

Efficiency "defines how software behaves when it comes to saving resources" [Taina 2011]. In our opinion this is applicable to websites. The following criteria specify and evaluate the Efficiency aspect in the context of sustainable websites.

- *File Sizes*: The file sizes of the website content should be kept to a minimum, e.g. by compressing source code and minimizing images. Recommendations for actions can be found in [Dick et al. 2010].
- *Number of Files*: The number of files should be kept to a minimum, e.g. by combining separated files.
- *Data Transfer*: The induced data transfer of the website should be kept to a minimum, e.g. by using caching. Recommendations for actions can be found in [Dick et al. 2010].

Feasibility considers how aspects of a sustainable development are supported [Taina 2011]. As presented in Section 2 we differentiate between resource-oriented and well-being oriented criteria. Resource-oriented Feasibility can be specified and evaluated by the following criteria:

- *Energy Type*: The server hosting a sustainable website should be operated by renewable energy.
- *Energy Consumption*: The energy consumption should be continuously monitored and optimized during the life cycle of a website.
- *Energy Management Options*: The possibility to deactivate energy intensive content like animations should be given to the user. The user should be able to get the whole content of the website also without using the energy intensive functionalities.
- *Carbon Footprint*: The carbon footprint specifies the amount of CO2 emissions caused by the website during its life cycle. It should be calculated and analyzed regularly in order to continuously reduce the induced emissions.

Well-being oriented Feasibility can be specified and evaluated by the following criteria:

- *Sustainability Support*: Sustainability support especially concerns the content of a website. The support is given if the website presents for example a community for sustainability relevant topics or a knowledge base with articles on how to save energy and resources etc. This criterion mainly goes for sustainability by websites.
- *Accessibility*: The website should meet the user's needs to give accessibility to as many users as possible. For example colors as well as font sizes should be adaptable. In this context, Web Content Accessibility Guidelines are developed through the W3C process [Web Accessibility Initiative (WAI) 2008].
- *Usability*: Features that enable a website to be user friendly should be integrated. Usability guidelines list specific criteria to check the usability of a website, e.g. the guidelines published by The Website Standard Association [Website Standards Association 2008]. In order to keep the usability of a website on a high level usability tests should be applied regularly.

Perdurability can be defined as "the degree to which a software product can be modified, adapted and reused in order to perform specified functions under specified conditions for a long period of time." [Calero et al. 2013b] Regarding websites the following criteria specify and evaluate the Perdurability aspect:

- *Maintainability*: The maintainability of a website should be kept as high as possible. That means that it is possible to analyze and change the files in an effective and efficient way. Here, having a well-structured and thorough documentation of the website makes sense. Different kinds of documentation should be created: guidelines for administrators and guidelines for authors. The guidelines can help to maintain the site in a sustainable way regarding the source code as well as the content.
- *Adaptability*: Adaptability belongs to aspects regarding user experience (1) as well as technical aspects (2). (1) In times of a huge amount of devices with various screen sizes, qualities, and usage scenarios, the website should be highly adaptable to meet the user's expectation. For example, the site should adapt itself spontaneously if the screen changes from portrait to landscape mode or from small screens to bigger ones. (2) Next to direct user interactions as social aspect technical configurations influence the environmental part. Here, e.g. reduced bandwidth or higher energy consumption of cellular networks for data transfer are issues. For example, images of the mobile version of a website can be smaller by default.
- *Reusability*: Reusability considers the aspect of using existing methods and code fragments of websites for different development projects. This gains importance both intern in a company and for cross-cutting projects in the World Wide Web, in other words it is more efficient to use the same methods for different websites as to create or rather load it again for each case.

Overall, there needs to be more research regarding social aspects of websites, web engineering or rather software engineering in general [Johann and Maalej 2013]. Johann et al. focus on equal

opportunities and participation as social aspects. Economic aspects are even more important while including the whole web engineering process, the project, the organization developing the website, etc.

As a next step the criteria need to be evaluated from a practical view, means reviewing if and how a website is checkable by this criteria and how a corresponding checklist as well as recommendations for actions can look like. This will be the basis for award guidelines for a label.

4.3 Form of Representation

Nowadays, the term "sustainability" is used in different ways depending on the specific context. Hence, the complexity how to understand the term the right way is high whereas the traceability is not given for most of the stakeholders. In order to react to this development the information about sustainable software needs to be clear. Thus, a label presenting this specific characterization of a software product should be expressive. Here, a label like the Blue Angel presenting the protective goal could be a good solution.

Thinking of a label for sustainable software there might be differentiated label versions informing about the sustainable aspect (social sustainability, ecological sustainability, and economic sustainability) that should be highlighted and is fulfilled most. In that way the user gets to know why and how the specific product is sustainable. Another possibility is to use this statement to figure out if the software product itself is sustainable (sustainable in software) or if it supports sustainable development (sustainable by software).

Above, a label with gradations seems to make sense for sustainable software products. So far, the development of sustainable software is just in the beginning and the different criteria might not be fulfilled completely. Different quality classes provide the opportunity to label a product that is sustainable in some aspects. Optimizations of the product arise as a result of progressing development of sustainable software engineering principles.

5 Focusing on Green Websites

Next to concentrating on a specific part of software products (here: websites), the complexity of creating a label for sustainable software products is additionally reduced by focusing on a specific sustainability aspect. Hence, we will have a more detailed look on possibilities of labelling green websites, meaning we will dwell on the ecological sustainability instead of including social and economic aspects additionally. The following considerations are based on our previous works. Focusing on websites corresponds to the results of Hilty et al. [Hilty et al. 2015]. They point out that traditional websites or web-based applications could be suitable for labelling. In our paper, we will predominantly consider websites providing information in a static way and using standard web

technologies like HTML, CSS, etc. Criteria for green web content management systems and web applications can be found in [Dick et al. 2012].

Corresponding to the general consideration of aspects in the context of labelling sustainable software products in Section 3 and of the mapping to sustainable websites in Section 4, the following consideration will also dwell on a definition, criteria, and possible forms of representation.

