Green Software

Final report of the project: Establishing and exploiting potentials for environmental protection in information and communication technology (Green IT), Subproject 3: Analysis of potentials for optimizing software development and deployment for resource conservation

Executive summary

Report commissioned by the German Federal Environment Agency

Project funding reference number 3710 95 302/3

University of Zurich, Department of Informatics, Informatics and Sustainability Research
Prof. Dr. Lorenz Hilty, Dr. Wolfgang Lohmann

IZT Institute for Futures Studies and Technology Assessment, non-profit limited company
Dr. Siegfried Behrendt, Michaela Evers-Wölk

Borderstep Institute for Innovation and Sustainability, non-profit limited company
Prof. Dr. Klaus Fichter, Dr. Ralph Hintemann

Acknowledgements

This study has benefited from numerous suggestions from colleagues. The authors would like to thank the following persons in particular: Dr. Heidrun Moser, Maike Janßen, Marina Köhn, and Stefan Schmitz of the German Federal Environment Agency for their detailed comments on several versions of the text; Prof. Dr. Stefan Naumann, The Environmental Campus Birkenfeld of the University of Applied Sciences Trier, for suggestions concerning the discussion of methodologies; and Patrizia Huber, University of Zurich, for editorial work.

How to cite this document

Hilty, L. M.; Lohmann, W; Behrendt, S; Evers-Wölk, M.; Fichter, K.; Hintemann, R: 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 (in press)

Table of contents

Abstract

Although software products are immaterial goods, their use can bring about significant materials and energy flows. Software characteristics determine which hardware capacities are made available and how much electric energy is used by end-user devices, networks, and data centers. The connection between software characteristics and the demand for natural resources caused by the manufacture and use of ICT systems has been the object of little scientific study to date. The present study breaks new ground by exploring the effects of software on the indirect use of natural resources by hardware. The study identifies starting points in the realm of software that can contribute to conserving natural resources or at least to slowing further growth of their use by ICT systems. A particular focus of the study is on methodological problems arising when assessing the resource use of software products. Such problems include difficulties in defining functional units as well as problems of measurement and allocation. Approaches such as standardizing patterns of use and benchmarks as well as defining and implementing sustainability requirements in the software development process are sketched out as possible solutions. Based on these considerations, the study formulates initial recommendations for action in the areas of research and standardization, product labeling, information for users concerning configuration, best practice guides as well as training and professional development.

Abbreviations

BITKOM	Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V. (Federal Association for Information Technology, Telecommunications and New Media)
BPMN	Business process model and notation
CRM	Customer relationship management
DCIM	Data Center Infrastructure Management
DSL	Digital subscriber line
EASED	Energy-aware software engineering and development
EEG	Erneuerbare-Energien-Gesetz (Renewable Energy Sources Act)
GeSI	Global eSustainability Initiative
GHG	Greenhouse gas
GHGP	Greenhouse Gas Protocol
GI	Gesellschaft für Informatik (German Informatics Society)
GPL	General Public License, more precisely: GNU General Public License
GPS	Global Positioning System
HD	High density
HDTV	High-density television
HSM	hierarchical storage management
ICT	Information and communication technology
IO	Input/Output
IP	Internet protocol
ISP	Internet service provider
IT	Information technology
ITU	International Telecommunications Union
LCA	Life-cycle assessment
LTE	Long-term evolution
MDD	Model-driven development
OSS	Open source software
SUT	System under test
WBCSD	World Business Council for Sustainable Development
WG	Workload generator
WLAN	Wireless local area network
WRI	World Resources Institute

1 Introduction

Software development and use offer potentials for optimization when it comes to natural resource conservation. Although software products are immaterial goods, their use can bring about significant materials and energy flows. Software characteristics determine which hardware capacities are made available and how much electric energy is used by end-user devices, networks, and data centers.

Thus, software is an important starting point for reducing the use of natural resources by current and future information and communications technology (ICT) systems. While "Green IT" has previously often focused on hardware resource efficiency, this study aims to identify starting points for resource efficiency in the field of software.

The connection between software characteristics and the demand for natural resources caused by the manufacture and use of ICT systems has been the object of little scientific study to date. In addition, developers, users, as well as political and business decision-makers are hardly aware of the topic. As the continual development of hardware has always created sufficient processing power in the past, efficiency was not accorded much importance for software development (with the exception of mobile devices). The present study breaks new ground by exploring the effects of software on ICT systems' indirect use of natural resources.

This task has proven to be a methodological challenge because each software product considered in isolation fulfills its function only as a part of a complex ICT system, and therefore only in interaction with other hardware and software components (as well as the user). But it is the total required hardware capacity that determines the demand for natural resources in the form of electricity consumption and the hardware life cycle. In addition, the innovation cycles in the realm of ICT are so short that results based on snapshots in time become outdated quickly. Therefore, the focus of the analysis is on qualitative causal relationships and the dynamics of developments in the field.

The present final report summarizes the findings of the research project in the following chapters:

Chapter 2 documents the result of an analysis of the potentials for resource conservation in various areas as well as individual measures. Against the background of the dynamics of the demonstrated trends, starting points in the realm of software that can contribute to conserving natural resources or at least slow further growth of their use by ICT systems are identified.

