EnviroInfo 2013: Environmental Informatics and Renewable Energies
Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

Integrating Aspects of Carbon Footprints and Continuous Energy Efficiency Measurements into Green and Sustainable Software Engineering

Eva Kern[1], Markus Dick[2], Jakob Drangmeister[3], Tim Hiller[3], Stefan Naumann[1], Achim Guldner[1]

Abstract

The energy consumption of information and communication technology (ICT) is still increasing. Even though, up to now, several solutions regarding the hardware side of Green IT exist, the software contribution to Green IT is not well investigated. In our paper, we discuss how to integrate some aspects of carbon footprint calculation into software development processes and we show how ongoing energy efficiency measurements can be established as an integral part of a software development project.

1. Introduction

As 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 into focus only slowly. Furthermore, standard techniques to assess the environmental impacts of products and services, like the calculation of a full-fledged carbon footprint or life cycle assessments, are complicated and laborious. If they are carried out at all, then often by external experts or after the product has already been released to the market.

In order to solve these problems, according to Braungart/McDonough (2009) and Kramer (2012), it is necessary to integrate the consideration of sustainability aspects as early as possible into design processes, because 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 two methods and accompanying tools that complement the generic extension for green and sustainable software processes that was presented by Dick/Naumann at Environmental Informatics 2010. These methods allow the software development team 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 and, especially regarding hardware requirements, also on some social impacts of the software product.

The first proposed method 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. 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 carbon footprint (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 (PAS 2050:2011, 17). 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.

[1] Environmental Campus Birkenfeld, P.O. Box 1380, D-55761 Birkenfeld, {e.kern|s.naumann|a.guldner}@umwelt-campus.de
[2] Sustainable Software Blog, http://sustainablesoftware.blogspot.de/, sustainablesoftwareblog@gmail.com
[3] Environmental Campus Birkenfeld, P.O. Box 1380, D-55761 Birkenfeld, j.drangmeister@gmail.com | tim.hiller@gmx.com

The second proposed method 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 (TDD), the main goals of CI are to minimize the integration effort and to maximize the feedback to software developers referred to their integrations, 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 its development.

2. Related Work

Even if sustainable product design is still focusing on the hardware side, there exist 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).

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 (PCF Pilot Project Germany 2009, Gombiner 2011, Wackernagel/Yount 1998), methods of calculating the CF (ISO/DIS 14067.2), and the representative status of the CF (Laurent et al 2012, Galli 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 that 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).

A first approach of the CF of software products is presented by Taina (2010). For this purpose, he develops metrics and applies these to a fictional project. The software engineering process is separated into different phases. Based on this, different methods are proposed to calculate the CF of each phase.

Hence, in order to get data of the energy consumption of software, measurement methods are necessary. There were already several measurement methods and metrics developed to rate the energy consumption of software. Dick et al (2011) proposed 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 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 as early as possible: during the design and development process. Hindle (2012) showed 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 the development. Johann et al (2012) introduced a method to rate energy efficiency by source code instrumentation. They suggested 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.

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

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 ISO/DIS 14067.2, PAS 2050:2011, and the GHG Protocol (Williams 2013).

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 pilot 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.

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 (PAS 2050:2011, 17). 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., for comparison reasons, we give two oversimplified emission series for an increasing amount of PCs and servers.

3.1. The Example Micro-Enterprise

As 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, 2.1) 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, 14).

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, 3). 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, 14).We assume that the company operates, besides the workstations of the employees, 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, 3). 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 distributed the two employees according to our own preference (longer distance by train, shorter distance by bus). Table 1 shows the resulting distribution of the base scenario.

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

Table 1
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, 63), 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, 2). 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 2011, Fujitsu Technology Solutions 2010). 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. For the storage, we found no adequate data. 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 96.5%. 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 aggregate 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 (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/Loitz 2007, 144). 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

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

a total of 9,234 kg CO_2e per year. Keep in mind that 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.