5.1 Criteria

The complexity of finding criteria for a green website based on the presented definition with focus on the energy consumption is less than for sustainable websites. Hence, based on previously published results, we propose the following criteria for green websites:

- *Optimization of Content and Data Transfer*: Applying green web engineering principles, like using caching, can have a positive impact on the energy consumption of the website [Dick et al. 2012]. It is important to continuously monitor the resulting effects of applying these principles to the induced energy consumption.
- *Optimization of Files*: The criteria Number of Files as well as the File Size are important and influence the resulting energy consumption while using a website. We proofed this by different energy measurements [Dick et al. 2012]. Both, file number and file size, should be kept to a minimum. This minimum cannot be exactly defined and depends on the specific website. The files should be as much compressed as possible as far as the functionality is not reduced.
- *Optimization of Energy Aspects*: Overall, energy is an interesting aspect seen from the view of ecology. On the one side, it is interesting which kind of energy is used to operate the server hosting the website (renewable energy or not). On the other side, the energy consumption should be continuously monitored and optimized during the life cycle of a website. Here, overlaps with the criteria Optimization of Content and Data Transfer exist. Additionally, there should be energy management options. A possible energy option for a website can be the deactivation of energy consuming content like animations without minimizing the presented information. Web advertising also looms large regarding the energy consumption while surfing a website [Simons and Pras 2010].

The proposed criteria belong to the website itself, meaning the product and not the whole engineering process and sum up some of the criteria listed separately in Section 4. Similarly to the discussion of sustainable websites, our first step of labelling green websites is to focus on the product. Afterwards, if implementing green web engineering principles and corresponding tools are established, the criteria could be extended to the whole process. For example that means that there need to be energy measurements of the website but no monitoring of the energy consumption of the whole project.

5.2 Form of Representation

Since, in case of our definition, the focus of a green website is clearly set on energy consumption the label should also concentrate on this focus. Thus, we advance a neutral label similar to the Energy Star or a declaration of the average induced energy consumption.

Indeed, to be able to realize a label presenting energy values based on measurements usage scenarios are necessary. So far, there are different approaches for energy measurements in the context of software [Amsel and Tomlinson 2010; Capra et al. 2012; Kern et al. 2013] but a standardized method to measure the energy consumption of websites is missing. In general, the information on the label is to be kept up to date. Caused by the already mentioned fast-moving nature of websites that leads to changing structure and content of the site it might be necessary to update the average resulting energy consumption more often than in case of other software products.

Hence, a neutral label just informing about the green characterization of the website might be the first step towards more transparency in the context of the energy consumption of the Internet. In order to create a recognition value we recommend to identify an existing label and concrete the presented criteria to its standards. However, to do so commonly accepted criteria for green websites are necessary. The criteria presented above might represent a starting point to reach a common acceptance.

6 Discussion

As presented in our paper the core issue what exactly is meant by the term “sustainable software” is not answered yet but the different authors agree on the relevance of bringing sustainability to the ICT field and emphasize the impacts of software on the environment. Many contributions discuss this topic. We referred to a variety of them in order to find a common basis. The labelling process is complicated by different understandings and foci on sustainable software. Nevertheless, by presenting ideas and approaches on the way of labelling sustainable software products we wanted to demonstrate that the potential in this context is high – on condition that basic issues will be solved. The efforts of creating a label can be understood as challenges to solve the still arising questions.

Hence, from our point of view the following questions gain a high importance: What are suitable criteria a label can be based on? How to represent these criteria in a reasonable way? Both, criteria and form of representation, do finally depend on the expectation of the label. In our paper we presented possibilities to find a solution for these questions. Indeed, the concrete aspiration of a label has still to be defined: Is the software product itself considered? Is the software development process considered? Is the whole life cycle of the software considered? Is the software company considered?

Regarding green websites we proposed to concentrate on the three aspects Optimization of Content and Data Transfer, Optimization of Files or Optimization of Energy Aspects (see Section 3). Here, the limitation is to concentrate on the environmental aspects or more specifically the energy consumption and considering the website itself, not the whole process behind.

Indeed, these limitations are not yet defined for sustainable software. However, apart from defining the general set-up for a label and breaking the criteria to reach an awarding guideline, the criteria itself are proposals so far. To find generally accepted criteria, they need to be discussed and weighed. Exemplarily, we will discuss the aspects “rebound and efficiency” and “carbon footprint vs. privacy” in the following paragraphs.

Additionally to the discussion of the criteria itself, further points of discussions will be mentioned in the outlook (Section 7). They are supplements to the addressed aspects (definition, criteria, form of representation, target groups, and stakeholders).

6.1 Rebound and Efficiency

Many of the criteria presented above are more or less about efficiency. Single aspects of the criteria are either minimized or maximized depending on the nature of the criteria. This is not only true for the *Efficiency* section, but also for the *Feasibility* and *Perdurability* sections, considering the properties *Energy Consumption* (see *Feasibility*) or *Maintainability* (see *Perdurability*) for example. Increasing the efficiency regarding those properties does not necessarily mean that a website in total causes less negative impacts, e.g. carbon dioxide emissions due to server operation, as it would have caused without any efficiency improvements.

To reduce complexity, assume that the number of servers hosting a website is fixed. If there are efficiency improvements to the website, e.g. less data transfer and thus necessary server performance, it is possible to serve more visitors with the now available performance. As a result, in total, there are no savings at all. The emissions remain the same as without any efficiency gains.

The same can be observed with efficiency gains regarding maintainability. Assume a maintenance department with a fixed number of employees who are responsible of maintaining a set of websites. If the maintainability of these websites is improved, the maintenance people can handle more websites in the same amount of time, but in total, the emissions caused by the required office space and infrastructure remain the same. Even if they do not handle more websites, there will be no savings as long as we do not fire any employees and release the then disposable office space (which will indeed have a negative social impact on the affected employees).

Naturally one may argue that without efficiency gains one would have required more servers to serve the same amount of visitors or that one would have required more maintainers to maintain the same amount of websites, but just with efficiency gains without a total reduction, it seems to be

impossible to contribute to the EU countries' goal announced at the UN 2014 Climate Change Summit of cutting carbon dioxide emissions to 40 per cent below 1990 levels by 2030 .

One may ask, why our criteria are then sticking on efficiency properties to label a sustainable website or rather software. As others found and discussed before us, efficiency gains are necessary but not sufficient to achieve sustainability goals [Hilty 2012; Gossart 2015]. The effects discussed above are also known as rebound effects. A strong rebound effect may not only compensate, but also even overcompensate the initial energy savings achieved by gains in efficiency. Hence, to face rebound effects, it is not sufficient to simply consider efficiency properties. It is rather additionally necessary to keep an eye on the total savings.