Chapter 3 discusses methodological problems and approaches to solutions—in particular with a view to a feasible assessment of software products as "green" software—and highlights the need for further research.

Chapter 4 presents recommendations for action for the purpose of setting political priorities: Where would measures pertaining to software have to begin in order to create incentives to design ICT systems in a more resource-efficient way?

2 Starting points for resource conservation

This chapter presents the most important starting points and potentials for indirect natural resource conservation that can be identified in the field of software.

We assume that application and systems software use natural resources via their utilization of hardware capacity. Software can conserve these resources by utilizing less hardware capacity per unit of performance, minimizing electricity consumption by hardware, or refraining from shortening the operating life of hardware products to less than their technical operating life.

A number of starting points for resource conservation can be identified in the fields of application software (section 2.1) and data centers (section 2.2). Starting points outside the realm of software are mentioned briefly in order to point to aspects above and beyond the questions studied here (section 2.3).

2.1 Starting points in the field of application software

2.1.1 Selectable image resolution

Processing high-resolution photos and videos generally places considerable demands on hardware. Today, multimedia communications services (such as Skype) and multimedia entertainment services (such as Internet TV and computer games) can provide the quality consumers are accustomed to only because sufficient bandwidth and processing power are available. Demands are in lockstep with technical developments: The difference between the only resolution for Youtube videos available in the beginning and the maximum resolution available today is a factor of 164 in data volume per unit of film time.

For this reason, it is important that software products at least give users the freedom to use a lower resolution than what is technically possible (e.g. in the case of video calls and games) and to reduce image resolution easily (e.g. to scale down photos automatically or by default when pasting them into presentations) if high resolution is not required.

However, just as relevant as the question of resolution is the problem that the growing popularity of IP television and the trend from *Broadcast* to *Unicast* (users decide themselves when to watch), which is linked to it, brings about an increase in data traffic that triggers significant consumption of materials and energy in the required infrastructure. Reducing this redundancy, however, is not a question of software, but of business models and network management, and is therefore beyond the scope of this study.

2.1.2 Mobile web preferably via WLAN

The density of publicly accessible wireless access points is already so high in many urban areas that WLAN presents at least a temporary alternative to the cellular network on the move. We expect that software products requiring mobile Internet access function in a more resource-efficient way if they use a wireless network instead of a cellular network. Not only do the cellular networks require the largest amount of energy per unit of data transferred (ranging from 328 to 615 microjJoules per bit according to CEET, 2013, p. 19),[1] they are also considered to be very materials-intensive (Scharnhorst et al. 2006; Emmenegger et al. 2006). From a technical point of view, this will not change fundamentally when the LTE networks are established.

Software affects the choice of communication channel, and it should favor energy efficiency or maintain users' freedom to make that choice. Software products should not force users to communicate via a cellular network even if wireless Internet access is available.

2.1.3 Applying the "app principle" more broadly

Mobile apps are highly efficient as they are reduced to the most important functions and because only limited hardware resources are available on end-user devices.

Mobile apps could provide useful ideas for the world of application software for stationary devices. The simplicity with which mobile apps can be installed and uninstalled as well as their being reduced to the relevant tasks at hand hold significant potential for stationary devices as well.

2.1.4 Implementing web-based applications efficiently

Web-based application software holds high potential for resource efficiency. This is a low-threshold form of software-as-a-service, as web browsers are available practically everywhere. This development may enable end-user devices with low storage and processing capacities to become attractive. Operations requiring large processing capacities can be carried out on the web server and do not burden the web client, and this can typically be a cloud-based service.

Examples include Thin Clients or the Google Chromebook. Prevalence of such devices can contribute to saving resources if activities can be shifted to them from classic stationary PCs. Whether this actually results in saving resources depends on the following conditions:

1. Internet access should be via LAN or WLAN, i.e. *not* via the cellular network.

2. Average capacity utilization of the servers must be high, i.e. the pooling effect must be large enough, and actual load management must make use of it.

3. For the most part, application software should be used that fulfills the requirements mentioned by Williams and Tang (2013).

The studies by Williams and Tang (2013) comparing traditional and web-based application software produced mixed results. In concrete terms, an empirical comparison of Office 2010 and its cloud version Office 365 showed the following: Overall energy savings did accrue (taking into account the sum of end-user devices, network, and data center, but disregarding manufacturing and disposal of the hardware) for Outlook (–8 %) and Excel (–17 %), but energy consumption of the cloud version of Word was 17 % higher (Williams & Tang 2013).

[1] The reason for the fundamental superiority of networks with small radio cells over those with larger ones is that the required transmission power increases with the square of the distance between the transmitter and the receiver.