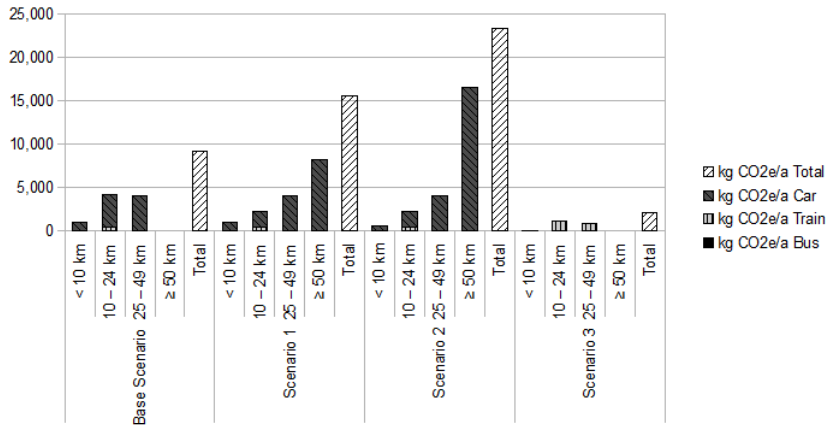

Figure 1
Comparison of the carbon footprint of different commuting scenarios

3.4. Discussion on Impacts from Commuting and Software Operation

In the scenario assumed in Table 1, 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 elimi-

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

nated 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.

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 example, 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. 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).

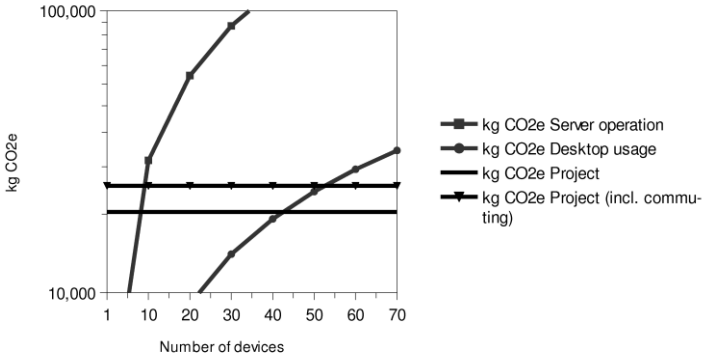

Figure 2
Project carbon footprint vs. oversimplified usage carbon footprints

4. Integrating 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 method with focus on rating energy efficiency during the development process on an on-going basis. In our paper, we introduce a model, 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 (Schubert et al 2012, Bircher/John 2012). Shorter execution times and less system load 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 model describes how to integrate the monitoring of energy consumption and the useful work done during test execution and how to handle the measurements to get an automatically generated energy efficiency report, subsequent to any integration. 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 method level, we are able to rate the energy efficiency on single methods, just as different modules and whole software systems.

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

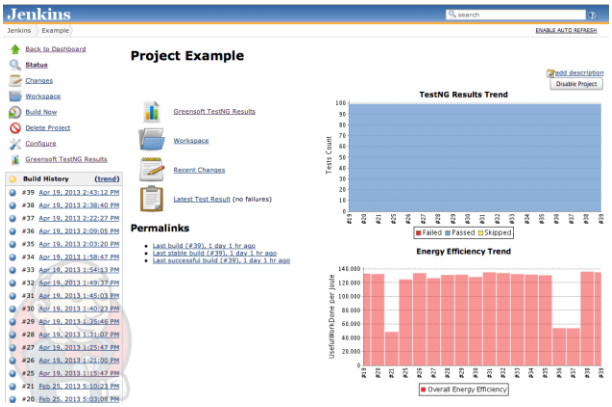

Figure 3
Energy Efficiency Trend in Jenkins

A deep look into the build and test execution workflow of the TestNG testing-framework and the Jenkins CI environment enables us to extend the workflows and instrument the test execution to record system load and energy consumption. We extend the test result report offered by the Jenkins 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 model 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.

5. Discussion and Conclusion

Up to now, several methods and processes for software engineering exist. However, there is a lack of quality aspects in these models regarding energy efficiency. From our point of view, two aspects are important to make ICT and especially 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 paper. 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 see immediately, whether or not a new build might be more energy inefficient. Since one main goal of Green IT is to produce environmental friendly software, the presented methods help developers and designers without changing their software engineering methods in general.