6.2 Carbon Footprint vs. Privacy

Conducting a Carbon Footprint study is nothing new. Even for websites, it is possible to buy a service that calculates an ongoing Carbon Footprint calculation, called co2stats[11]. Despite the fact that this service is online since 2008 [TechCrunch Network 2008], nearly no information except the FAQ on their own website10 is publicly available, that describes how it exactly works. According to the FAQ, they collect data about the website usage, network traffic, as well as client computers, client location (to get an approximate of the energy mix) and used browsers of the visitors of a certified website. The Carbon Footprint is calculated continuously on the basis of the collected data. They claim to buy enough renewable energy certificates with your service-subscription to green the whole website, including approximated energy for the network traffic, as well as for the clients accessing your site. We do not want to discuss whether the co2stats method is suitable to green a website or not. Obviously, you need to collect huge amounts of data about your visitors (type of computer, browser, location, used network connections, accessed pages, usage time, etc.) to get an accurate carbon footprint estimation. All these data has to be stored and processed, which may introduce privacy risks, if the collected data is not anonymized instantaneously. However, according to EU privacy regulations (directive 2009/136/EC European Parliament 2009), it is generally necessary to obtain visitor's consent before installing a cookie or some other technology necessary for tracking the actions of a visitor on a website. Whether or not such privacy risks may be acceptable for a sustainable website has to be further discussed.

7 Conclusion and Outlook

In our paper we summed up the aspects that are relevant while creating a label for sustainable software products: definition, criteria, form of representation, target groups, and stakeholders. In order to find a

[11] http://www.co2stats.com/faq.php

common basis we compared existing approaches and extended them by own ideas. At first, we considered software products in general and focused on websites afterwards. Here, we proposed especially criteria and suitable representations for a sustainability label or rather an eco-label. The presented approaches need to be evaluated in corresponding case studies and duly appropriated. The discussion section highlights open questions and challenges. Overall, based on the previous sections and the literature cited we came to the following findings:

1. In order to move forward in creating a sustainable software label it is necessary to define a general set-up including: aspiration, object of investigation, considered sustainability aspect, focus of by- or in-aspects.
2. It is possible to combine existing approaches of definitions and criteria for sustainable software products and its engineering. A selection of these depends on the result of defining the set-up for a sustainability label for software.
3. A differentiated label presenting details about the awarding basis and quality meets the expectations of the end users in the best way and should be a long-term result. A bold and simple label might be a first step and a starting point for advancements.

Based on these findings the next steps in labelling sustainable software products should include the following activities:

1. The demand and problems of a sustainability label for software products and its engineering need be discussed in general.
2. The criteria for sustainable software products and projects need to be evaluated from a practical proof and broke down to checklists for developers and award guidelines for certification offices.
3. The proposed criteria should be further proofed in perspective of quality criteria for software products and websites. So far they are primarily just laid on findings of sustainability informatics results.
4. Possibilities for certification bodies (Who is awarding the software products?) need to be collected, compared, and evaluated.

Overall, the listings of aspects regarding labelling sustainable software products need to be further developed.

References

ABDULLAH, R., ABDULLAH, S., DIN, J., TEE, M., AND OTHERS. 2015. A Systematic Literature Review of Green Software Development in Collaborative Knowledge Management Environment. *International Journal of Advanced Computer Technology (IJACT) 9*, 136.

AGARWAL, S., NATH, A., AND CHOWDHURY, D. 2012. Sustainable Approaches and Good Practices in Green Software Engineering. *IJRRCS 3*, 1, 1425–1428. http://scholarlyexchange.org/ojs/index.php/IJRRCS/article/view/9903/7030.

ALBERTAO, F., XIAO, J., TIAN, C., LU, Y., ZHANG, K.Q., AND LIU, C. 2010. Measuring the Sustainability Performance of Software Projects. In *2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China*, IEEE Computer Society, Ed., 369–373.

AMSEL, N., AND TOMLINSON, B. 2010. Green Tracker: a tool for estimating the energy consumption of software. In *CHI EA '10: Proceedings of the 28th international conference extended abstracts on Human factors in computing systems*, Association for Computing Machinery, Ed. ACM, New York, 3337–3342.

BETZ, S., AND CAPORALE, T. 2014. Sustainable Software System Engineering. In *Big Data and Cloud Computing (BdCloud), 2014 IEEE Fourth International Conference on*, 612–619.

CALERO, C., BERTOA, M.F., AND MORAGA, M.Á. 2013a. Sustainability and Quality: Icing on the Cake. In *RE4SuSy@RE*.

CALERO, C., MORAGA, M., AND BERTOA, M.F. 2013b. Towards a software product sustainability model. *arXiv preprint arXiv:1309.1640*.

CAPRA, E., FORMENTI, G., FRANCALANCI, C., AND GALLAZZI, S. 2010. *The Impact of MIS Software on IT Energy Consumption*. 18th European Conference on Information Systems, 7-9 June 2010, Pretoria, South Africa. http://web.up.ac.za/ecis/ECIS2010PR/ECIS2010/Content/Papers/0073.R1.pdf. Accessed 25 October 2010.

CAPRA, E., FRANCALANCI, C., AND SLAUGHTER, S.A. 2011. Is software green? Application development environments and energy efficiency in open source applications. *Information and Software Technology 54*, 60–71. http://ac.els-cdn.com/S0950584911001777/1-s2.0-S0950584911001777-main.pdf?_tid=a288bc86-f725-11e1-960e-00000aacb362&acdnat=1346827861_377674c8edd54a640c2d57df2bd4951b.

CAPRA, E., FRANCALANCI, C., AND SLAUGHTER, S.A. 2012. Measuring Application Software Energy Efficiency. *IT Professional*, 54–61.

Climate Change Summary - Chair's Summary - UN Climate Summit 2014. http://www.un.org/climatechange/summit/2014/09/2014-climate-change-summary-chairs-summary/. Accessed 15 March 2015.