The differences, which are all fairly small, point to the fact that the benefit of the new architecture is seriously limited if the goal is to reproduce the old concepts.[2]

This shows that instead of web applications imitating complex PC software, new forms of use that are oriented toward reducing the software to manageable, clearly defined tasks will tend to be more beneficial from a resource perspective. Accordingly, only a small amount of data will need to be transmitted. Incorporating the "app principle" into web-based architecture to create "web apps" links the advantages of the two concepts in terms of resource efficiency:

- Requirements for local hardware capacity are low. Provided that data centers transfer data and execute programs in a resource-efficient manner (taking the entire life cycle into account in both cases), this has beneficial effects on the use of natural resources.
- In the case of well-programmed web apps combined with efficient web servers, data transfer is minimal both when starting and when running the program; thus, the risk that increased data traffic would overcompensate for savings is very low.

The combination of these two advantages points to very resource-efficient solutions, e.g. for workstations.

2.1.5 Demand-adaptive software

We characterize a software product as demand-adaptive if it is capable of requiring only those hardware resources (processing power, memory, bandwidth in the network) necessary for a particular task at each point in time. Most software products are not even close to this.

The working group "Software and Green IT" of BITKOM's "Green IT Allianz" ("Green IT Alliance") estimates that the average user utilizes only 7°% of the functionality of standard software frequently and never uses 47°% of all features. The attainable potentials for saving energy are estimated at 10 to 20°% (BITKOM 2010b).

Adaptivity to demand would also be a means for preventing hardware obsolescence. Hardware, less powerful because it was older, could be used for a longer time, if only those modules of new versions of software were installed and run that were actually required.

This would require systematically modular software architecture (and thus also a return to classic principles of software engineering). The modules could be selected when the software is installed and configured, or during operation (web-based software does not require installation in any case). Even though this is possible today in many cases, it is usually difficult for end users to understand the consequences of each available option or whether it is possible to revise such decisions later on, which is why they always select the maximum when in doubt.

Therefore, a central aspect of demand adaptivity is how easy the product makes it for the user to select settings that conserve resources. It should be expected of a "green" software product that resource-relevant settings can be selected at a single, easy-to-access point (Naumann 2013). The simplest way to do so would be by selecting an option, such as

[2] This seems to contradict the study by Masanet et al (2013), which estimated significantly higher potentials for saving energy by means of cloud computing, especially for "productivity software," i.e. everyday office software. However, the two studies are not directly comparable, as Williams and Tang studied a special case in isolation, whereas Masanet et al. estimated the technically possible consolidation effects in the event of widespread introduction and also took embedded energy into account.

"resource conservation mode." In the case of a trade-off between different resources and performance characteristics, the decision can be transferred to users by providing them with sliders to express their preferences (for example, to prioritize speed or resolution).

It is also imaginable that in the future, application software would dynamically and automatically determine a balance between various resources (Naumann 2013). If electricity generated from renewables is available in abundance at a given point in time, CPU-intensive tasks are carried out then. Indexing, compression, and deduplication are examples of CPU-intensive tasks that can sometimes be postponed. If they are performed at times when energy is cheap, then this enables sparing use of network or storage resources, which saves more expensive energy. ICT end-user devices can be part of demand shaping in the smart grid by means of such context-sensitive behavior (see also section **Error! Reference source not found.**).

Overall, demand adaptivity has a static aspect (modular installation, configuration) and a dynamic aspect (resource management at runtime, context sensitivity). However, these aspects are merging due to the trend toward software-as-a-service, e.g. web-based applications, thus opening up development perspectives with a high potential for resource conservation.

2.1.6 The role of open source software

Open source software (OSS)[3] has become more important in recent years. Ernst & Young (2011) calls the period since 2004 the "mainstream era" for OSS. A trend study (Diedrich 2009) found that its degree of use varied across industrial sectors: "OSS is used especially often as a server operating system (84 percent) as well as in other classic fields of operation, such as web servers (81 percent), databases (79 percent), and network infrastructure (73 percent)." Public institutions also rely heavily on OSS. For example, Wikipedia (2013b) lists 18 migration projects in Germany in recent years, among others the municipalities of Munich and Leipzig and the Federal Employment Agency. Consulting firms such as Accenture (n. d.) now offer comprehensive open source services. Android, an open source operating system, is currently most widespread on tablet computers and smartphones (Gartner 2013).

Linux is not yet widely used on desktop computers (its market share was 1.41 % in December, 2011; Heise Open Source 2012a), but is more common on servers and mainframe computers.

For this reason, it makes sense to examine the resource efficiency of OSS separately. Technically speaking, OSS is just like any other software and can, in theory, be built to be just as efficient or inefficient in terms of its resource use. Because of its openness, however, it has advantages that can be exploited in the interest of resource efficiency.

[3] "'Open source' is the term for a range of software licenses whose source text is accessible to the public and for which the license promotes further development. Open source software (abbreviated OSS) is subject to a license recognized by the Open Source Initiative (OSI). In its evaluation, this organization relies on the criteria of the definition of Open Source, which goes far beyond source code accessibility. It is almost identical to the definition of free software." (Wikipedia 2012a). "Free software […] is software that can be executed, examined, modified and disseminated in its original or modified form for any purpose. This includes commercial uses. Free software licenses may include a copyleft clause which states that revised and republished versions of the software must also be free." (Wikipedia 2012b).