In the future, we plan three steps: at first, the criteria for the calculation of the CF have to be detailed and evaluated. Secondly, regarding CI, we need more research, in order to find out how usage scenarios and energy consumption are connected. Thirdly, we need to develop some tools – or extend existing ones - to support both, the CF calculation process, as well as the CI.

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

6. Bibliography

Albertao, F. (2004): Sustainable Software Engineering, Carnegie Mellon University Silicon Valley, http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering#about, 2010-11-30

Bircher, W.L., John, L.K. (2012): Complete System Power Estimation Using Processor Performance Events. In: IEEE Transactions on Computers, volume 61, number 4, pp. 563–577, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5714687

Braungart, M., McDonough, W. (2009): Cradle to cradle, Remaking the way we make things, London

PAS 2050:2011: Specification for the assessment of the life cycle greenhouse gas emissions of goods and services, ICS 13.020.40; 13.310; 91.190, 2011. http://shop.bsigroup.com/upload/Shop/Download/PAS/PAS2050.pdf

Dick, M., Kern, E., Drangmeister, J., Naumann, S., Johann, T. (2011): Measurement and Rating of Software-induced Energy Consumption of Desktop PCs and Servers, in: Pillmann, W., Schade, S., Smits, P. (eds.): Innovations in sharing environmental observations and information, Proceedings of the 25th International Conference EnviroInfo October 5 - 7, 2011, Ispra, Italy, Aachen, pp. 290–299

Dick, M., Naumann, S. (2010): Enhancing Software Engineering Processes towards Sustainable Software Product Design, in: Greve, K., Cremers, A.B. (eds.): 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, pp. 706–715

Energieinstitut der Wirtschaft (ed.) (2010): KMU-Initiative zur Energieeffizienzsteigerung, Begleitstudie: Kennwerte zur Energieeffizienz in KMU, Wien, http://www.energieinstitut.net/portal/page/portal/EIW_HOME/DOWNLOADS/08053_KMU-Branchenkennwerte-Endbericht_2Q_klimafonds.pdf, 2013-06-28

European Commission (ed.) (2005): The new SME definition, User guide and model declaration, http://ec.europa.eu/enterprise/policies/sme/files/sme_definition/sme_user_guide_en.pdf, 2013-06-28

Fujitsu Technology Solutions (ed.) (2010): White Paper: Life Cycle Assessment and Product Carbon Footprint, PRIMERGY TX 300 S5 and PRIMERGY RX 300 S5 Server, http://globalsp.ts.fujitsu.com/dmsp/Publications/public/wp-LCAPCF-py-tx-rx-300-S5.pdf, 2013-06-29

Fujitsu Technology Solutions (ed.) (2011): White Paper: Energy Consumption ESPRIMO E9900 E-Star5, http://globalsp.ts.fujitsu.com/dmsp/Publications/public/wp-energy-ESPRIMO-E9900-EStar5.pdf, 2013-06-29

Galli, A., Wiedmann, T., Ercin, E., Knoblauch, D., Ewing, B., 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, pp. 100–112

Gombiner, J. (2011): Carbon Footprinting the Internet. In: Consilience-The Journal of Sustainable Development, volume 5, number 1

Grabolle, A., 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, 1. ed., München, Zürich

Grau, A. (2009): Pendler: Die Mehrheit nimmt weiter das Auto, Destatis - Statistisches Bundesamt Deutschland, Wiesbaden, http://www.destatis.de/DE/Publikationen/STATmagazin/Arbeitsmarkt/2009_10/2009_10PDF.pdf, 2013-06-28

Hindle, A. (2012): Green mining: investigating power consumption across versions, in: Proceedings of the 2012 International Conference on Software Engineering, Piscataway, NJ, USA, pp. 1301–1304

ISO/DIS 14067.2: Carbon footprint of products -- Requirements and guidelines for quantification and communication, ICS 13.020.40, 2014-04-15

Johann, T., Dick, M., Kern, E., Naumann, S. (2012): How to Measure Energy-Efficiency of Software: Metrics and Measurement Results, in: IEEE (ed.): Proceedings of the First International Workshop on Green and Sustainable Software (GREENS) 2012, June 2-9, Zurich, Switzerland, pp. 51–54