DICK, M., KERN, E., JOHANN, T., NAUMANN, S., AND GÜLDEN, C. 2012. Green Web Engineering - Measurements and Findings. In *EnviroInfo 2012: Man Environment Bauhaus: Light up the Ideas of Environmental Informatics. Proceedings of the 26th International Conference on Informatics on Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management; Federal Environment Agency, Dessau, 2012*, H.-K. ARNDT, G. KNETSCH AND W. PILLMANN, Eds. Shaker Verlag, Aachen, 599–606.

DICK, M., AND NAUMANN, S. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, October 6 - 8, 2010, Cologne/Bonn, Germany*, K. GREVE AND A. B. CREMERS, Eds. Shaker, Aachen, 706–715.

DICK, M., NAUMANN, S., AND HELD, A. 2010. Green Web Engineering. A Set of Principles to Support the Development and Operation of "Green" Websites and their Utilization during a Website's Life Cycle. In *WEBIST 2010 - Proceedings of the Sixth International Conference on Web Information Systems and Technologies, Volume 1, Valencia, Spain, April 07-10, 2010*, J. FILIPE AND J. CORDEIRO, Eds. INSTICC Press, Setúbal, 48–55.

EUROPEAN PARLIAMENT. 2009. *Directive 2009/136/EC of the European Parliament and of the Council of 25 November 2009 amending Directive 2002/22/EC on universal service and users' rights relating to electronic communications networks and services, Directive 2002/58/EC concerning the processing of personal data and the protection of privacy in the electronic communications sector and Regulation (EC) No 2006/2004 on*

cooperation between national authorities responsible for the enforcement of consumer protection laws, 2009/136/EC. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:en:PDF.

GeSI, Global e-Sustainability Initiative; wbcsd; World Resource Institute; Carbon Trust. 2013. *GHG Protocol Product Life Cycle Accounting and Reporting Standard ICT Sector Guidance. Chapter 7, Software*. Draft v2.9.

GOSSART, C. 2015. Rebound effects and ICT: a review of the literature. In *ICT Innovations for Sustainability*. *Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 435–448.

HERZOG, C., LEFÉVRE, L., AND PIERSON, J.-M. 2015. Actors for Innovation in Green IT. In *ICT Innovations for Sustainability*. *Advances in Intelligent Systems and Computing*, L. M. HILTY AND B. AEBISCHER, Eds. Springer, Switzerland, 49–67.

HILTY, L., LOHMANN, W., BEHRENDT, S., EVERS-WÖLK, M., FICHTER, K., AND HINTEMANN, R. 2015. Green Software. Final report of the project: Establishing and exploiting potentials for environmental protection in information and communication technology (Green IT). Report commissioned by the Federal Environment Agency, Berlin, Förderkennzeichen 3710 95 302/3, 23.

HILTY, L.M. 2012. Why energy efficiency is not sufficient-some remarks on "Green by IT". In *EnviroInfo 2012, Proceedings of the 26th Environmental Informatics Conference, Shaker, Aachen*, 13–20.

JOHANN, T., AND MAALEJ, W. 2013. Position Paper: The Social Dimension of Sustainability in Requirements Engineering. In *Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems*.

KERN, E., DICK, M., NAUMANN, S., GULDNER, A., AND JOHANN, T. 2013. Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In *ICT4S ICT for Sustainability*. *Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 87–94.

LAMI, G., FABBRINI, F., AND FUSANI MARIO. 2012. Software Sustainability from a Process-Centric Perspective. In *EuroSPI 2012, CCIS 301*, D. Winkler, R.V O'Connor and R. Messnarz, Eds. Springer, 97–108.

MAHMOUD, S.S., AND AHMAD, I. 2013. A Green Model for Sustainable Software Engineering 2013. *International Journal of Software Engineering and Its Applications 7*, 4, 55–74. http://www.sersc.org/journals/IJSEIA/vol7_no4_2013/5.pdf.

MARCU, M., AND TUDOR, D. 2011. Power consumption measurements of virtual machines. In *6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, IEEE, Ed., 445–449.

NAUMANN, S., DICK, M., KERN, E., AND JOHANN, T. 2011. The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. *SUSCOM 1*, 4, 294–304. doi:10.1016/j.suscom.2011.06.004.

NAUMANN, S., GRESK, S., AND SCHÄFER, K. 2008. How Green is the Web? Visualizing the Power Quality of Websites. In *EnviroInfo 2008: Environmental Informatics and Industrial Ecology*. *proceedings of the 22nd International Conference Environmental Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management, September 10 - 12, 2008, Leuphana University Lueneburg, Germany*, A. MÖLLER, B. PAGE AND M. SCHREIBER, Eds. Shaker, Aachen, 62–65.

NOUREDDINE, A., BOURDON, A., ROUVOY, R., AND SEINTURIER, L. 2012. A preliminary study of the impact of software engineering on greenit. In *Green and Sustainable Software (GREENS), 2012 First International Workshop on*, 21–27.

PENZENSTADLER, B. 2013. Towards a Definition of Sustainability in and for Software Engineering. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 1183–1185.

PENZENSTADLER, B., FEMMER, H., AND RICHARDSON, D. 2013. Who is the advocate? stakeholders for sustainability. In *Green and Sustainable Software (GREENS)*. *2nd International Workshop on Green and Sustainable Software*, 70–77.

PENZENSTADLER, B., RATURI, A., RICHARDSON, D., CALERO, C., FEMMER, H., AND FRANCH, X. 2014. *Systematic Mapping Study on Software Engineering for Sustainability (SE4S) - Protocol and Results* ICS2 221. Institute for Software Research, University of California, Irvine, Irvine.

SAHIN, C., CAYCI, F., MANOTAS GUTIERREZ, I.L., CLAUSE, J., KIAMILEV, F., POLLOCK, L., AND WINBLADH, K. 2012. Initial Explorations on Design Pattern Energy Usage. In *Proceedings of the First International Workshop on Green and Sustainable Software (GREENS) 2012. held in conjunction with ICSE 2012, The International Conference on Software Engineering, June 2-9, Zurich, Switzerland*, IEEE, Ed. IEEE Computer Society, 55–61.

SCHMIDT, B., WYTZISK, A., PLÖDEREDER, I.E., GRUNSKE, L., SCHNEIDER, E., ULL, D., AND OTHERS. 2014. Software Engineering und Integrative Nachhaltigkeit. In *INFORMATIK 2014. Big Data – Komplexität meistern. – Proceedings. 2. Workshop „Umweltinformatik zwischen Nachhaltigkeit und Wandel (UINW) / Environmental Informatics between Sustainability and Change"), 26.09.2014, im Rahmen der INFORMATIK 2014, 22.-26.09.2014 in Stuttgart*, E. PLÖDEREDER, L. GRUNSKE, E. SCHNEIDER AND D. ULL, Eds. Lecture Notes in Informatics (LNI), 1935–1945.