The following advantages are mentioned in the literature: adaptability, reusability of code, higher product quality, higher security, open standards, and the absence of licensing costs (e.g. Renner et al. 2005), as are the following disadvantages: a lack of warranty rights, little support provided by developers, uncertain future development of the software, lack of applications, or lack of interoperability with commercial software. These disadvantages are, however, being minimized to an ever greater extent.

In spite of the model of openness (or precisely because of it), the quality of OSS is considered to be equal to that of proprietary software ("albeit not better as a matter of principle," Heinrich et al. 2006) or better (Renner et al. 2005, Heise Online, 2012).

In the case of server operating systems, web servers, and databases, OSS is considered to be more efficient and can generate better performance with the same hardware (Heinrich et al. 2006, referring to Creber 2004 and Enterprise Management Associates, 2006), for which reason it is more widespread in these areas (Diedrich 2009, Heise Open Source 2012a).

Heinrich et al. (2006) explain that "open source software often has lower hardware requirements, which is why existing computer systems can continue to be used even after they have been written off for tax purposes and the costs of new hardware investments can be postponed." The authors refer here to Bräuner (2005), Bokhari & Rehman (1999), and Enterprise Management Associates (2006).

The free availability of the software and independence of a particular software manufacturer can also help achieve a longer hardware operating life because users do not (have to) follow the trend to constantly increasing memory and processing capacity to the same extent as users of proprietary software. One the other hand, security gaps and errors that have become known make regular updates necessary, at least in the operating system, which over time results in inconsistencies in the interfaces and cumbersome configurations, in particular in the case of complex systems, such as Linux distributions.[4]

OSS benefits from speedier development processes (Heise Open Source, 2008) and enables software to be distributed rapidly (Ernst&Young, 2011). For this reason, measures to increase resource efficiency can scale up very quickly.[5] Paulson et al. (2004) compared source code development in various OSS and non-OSS projects and demonstrated that the likelihood of problems being solved was higher in the case of OSS projects.

On the one hand, OSS provides the opportunity to make adjustments to a system to increase energy efficiency at any time. Yet this is balanced by the as yet low number of specialists interested in doing so who are in a position to implement such improvements properly. This is true in particular for adjustments to source code.[6] This problem could be ameliorated by means of campaigns to raise awareness of resource efficiency in the developer community. When it comes to improvements based on exploiting hardware best,

[4] In some cases, proprietary software can enable hardware to be used for a long time, too. For example, Windows XP was introduced in 2001. Microsoft will support users with security updates through 2014.

[5] Potential late delivery of new software under Linux is often caused, among other things, by hardware manufacturers. For example, Intel, the manufacturer of Sandy Bridge, took its time debugging the Linux drivers. The patches based on them were included in official distribution later: only in Kernel 3.4, and in the case of Ubuntu, it was included in the old kernel, but distributed only with the new Ubuntu version 12.04 (Thoma, 2012a).

[6] For example, Linux provides Powertop (n.d.), a fairly good analysis tool for measuring the energy requirements of individual programs. However, its measurement of an experimental change of the framerate depending on CPU load when playing videos, which cut MPlayers' energy use during a lecture at the ETH Zurich, has not yet been published officially.

many developers lack knowledge about the internal workings of the hardware.[7] Here, the task is to seek incentive systems for hardware manufacturers to disclose information about interfaces so that the community can use this knowledge to improve OSS. (Open source hardware projects do exist. They, too, are important and worthy of support from a resource perspective in order to provide OSS with more opportunities for improvements to resource efficiency.)

While settings for good energy efficiency are often provided by manufacturers of proprietary software out of the box or via updates, Linux usually requires manual adjustment of energy-saving settings (Thoma 2012b). This problem could be ameliorated by means of campaigns to raise awareness of resource efficiency in the developer community.

The fact that volunteer work is still required in many areas and that there is no way to enforce implementation of guidelines can be potential disadvantages in the open source universe. However, there is a trend toward hiring developers to work on OSS (Heise Open Source 2012b).

Also, different licenses apply to free software and to OSS, and they can result in legal and financial risks for companies (BITKOM n. d.). This has consequences for decisions on how to design support measures for OSS strategically.

GPL, the best-known license for free copyleft software, applies to many tools distributed with Linux, the Linux kernel itself, and a multitude of projects on the open source SourceForge host (SourceForge n. d.). Taking on resource-efficiency improvements made to GPL software in proprietary software without making the latter public under GPL is not permitted. On the one hand, one would have to include measures to increase resource efficiency in free OSS to enable rapid dissemination of improvements in proprietary software. On the other hand, an increasing number of Linux distributions are including proprietary software as well, thereby undermining the principle of free software. Support of GPL-licensed software would result in disseminating knowledge and would ensure that end users could make use of all the improvements[8]. At the same time, strengthening free (and open source) software would continue. Campaigns informing companies about legal issues could lower the obstacles to switching to OSS.