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5

Jones Lang LaSalle (ed.) (2009): Büroflächenkennziffern – 2009, Studie zur Belegung von Büroflächen, Frankfurt am Main, http://www.joneslanglasalle.de/ResearchLevel1/JLL_Germany_Bueroflaechenkennziffern_2009.pdf, 2013-06-28

Käfer, G. (2009): Green SE: Ideas for Including Energy Efficiency into your Software Projects, Technical Briefing (TB2), 31st International Conference on Software Engineering, Vancouver, http://www.cs.uoregon.edu/events/icse2009/images/postConf/TB-Energy-Efficient-SW-ICSE09.pdf, 2012-03-13

KfW (ed.) (2003): KfW-CO2-Gebäudesanierungsprogramm, Technisches Merkblatt für das Maßnahmenpaket 4, Frankfurt am Main, http://www.umweltheizung.de/images/downloads/F%F6rderungen/KFW-CO2-Geb%E4udesanierungsprogramm-Anlage-B.pdf, 2013-06-29

Kramer, K.-L. (2012): User experience in the age of sustainability, A practitioner's blueprint, Waltham

Laurent, A., Olsen, S.I., Hauschild, M.Z. (2012): Limitations of Carbon Footprint as Indicator of Environmental Sustainability. In: Environmental Science & Technology, volume 46, number 7, pp. 4100–4108

Liebert (ed.) (2000): High-Availability Power Systems, Part I: UPS Internal Topology, http://www.emersonnetworkpower.com/en-US/Brands/Liebert/Documents/White%20Papers/High-Availability%20Power%20Systems,%20Part%20I_UPS%20Internal%20Topology.pdf, 2013-06-29

Mahaux, M., Heymans, P., Saval, G. (2011): Discovering Sustainability Requirements: An Experience Report, in: Berry, D.M., Franch, X. (eds.): Requirements Engineering: Foundation for Software Quality, 17th International Working Conference, REFSQ 2011, Essen, Germany, March 28-30, 2011, Berlin, Heidelberg, pp. 19–33

Naumann, S., Dick, M., Kern, E., Johann, T. (2011): The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. In: Sustainable Computing: Informatics and Systems, volume 1, number 4, pp. 294–304, doi:10.1016/j.suscom.2011.06.004, 2011-11-16

PCF Pilot Project Germany (ed.) (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

Schubert, S., Kostic, D., Zwaenepoel, W., Shin, K.G. (2012): Profiling Software for Energy Consumption, in: Green Computing and Communications (GreenCom), 2012 IEEE International Conference on, pp. 515–522

Statistik der Bundesagentur für Arbeit (ed.) (2013): Arbeitsmarkt in Zahlen, Betriebe und sozialversicherungspflichtige Beschäftigung, Deutschland, Stichtag 30. Juni 2012, Nürnberg, http://statistik.arbeitsagentur.de/Statistikdaten/Detail/201212/iiia6/sozbe-bstbetr-heft/bstbetr-heft-d-0-xls.xls, 2013-06-28

Taina, J. (2010): How Green Is Your Software?, in: Tyrväinen, P., Cusumano, M.A., Jansen, S. (eds.): Software Business, First International Conference, ICSOB 2010, Jyväskylä, Finland, June 21-23, 2010. Proceedings, Berlin, Heidelberg, pp. 151–162

Umweltbundesamt (ed.) (2011): Prozessorientierte Basisdaten für Umweltmanagement-Instrumente (ProBas), Kraftwerksmix zur Stromerzeugung in Deutschland, Record: El-KW-Park-DE-2010, http://www.probas.umweltbundesamt.de/php/themen.php?&prozessid={F6A07B4C-E31B-4838-9AE6-5E18734DFF85}&id=4584374272&step=4&search=, 2013-06-29

Wackernagel, M., Yount, J.D. (1998): The ecological footprint: an indicator of progress toward regional sustainability. In: Environmental monitoring and assessment, volume 51, 1-2, pp. 511–529

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, ICT Sector Guidance initiative (ed.), http://www.ghgprotocol.org/files/ghgp/GHGP-ICT-Software-v2-9-26JAN2013.pdf, 2013-02-06

Copyright 2013 Shaker Verlag, Aachen, ISBN: 978-3-8440-1676-5