SIMONS, R.J., AND PRAS, A. 2010. The Hidden Energy Cost of Web Advertising.

TAINA, J. 2011. Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. In *Green ICT: Trends and Challenges*, J.-C. LOPEZ-LOPEZ, G. SISSA AND L. NATVIG, Eds., 22–27.

TECHCRUNCH NETWORK. 2008. *CO2Stats Compensates For Your Site's Pollution*. http://techcrunch.com/2008/08/14/co2stats-compensates-for-your-sites-pollution/. Accessed 15 March 2015.

WEB ACCESSIBILITY INITIATIVE (WAI). 2008. *Web Content Accessibility Guidelines (WCAG) 2.0*. http://www.w3.org/TR/WCAG20/. Accessed 19 January 2014.

WEBSITE STANDARDS ASSOCIATION. 2008. *Business Website Usability Guidelines*. http://www.websitestandards.org/Standards_download.html.

WILKE, C., RICHLY, S., PÜSCHEL, G., PIECHNICK, C., GÖTZ, S., AND AßMANN, U. 2012. Energy Labels for Mobile Applications. In *GI-Jahrestagung*, 412–426.

WILLNECKER, F., BRUNNERT, A., AND KRCMAR, H. 2014. Model-based Energy Consumption Prediction for Mobile Applications. In *Proceedings of Workshop on Energy Aware Software Development (EASED)@ EnviroInfo*, 747–752.

ZHANG, C., HINDLE, A., AND GERMAN, D.M. 2014. The Impact of User Choice on Energy Consumption. *Software, IEEE 31*, 3, 69–75.

Green computing, green software, and its characteristics: Awareness, rating, challenges

Eva Kern[a,b*]

[a] Leuphana University Lueneburg, Scharnhorststrasse 1, 21335 Lueneburg, Germany

[b] Institute for Software Systems in Business, Environment, and Administration, Trier University of Applied Sciences, Environmental Campus Birkenfeld, P.O. Box 1380, 55761 Birkenfeld, Germany

Abstract. Issues of green computing, especially on the hardware side, arouse interest of scientific communities since several years. The number of scientific publications and research activities rises. This begs the question if resulting ideas, approaches, and findings find their way from science to society. Hence, similar to different researchers, we conducted a survey regarding corresponding issues. Since we focused on software users, we will compare our results to those addressing students, practitioners, and IT managers. The following paper will provide an insight into awareness of green software and green computing, present a user survey we conducted in the summer of 2016 in Germany, and approaches on how to create awareness to environmental issues of ICT, especially on the software side.

Keywords. Green Computing, Eco-labelling, Awareness, Green Software

1 Introduction

Information and communication technologies (ICT) involve with a negative and a positive side: On the one hand, they have a significant impact on the worldwide energy consumption and global CO2 emissions [Malmodin et al. 2013; Andrae and Edler 2015]. On the other hand, according to the #SMARTer2030 report [2015], "ICT can enable a 20% reduction of global CO2e emissions by 2030, holding emissions at 2015 levels". Thus, it is "both a solution and a problem for environmental sustainability" [Selyamani and Ahmad 2015]. In order to strengthen the positive role of ICT, the Global e-Sustainability Initiative sees especially three important stakeholders: policymakers, business leaders, and consumers. [GeSI, Global e-Sustainability Initiative 2015]

Overall, environmental issues of software comprise not only energy issues but also aspects of resource consumption (including e.g. hardware needed to operate a software product, adaptability of the hardware in use, and the way of delivery of the software product), issues regarding the potential

hardware operating life (like backward compatibility and platform independency), and characteristics of the software supporting user autonomy to enable a more environmental friendly way of using the products. In [Kern et al. 2015] we identified possible criteria for green software.

We see the creation of awareness for environmental issues of ICT and especially software as the first step in moving towards a more environmental friendly way of using ICT. We agree with Ramachandiran [2012] in regards of awareness: "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". In the context of green computing, Lago and Jansen [2010] differ between three kinds of awareness: "the direct environmental impact of executing software services (service awareness); the indirect environmental impact of their development process (process awareness); and the use of services to monitor and measure the two above, so to create people awareness).". The following paper comprises all of them when talking about awareness. The focus is set on the view of end users (being consumers) and practitioners (being developers, managers etc.) to show possibilities "from science to society".

Thus, the following paper addresses the two questions: (i) *Is there an awareness towards green computing and especially green software?* (ii) *How can the awareness towards these topics be raised?*

2 Awareness on green computing and green software

While there are more and more scientific conferences, workshops, and journals addressing Green IT topics and integrate increasingly also green software aspects, the issues still gain little to no awareness by non-academics. Recent research activities "mainly focused on solutions for the IT industry and business, and neglected the end users" [Selyamani and Ahmad 2015]. The surveys we present in the following section prove these impressions and theses.

Talking about "Green Computing" or rather "Green ICT" mainly sets the focus on hardware issues while software issues are directly named as "green software" or similar terms. Here, both aspects are included: the specific focus of the study in question is mentioned in the second column of the table (*italicized*).