Knowledge about designing and configuring software in an energy-efficient manner is insufficient today, both for OSS and for proprietary software, and there is a general lack of experts who apply this knowledge in practice. For this reason, it is important that existing knowledge is made available, for example through workshop series such as “Energy Aware Software Engineering and Development” (EASED), “Green and Sustainable Software” (GREENS), “Software Engineering Aspects of Green Computing” (SEGC), as well as conferences (such as ICT4S, ICT for Sustainability) and magazines. The problem has been recognized by the research community and is being tackled. Available knowledge can be brought to bear and disseminated quickly especially in OSS. Research initiatives in this regard that develop, collect, or disseminate knowledge on resource and energy efficiency of software contribute to resource conservation by software in the medium term.

---

[7] For example, manufacturers of graphics cards usually provide device drivers only for Windows (in compiled binary form) and refuse to give developers of open source drivers the necessary hardware information so as not to reveal details of their technology via the sources which would then be open. In contrast, hardware manufacturers collaborate more closely with Microsoft and provide details of their proprietary drivers.

[8] This is not guaranteed if open source and proprietary software are combined.

2.2 Starting points in data centers

Data centers are among the largest electricity consumers in Germany. The connected load of large data centers amounts to several megawatts; thus, they are comparable to the energy-intensive industrial enterprises exempted from the EEG surcharge (surcharge under the Renewable Energy Sources Act, the *Erneuerbare-Energien-Gesetz*). Servers and data centers account for 1.8 % of electricity consumption in Germany (Hintemann & Fichter 2013). Stobbe et al. (2009) assume that the electricity consumption of servers and data centers will increase by almost 30 % between 2010 and 2020 in the baseline scenario. Even in a "Green IT" scenario, they expect an increase in electricity consumption of more than 10 % by 2020.

While opportunities for saving resources in data centers through more efficient hardware have been under discussion for years and are being implemented at least in part, the corresponding potential when it comes to software has been granted little attention to date, with the exception of server virtualization. In the following, three approaches promising relatively high potential for increasing resource efficiency will be discussed.

2.2.1 Dynamic predictive load management

Dynamic predictive load management means managing IT hardware and the infrastructure of a data center in such a way that they are utilized as best as possible. Tasks that are not time-critical are shifted to periods when hardware utilization is low, or hardware is turned off when it is not needed. Comprehensive dynamic load management is not limited to IT services and IT hardware, but also includes the data center's infrastructure components. For example, parts of a modular uninterruptible power supply (UPS) can be switched off, or air conditioning can be turned off in areas of the data center where hardware has been switched off (Nebel et al. 2009).

To date, the various parts of the data center have usually been managed by different individuals or departments of the data center, and with different tools—some by systems and network managers, others by facility managers. Many different solutions for Data Center Infrastructure Management (DCIM) aiming to unify the two worlds and thus improve energy efficiency in data centers are currently on the market. However, providers and market research or consulting firms use the term DCIM in different ways. At least, the solutions usually enable comprehensive monitoring and management of IT hardware and data center infrastructure. However, they only rarely achieve complete and integrated load management, including the levels of hardware virtualization and services (Reder 2012).

In the case of peak loads or unfavorable conditions, e.g. very high ambient temperatures, shifting IT loads to other data centers is a possibility.

Experience shows that energy savings on the order of 30 % could often be realized *even with the energy management software systems already available in companies* (e.g. providers' server management suites; Baumeister 2012), but that they are often not implemented because of the time and effort required (Müller 2013). Providers of DCIM solutions also hold out the prospect of savings on the order of 25 % (Lanline 2012).

These assessments are confirmed by research projects such as GAMES (Green Active Management of Energy in IT Service centres, www.green-datacenters.eu), which established potential CO_2 footprint reductions of approx. 25 %. According to the responsible project managers of "Cool em All" (www.coolemall.eu) and "Adaptive Computing for Green Data Centers" (www.ac4dc.de), those projects have established similar figures.

Load management across data centers can also entail cost advantages for data center operators by using regionally differentiated dynamic electricity prices, in addition to improvements in their own capacity utilization. A study by the Offis Institute established additional (financial) savings potential of approx. 5 to 10 % here, depending on the dynamics of electricity pricing (Nebel et al. 2009). From the perspective of the entire energy system, application of this solution means better adaptability to dynamic electricity generation from renewables (demand shaping). From a resource point of view, adaptability to price signals is an important advantage of dynamic predictive load management: setting prices accordingly makes it possible to save fossil fuels and limit power grid capacities. This advantage accrues even under the pessimistic assumption that the same ICT performance is achieved with the same electric capacity.

On the basis of the studies and sources mentioned, it can be said that software solutions for dynamic predictive load management in data centers promise energy savings potentials of 25 % to 30 %. Improving average capacity utilization also means that significantly less hardware is required, which in turn entails high potentials for improving materials efficiency.

2.2.2 Information and data management

Calculations prepared by the Borderstep Institute for the research project AC4DC have shown that data storage currently accounts for more than 10 % of data centers' energy consumption, and this figure is rising. One reason for this is the fact that storage costs per gigabyte are constantly decreasing, which often determines how data is handled in data centers. In the past, the capacity of hard drives has doubled roughly every 18 months, as predicted by Moore's Law.[9] According to Experton (as cited in Bayer 2009), the costs of storage hardware are decreasing by 30 % per year. As a result, there is little incentive for businesses to use memory efficiently (Bayer 2009).