Table 1 *Overview of studies addressing the awareness of green computing issues*

Target group	**Objective & focus**	**Findings**	**Source**
Programmers	Do programmers know about energy consumption of *software* and how to reduce it although such issues are not addressed in education or trainings?	"Programmers had limited knowledge of energy efficiency, lacked knowledge of the best practices to reduce energy consumption and were unsure about how software consumes energy"	[Pang et al. 2016]
Practitioners	Relationship between	"Practitioners care about	[Manotas et

	research community and practitioners: same interests? Asking for the practitioners' perspective to get a motivation and guide for green *software* engineering research	energy but are not as effective as they could be at creating efficient applications."	al. 2016]
Students in Mauritius	Analyze the awareness of green computing (focus: *hardware*) among students	"Students have a moderate knowledge about green computing but their everyday green computing practices are not satisfactory"	[Dookhitram et al. 2012]
Students in Malaysia	*Hardware* focus: usage and selling of computers and related resources	"Students have an average level awareness about green computing and their everyday practices are not satisfactory"	[Selyamani and Ahmad 2015]
Buyer and supplier	Analyze people's awareness of *software* engineering issues	"People are aware of the importance of energy efficient software, but lack information and resources to engage this topic."	[Ferreira 2011]
ICT Managers	How do ICT managers implement energy efficient methods (focusing on *hardware* issues) in their organizations?	"just over a third of organisations in Europe do not implement green IT practices"	[Kogelman 2011]

Different surveys address the knowledge and awareness of green computing. They aim at target groups like programmers, practitioners, ICT managers, and students (Table 1). Especially the last ones were questioned in different projects. Here, the aim is to improve education and information activities in the context of Green IT [Dookhitram et al. 2012; Selyamani and Ahmad 2015]. After finishing their studies, the former students might bring their knowledge about green computing into companies and organizations. Here, ICT managers are in charge of implementing green strategies. This is why Kogelman [2011] surveyed ICT managers. For Manotas et al. [2016], the motivation for implementing an awareness study among practitioners in general was to gain insights for researching activities regarding green software engineering.

Summarizing the findings of the considered studies, the knowledge about green computing of the interviewed programmers, students, practitioners, buyers, suppliers, and managers are a great deal. Most of them seem to have some knowledge. However, the lack of practice is large. The studies addressing students mainly focus on the hardware aspects and integrate software aspects just slightly (e.g. operating system and search engines [Selyamani and Ahmad 2015]) or concentrate on specific single concerns of environmental issues of software (e.g. energy [Ferreira 2011; Kogelman 2011]). Nevertheless, all of them provide an impression of the awareness of green computing and green software. Like Manotas et al. [2016] describe it, the studies "can motivate and guide green software engineering research".

Indeed, a study addressing the group of software users was, as far as we know, missing. Thus, we conducted a survey on social interests in green software and acceptance factors influencing an eco-label for software products. The study as well as its result is presented in the following section. Above that, we combine our findings with the results of the surveys listed in Table 1.

3 User survey: environmental issues of software

From August, 16 to October, 5 2016 we conducted a user survey addressing environmental issues of software and their certification. The overall aim was to bridge from science to society. This requires the knowledge about society's interests. Thus, the data was collected via an anonymous online questionnaire, addressing German software users by asking the questions in German.

3.1 Objectives and methodology

The objective of the survey was to gather the aspects of green software that arouse social interest and factors that influence the acceptance of a possible certification of green software [Kern 2016]. Thus, the first step is to find out if there is awareness for environmental issues of software. In contrast to the other surveys, here, the focus target groups were end users of software.

The questionnaire was divided into three sections: (1) factors of influence on the acceptance of a certification of green software products, (2) evaluation of possible criteria for an eco-label for software products, and (3) demographic characteristics. A total of 712 questionnaires were completed and used for data analysis. The data was analyzed using both, descriptive methods and correlations between variables.

In the following, the results of section (2) and (3) are described. We start with the demographic data to first give an impression of the survey responders. The outcomes of section (1) are not presented in detail, since the paper focuses on the awareness of and interest in environmental issues of software instead of labeling them.

3.2 Results

Asked for their role in using software, most of the survey respondents (78.2 %) see themselves primarily as software users (possible answers: developer, user, purchaser, distributor, administrator, vendor, IT manager, and others). 55.8 % of them were female, 40.7 % male, and 3.5 % did not answer this question. More than two-thirds are between 20 and 39 years old (39.3 % are 20 to 29 years old, 29.4 % between 30 and 39 years). About half of them have a higher university degree (48.3 %); 52.5 % of the survey participants work in the context of natural sciences, geography and computer sciences.

Most of the respondents know the Blue Angel1 (94.0 %), whereas the other eco-labels that were part of the survey are known by 68.0 % in case of the ENERGY STAR2 and 10.3 % in case of the TCO Certification[3]. Only 4.6 % of the respondents did not know any of the three labels. However, there is quite a lack of knowledge regarding the details of the awarding criteria and awarded products. This could be interpreted as demand to better inform about these aspects and thus about environmental issues. Indeed, according to their answers, the survey participants have a high environmental awareness: 61.4 % of them are willing to pay more for organic products, 23.7 % catch up on environmental topics.

In spite of the environmental interests and knowledge about corresponding labels, more than half of the participants of the survey never paid heed to an eco-label for software products (56.3 % "agree", 18,7 % "somehow agree", 12,6 % "somehow do not agree", 10.4 % "do not agree" to "I never cared if there is a certification of environmental issues of software."). If there are two products and the costumers know just the one that is not labeled as "green", they would buy the product that is not-green but known instead of the green software product. (63.3 % agree). However, 23.2 % of the respondents can imagine taking "certification of environmental issues of the software product" as one possible buying criteria. 51.1 % would do that if there are products with the same functionality, some being "green" and others being "not green".

Comparing the results in respect to the gender, the differences between women and men are nearly the same in case of agreeing positions whereas the differences are bigger in negative responses: 11 % of the men say that they cannot imagine that environmental issues of software can be a buying criterion for software while 2 % of the women object to this idea.

In respect of the role of software usage, the following result occurs: Especially purchasers can imagine taking "rating of environmental issues" as a criterion while searching for software products (Fig. 1). However, the significance of this result is quite disputable, since just 13 participants characterize themselves as being primary a software purchaser. As mentioned, the main focus group of the survey were software users. Thus, they represent the biggest group of survey participants (78.2 %), followed by developers (11.5 %). Between these two groups, the distribution of the answers is quite similar. However, there is a marginal tendency towards "I cannot imagine doing so" in case of developers.

According to the results, there seems to be an interest in environmental topics on the one hand (e.g. knowing eco-labels and taking them into account while buying products). However, on the other

[1] https://www.blauer-engel.de/en (Blauer Engel)

[2] https://www.energystar.gov/

[3] http://tcodevelopment.com/

hand, these interests and awareness do not include environmental issues of software. In order to counteract here, the idea is to develop an eco-label for software. In the context of a project called "Sustainable Software Design"[4], we are currently working on such a certification.

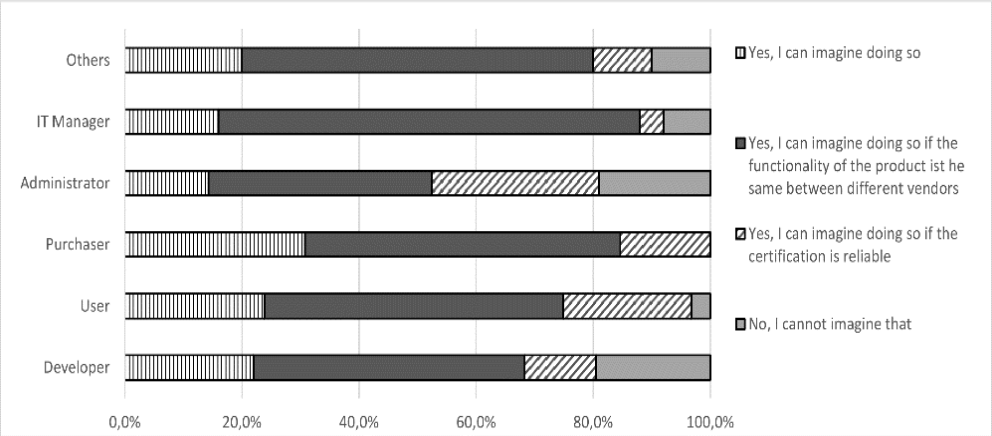

***Figure 1**. Results of the questions regarding a possible selection criterion "environmental issues of software", distinguished by the role in using software, leaving out vendors and distributors due to too little respondents (Question: „Can you imagine to take 'grading of the environmental impact of the product' as a criterion while searching for a software product?")*

The second part of the survey addressed the question if the ideas for criteria developed in the project arouse social interests. The survey participants were asked if the criteria "should be labeled", "should rather be labeled", "should rather not be labeled" or "should not be labeled". Overall, especially the aspects *energy efficiency* (77.5 % said that it should be labeled) and *hardware efficiency* (62.5 %) gain big interests. The idea to label *platform independence* and *backward compatibility* also seems to be attractive, whereas especially the criteria *hardware sufficiency* (35.4 % "should be labeled") and *quality of product information* (31.6 %) are deemed less important to be labeled by the survey participants. Overall, the answers to the question "Which ones of these environmental issues of software should be labeled?" show that there seems to be a general interest for the presented aspects.

3.3 Discussion and comparison

Comparing our survey to the other studies introduced in section 2, we noticed that most of them relate eco-labels to Green IT contexts. The different studies show that many people generally know environmental labels. In our study, only 4.6 % of the respondents did not know any of the three labels (ENERGY STAR, Blue Angel, TCO Certification). Indeed, knowing the certifications is just one part of the intended awareness. More important seems to be that the products and information of the labels are also known and thus recognized in the purchasing processes. We observed a lack of knowledge in the products and even more in the criteria behind the eco-labels. Ramachandiran [2012] found a similar knowledge gap about ENERGY STAR products.

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

In case of the Malaysian students, about half of the interviewed persons consider the ENERGY STAR while purchasing new electronic devices and computer equipment (50 % of the non-ICT students agreed, 57 % of the ICT students agreed [Selyamani and Ahmad 2015]). Ferreira [2011] states that "58 % of respondents would use energy labeling in their software application selection process" (21 % would not).

Indeed, so far, energy consumption of software or similar environmental aspects have not been connected to software products by the users. This is shown in the Green Software Awareness Survey by Ferreira [2011]. Our results confirm this as well.

Overall, respondents state that the survey made them aware of the addressed topics (Ferreira [2011]: "Until today, I have never thought much about it, but it´s a very interesting subject."; similar observations in our study) and lead to further consideration of green computing issues [Pang et al. 2016]. This can be seen as a first step of bringing (mainly) scientific topics to society. Further approaches into this direction will be discussed in the following section. Generally, after informing about the topics, it seems to be very important to move from knowledge to acting.

4 Approaches on how to create awareness

Although there seems to be a general awareness of energy consumption of software and the potential for energy savings by optimizing these products (Ferreira [2011]: "98% of respondents believe that two pieces of software can have different energy consumption profiles for the same functionality", "35% of respondents consider that there's a large potential for savings. 34% of respondents consider that there's a moderate potential for savings."), neither do the programmers address nor do the user request energy efficiency to a great extent [Pang et al. 2016]. Above that, "[energy] concerns are largely ignored during maintenance" [Manotas et al. 2016]. Ferreira [2011] comes up with the following hypothesis: "Given the clear awareness, respondents are taking energy consumption into consideration when buying or developing software applications." Thus, a lot of ideas on how to create this awareness have been published in the last years. They will be briefly presented in the following.

4.1 Education

According to Pang et al. [2016] the integration of green computing aspects in the education and training of programmers is just slightly implemented. Thus, one option to create a bigger awareness for green issues in the context of ICT, is to extend these topics in educational programs. Especially traditional IT projects should be addressed by these activities to show the link between algorithm and energy consumption [Pang et al. 2016] since, so far, "energy usage requirements are more common for mobile and data center projects" [Manotas et al. 2016]. Next to integrating these issues into teaching and education, universities and schools could set a good example. This could support the education

activities even if first studies could not show the positive effects [Selyamani and Ahmad 2015]. Furthermore, the information channels primarily used by students could also be used to spread environmental IT information [Dookhitram et al. 2012]. After leaving universities, the Green IT training should be provided by ICT organizations. According to Kogelman [2011] this is done in under half of the cases as stated by the surveyed ICT managers.

4.2 Labels and information for end users

Overall, labels like the ENERGY STAR, are mentioned quite often regarding the question how to create awareness for environmental topics [Lago and Jansen 2010; Zhang et al. 2014; Selyamani and Ahmad 2015]. According to Ferreira [2011] "46% of respondents classified the impact of energy labeling as strong or very strong". Different studies point out that informing initiatives can increase the user awareness towards energy efficiency of software and consumer's positive attitude towards green products. [D'Souza et al. 2006; Zhang et al. 2014]. As far as we know, there are some initiatives towards labeling green software [Kern et al. 2015]. However, a standardized eco-label is still missing in this context.