One starting point for more efficient data management is to reduce the amount of data to be stored. For example, unnecessary copies of data could be avoided and data no longer needed could be deleted. One way to implement this idea is that staff members must explicitly mark those e-mails they receive that are to be stored long-term. All other e-mails will be deleted automatically after a certain period of time (Rüdiger 2011).

In addition, electricity consumption can be reduced if data is always stored in the most energy-efficient medium. For example, archive data could be stored automatically and in an energy-efficient manner on tapes. Such software tools for hierarchical storage management (HSM) have been on the market for quite some time. They shift data accessed only rarely to cost-efficient storage systems which are usually also the most resource-efficient ones. As HSM operates automatically and without requiring users to do anything, its application in data centers is quite common (Müller 2013). With increasing prevalence of solid state disks in data centers, the correlation between cheap and energy-efficient storage systems no longer holds. Solid state disks are currently among the costly, but energy-efficient storage systems (Wilde 2013).

Software tools and concepts for information and data management that go beyond HSM are available: information lifecycle management solutions (Ehmann & Hintemann 2004) and enterprise content management solutions. However, they are seldom used in practice as

[9] In 1983, 10 MB IBM hard drives were common; today (2013), the first 8 TB hard drives are on the market, which corresponds roughly to 20 reduplications.

technologies for resource conservation as they require substantial time and effort for categorizing the information and data (Müller 2013).

Because of constantly increasing storage capacities and decreasing prices for storage on hard drives, there is little incentive to reduce amounts of data. The advantage of cost-efficient memory also has a disadvantage for companies, namely that identifying and finding the relevant data requires increasing amounts of time and effort (Vilsbeck 2012). Software solutions that help *avoid* the flood of data would contribute more to resource efficiency than "big data" solutions that help evaluate very large amounts of unstructured data.

2.2.3 Data compression and data deduplication

Another way to reduce the amount of data to be stored is data compression and deduplication. Data compression encompasses techniques with which digital data can be altered so that less memory is needed and transmission time between two IT systems is reduced. Compression of data is also called coding, decompression at a later point in time decoding.

Lossless data compression ensures that data matches the original exactly even after it has been coded and decoded. Lossy data compression usually does not enable error-free decoding, but it does permit higher compression rates. While lossless compression is required above all for program files, company databases, etc., lossy compression is used in particular for multimedia files such as images, videos, or audio files.

Deduplication of data is a special compression technique which analyzes the existing data for redundancy. It identifies and deletes redundant data, i.e. data existing in multiple copies (Geer 2008; Pelkmann 2010). The efficiency of deduplication depends on the data available in a concrete case. If there is a lot of similar data—e.g. in the case of numerous versions of a PowerPoint presentation with minor changes—then very high compression rates are possible. Manufacturer EMC reports a factor of 10 to 30 by which the volume of data can be deduplicated (EMC 2013).

Other manufacturers (Unterseher 2008) and consultants (Wilde 2013) assume compression rates of 1:10 which are common in practice for backup and archiving. Measurements of real systems confirm these orders of magnitude (Meyer & Bolosky 2012). Initial applications in a project funded by the German Federal Environment Ministry's Environmental Innovation Programme and carried out by Erecon AG demonstrate that data deduplication can be suitable for direct use "on the fly" and not only for backup systems in data centers (BMU 2012). However, only lower deduplication rates (e.g. 1:5) are possible here (Wilde 2013).

Deduplication significantly reduces the amounts of both energy and hardware (He et al. 2010). According to Borderstep Institute calculations in the AC4DC project (n.d.), approx. 12.5 million hard drives were installed in German data centers in 2012. A rough calculation demonstrates the order of magnitude of savings potentials: Assuming that the technique were to be used on only 20 % of all systems because of technical and organizational limitations, inline deduplication with a deduplication rate of 1.5 would permit using approx. 2 million fewer hard drives. Data centers' energy consumption could be decreased by approx. 2 % overall in this way.[10] In addition, less hardware would be required for infrastructure (air conditioning, etc.).

[10] According to calculations prepared by the Borderstep Institute in the ongoing AC4DC project (n. d.), approx. 12.5 million hard drives were installed in German data centers in 2012, and they accounted

It is questionable whether deduplication will be employed on a larger scale in the future at all. For example, the author of an article in the specialist magazine speicherguide.de believes that using a cost-efficient hard drive system with SATA drives is more economical in many cases than purchasing a deduplication system (Rieß 2012).

2.2.4 The challenge of a heterogeneous data center market

Resource-efficient software solutions for data centers must compete in a very heterogeneous market. This begins even with the structure of the data centers. Most of the approx. 52,000 data centers in Germany are equipped with fewer than 10 physical servers. Approx. 60 data centers with more than 5,000 servers each are at the other end of the spectrum. In total, more than twice as many servers are installed in these 60 data centers than in the 31,500 small locations (Hintemann & Fichter 2012, 2013).