4.3 Recommendations and information during development

Since the programming phase seems to be important to monitor energy consumption (Ferreira [2011]: "58% of suppliers monitor energy consumption in the programming stage"), the programmers should be informed about green computing and software issues. Pang et al. (2016) point out that "[only] 18% of the respondents take energy consumption into account during software development.". According to the same study, the energy consumption is more interesting for the programmers in case of mobile development (65 % vs. 12 % in case of software respond that it is a decision factor). So far, corresponding recommendations are missing. However, the practitioners are sure that they can learn about the improvement of the energy efficiency. [Manotas et al. 2016; Pang et al. 2016] Similar to the priorities of the respondents in our survey, the functionality of the software seems to rank first for programmers [Pang et al. 2016].

4.4 Other approaches

Lago et al. propose to do continuous measurement and feedback to bring green knowledge to researchers and practitioners. Such services could "give feedback on the carbon footprint of SBAs [service-based applications] and of their end users". A so-called "service greenery" can offer green software services via Web. However, these ideas are, so far, theoretical concepts. They need to be proven and brought to practice. [Lago and Jansen 2010]

Another possibility to increase awareness for Green IT topics could be related policies [Dookhitram et al. 2012] or official legislation [Kogelman 2011]. According to Kogelman [2011] the

situation that European organizations did not implement Green IT practices is caused by the absence of official legislation and "no pressure from management or customers".

5 Conclusion and outlook

Summarizing, the different studies show that the awareness for environmental issues of ICT could be higher. This is true for different roles in using hard- and software. There are approaches on how to increase the awareness. Some of them have been shortly described in this paper. Indeed, many of the proposed activities still need to be transferred from science to society. The Green IT knowledge of users, developers (including students and practitioners), and managers needs to be enlarged. They need (1) to know that ICT, including hard- and software, has environmental impacts, (2) to get information, tools, methods, and advice on how to transfer their knowledge into actions, and (3) to overall increase the awareness to these issues in their surroundings and organizations. Thus, the next steps in increasing the social interest in green computing should include activities that bring the research activities to industry, policy, education, and consumers. To do so, we will follow up on the development of an eco-label for software products, based on the results of our survey.

References

ANDRAE, A.S.G., AND EDLER, T. 2015. On global electricity usage of communication technology: trends to 2030. *Challenges 6*, 1, 117–157.

D'SOUZA, C., TAGHIAN, M., AND LAMB, P. 2006. An empirical study on the influence of environmental labels on consumers. *Corporate Communications: An International Journal 11*, 2, 162–173.

DOOKHITRAM, K., NARSOO, J., SUNHALOO, M.S., SUKHOO, A., AND SOOBRON, M. 2012. Green computing: an awareness survey among university of technology, mauritius students. In *Conference Proceeding of International Conference on Higher Education and Economic Development, Mauritius. Available from http://tec.intnet.mu/pdf% 20downloads/confpaper/confpaper091224. pdf*.

FERREIRA, M. 2011. *Green Software Awareness Survey. Results*. Presented at Report Workshop Green Software Architecture, Tuesday 7 June 2011, Amsterdam, Netherlands, Amsterdam.

GESI, GLOBAL E-SUSTAINABILITY INITIATIVE. 2015. *#SMARTer2030. ICT Solutions for 21st Century Challenges*. http://smarter2030.gesi.org/downloads/Full_report.pdf.

KERN, E. 2016. The Development of an Eco-Label for Software Products – a Transdisciplinary Process? In *INFORMATIK 2016. Lecture Notes in Informatics (LNI)*, H. C. MAYR AND M. PINZGER, Eds., Bonn, 1285–1296.

KERN, E., DICK, M., NAUMANN, S., AND FILLER, A. 2015. Labelling Sustainable Software Products and Websites: Ideas, Approaches, and Challenges. In *Proceedings of EnviroInfo and ICT for Sustainability 2015. 29th International Conference on Informatics for Environmental Protection (EnviroInfo 2015) and the 3rd International Conference on ICT for Sustainability (ICT4S 2015)*. Copenhagen, September 7 - 9, 2015, V. K. JOHANNSEN, S. JENSEN, V. WOHLGEMUTH, C. PREIST AND E. ERIKSSON, Eds. Atlantis Press, Amsterdam, 82–91.

KOGELMAN, C.-A. 2011. CEPIS Green ICT Survey – Examining Green ICT Awareness in Organisations: Initial Findings. Carol-Ann Kogelman on behalf of the CEPIS Green ICT Task Force. *CEPIS UPGRADE XII*, 4, 6–10. http://www.cepis.org/upgrade/media/kogelman_2011_42.pdf.

LAGO, P., AND JANSEN, T. 2010. Creating environmental awareness in service oriented software engineering. In *International Conference on Service-Oriented Computing*, 181–186.

MALMODIN, J., BERGMARK, P., AND LUNDÉN, D. 2013. The future carbon footprint of the ICT and E&M sectors. In *ICT4S ICT for Sustainability*. *Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013*, L. M. HILTY, B. AEBISCHER, G. ANDERSSON AND W. LOHMANN, Eds. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, Zürich, 12–20.

MANOTAS, I., BIRD, C., ZHANG, R., SHEPHERD, D., JASPAN, C., SADOWSKI, C., POLLOCK, L., AND CLAUSE, J. 2016. An empirical study of practitioners' perspectives on green software engineering. In *Proceedings of the 38th International Conference on Software Engineering Companion*, 237–248.

PANG, C., HINDLE, A., ADAMS, B., AND HASSAN, A.E. 2016. What do programmers know about software energy consumption? *IEEE Software 33*, 3, 83–89.

RAMACHANDIRAN, C.R. 2012. Green ICT practices among tertiary students: A case study. In *Business, Engineering and Industrial Applications (ISBEIA), 2012 IEEE Symposium on*, 196–201.

SELYAMANI, S., AND AHMAD, N. 2015. Green Computing: The Overview of Awareness, Practices and Responsibility Among Students in Higher Education Institutes. *Journal of Information Systems Research and Innovation*. https://seminar.utmspace.edu.my/jisri/download/Vol9_3/JISRI_dec15_P4_Asnita.pdf.

ZHANG, C., HINDLE, A., AND GERMAN, D.M. 2014. The Impact of User Choice on Energy Consumption. *Software, IEEE 31*, 3, 69–75.

Die Promotion wurde im Rahmen
eines Promotionsstipendiums
(1.10.2015 bis 30.09.2018)
von der Heinrich Böll Stiftung gefördert.