According to a survey by the Borderstep Institute,[11] almost 50 % of the large data centers (more than 5,000 servers each) are so-called colocation data centers that provide IT space including infrastructure services as a package. As IT hardware and infrastructure are not operated by a single entity there, this reduces their potential for comprehensive and efficient software solutions significantly. In addition, about one-quarter of the major data centers are host computing centers where the IT hardware is managed by the operator, but the software is managed by the customers. Here, too, the opportunities for using resource-efficient software solutions are severely limited.

Not only the data centers themselves display a very heterogeneous structure. A large number of different players are active in the data center market, including IT hardware and software providers, infrastructure solution providers, IT consultants, data center planners, and firms offering total system-based solutions. Some of them pursue goals contrary to establishing resource-efficient software solutions in the market. For example, hardware providers are primarily interested in selling hardware. Some players also attempt to establish proprietary systems in the market—in combination with appropriate software solutions—which makes comprehensive solutions more difficult to achieve.

Because this market is not transparent, consumers have only severely limited opportunities to make decisions contributing to resource-efficient service provision.

2.3 Additional starting points for resource efficiency

2.3.1 User behavior in social networks

The palette of services originally offered by Facebook, for example, required relatively little computing power. Implementation of videotelephony, facial recognition, games, and tracking functions with information about the locale and advertising has significantly increased the need for it.

In the case of the major social networking platforms, one must assume today that the resource efficiency potential available at the *software level* has largely been exploited,

for just under 12% of the electricity consumed by the data centers. Reducing the number of hard drives by 2 million would thus correspond to a reduction in electricity consumption of about 2 %.

[11] The Borderstep Institute has data on the size, location, operator, and purpose of approx. 80 % of the major data centers in Germany.

simply because it would be impossible to process data for hundreds of millions of users in the absence of innovative measures. As the operators of data centers come up against absolute limiting factors (for example, with regard to their power consumption) sooner or later, it is in their own interest not to waste energy or server capacity. Investments even in small resource-saving software improvements pay off because of the multiplication effect: the code in question is run thousands of times for many millions of users.

Resource conservation when using social networking sites is therefore not a question of software, but can be achieved only by the following means:

1. the operator's data storage policy,
2. operating data centers in a way that conserves resources,
3. user behavior.

The first point concerns questions such as: Does the operator provide for physical deletion of data; does the user have a "right to be forgotten"?[12] In an optimized software landscape, the second point concerns cooling as well as sourcing electricity (green electricity). The third point will be discussed briefly in the following, as it is about interactions with software design.

The software is usually designed so that it motivates users to interact with the platform frequently and for a long time. After all, the more intensely members interact with a platform, the more attractive it is for companies to place their advertising there. That is why various means for increasing the amount of time users spend on the platform are implemented (e.g. games). The result is increased data traffic, which does make a difference in combination with energy-intensive mobile access.

From a resource perspective, the combination of social networks with location-based services seems especially unfavorable because it promotes access via cellular networks in particular. The fact that the market of mobile social networks (e.g. Foursquare[13]) is considered relevant for the future is evidenced by the involvement of major corporations such as Nokia, Intel, or Samsung. For example, all of them are collaborating with GyPsii, a provider of mobile social network software (Heise Online 2011b).

Mobile social networks have strengthened the "always on" mentality further. In the interest of resource conservation, platforms should be supported that back an opposing trend which can be characterized roughly as follows: minimizing interaction, providing only the most necessary functions, reducing images to the resolution required as the default setting, providing easy ways to physically delete data. Such a trend would be advantageous not only for resource conservation, but also for informational self-determination.[14]

[12] Regarding a "right to be forgotten" from a privacy perspective, see also Hilty et al. (2012).

[13] Foursquare claims to have eight million users globally who sign on 2.5 million times per day with check-ins. The number of members is said to increase by approx. 35,000 new members daily. The platform for retailers is used by 250,000 companies (as of November, 2011).

[14] Frequent mobile interaction with internet platforms makes it possible to generate profiles of users' movements and contacts (Hilty et al. 2012).

2.3.2 The polluter pays principle

As the example of spam demonstrates, the functional principles of the Internet are far removed from the polluter pays principle. Otherwise, spammers would have had to pay an electricity bill as high as that of the entire population of Bangladesh (150 million) in 2008.[15]

Allocating the use of technical (and thereby indirectly of natural) resources on the Internet to those responsible is extremely difficult, and even in the case of criminal offenses, such as the operation of botnets[16], it is almost impossible to accomplish.

Another example: billing customers for the electric energy used by an Internet service provider (ISP) for the services provided would be possible only to a limited extent because the ISP cannot track everything the customer’s data traffic triggers.

If one views the Internet as a public good and not as a commercial service, then it is very difficult to allocate costs for resource use to a particular person or other entity causing them. Changing this situation would, however, have far-reaching consequences, for example the danger of widespread monitoring of activities on the Internet. In the long term, this could result in a conflict of goals with democratic fundamental rights. From a technical point of view, solutions respecting data protection are possible in principle.

If the latent conflict of goals on the Internet between the polluter pays principle and fundamental rights is not resolved, the resource load of further exponential growth of data traffic will inevitably be shifted to where no costs accrue to the person or entity causing them.

[15] McAfee’s estimate of spam-related electricity consumption totaling 33 TWh corresponds roughly to the annual electricity consumption of Bangladesh (Hilty & Lohmann 2011).

[16] A botnet is a set of computers that are controlled remotely via the Internet, usually without their owners’ knowledge. Botnet operators can use these computers for their own purposes, for example to send spam, attack websites, host illegal contents, or spy out data.

3 Methodological challenges and approaches

The question as to how a “resource-efficient” software product can be differentiated from a “less resource-efficient” one will now be made more concrete, based on the starting points identified above. This question is confounded by unsolved methodological problems, as software cannot be examined by applying the standardized method of life-cycle assessment in a routine fashion. Significant difficulties are yet to be overcome on the path to a label for “green software.”

This chapter will draw upon the relevant literature to show which methodological challenges arise in particular and will provide an overview of existing approaches to solutions.

3.1 Methodological challenges

3.1.1 Definition of functional units

Application of the LCA methodology begins with defining functional units, i.e. precise, quantitative definition of the service which the product system in question is to perform (Rebitzer et al. 2004). All resource uses are set in relation to this unit. It is the basis for comparing different product systems in terms of the efficiency with which they provide the service. This presupposes functional equivalence, i.e. the functional units must be indistinguishable from one another, or at least interchangeable.

Resource efficiency is the relationship between a particular use or result and the resource use required to produce it. Here, a functional unit corresponds to the use or result. In other words, it is the unit for the numerator of efficiency, while a measure for resource use is in the denominator:

$$\text{resource efficiency} = \frac{\text{number of functional units provided}}{\text{number of units of resources used}}$$

Defining functional units for software is generally difficult. One reason for this is that most software products fulfill a large number of functions, and only rarely do two products fulfill the same ones. Even different versions of the same software product often differ in functionality. Even if one were to focus on a single function, e.g.

“creating and sending 1 e-mail,"

functional equivalence could easily be called into question; for example, e-mail programs may offer different support when creating the e-mails (managing contacts, correcting typing errors, etc.). In addition, this unit is not defined precisely: the number of characters, the size of attachments, etc., would have to be laid down exactly. Thus, functional units are so specific that special methodological effort is required to describe one or more statistically representative functional units. However, whether or not a functional unit is statistically representative can be ascertained only on the basis of empirical user behavior data.

If software is to be evaluated in terms resource efficiency, the functional unit(s) selected must be representative, which would require a large amount of effort for gathering data. As changes are possible to usage patterns as well as the characteristics of the software products (via updates), results would be out of date in a short time. This problem is compounded by the methodological challenge of multifunctionality: an individual software product can fulfill several functions; this requires defining various functional units which, in the context of an LCA, are to be treated as co-products.

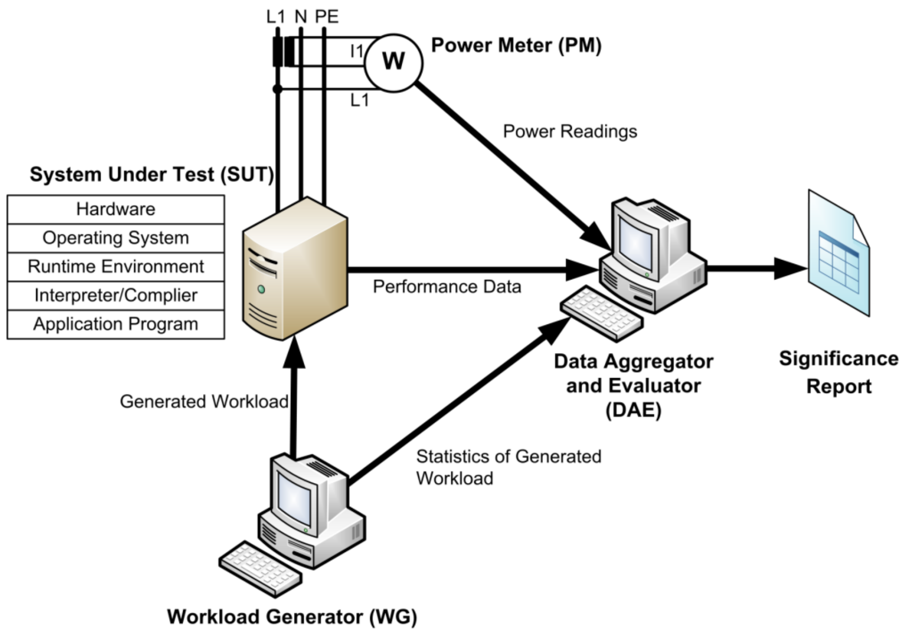
3.1.2 Measuring the energy consumption of software

The seemingly simplest aspect of resource use by software—namely electricity consumption via hardware—poses a methodological challenge. This is due to the fact that a software product is always part of a complex ICT system which is responsible as a whole for energy consumption, whereby each individual component of the system influences the resulting electricity consumption.[17] As a rule, existing indicators for ICT systems' energy efficiency (Erek et al. 2013, Drenkelfort 2013) do not provide for considering the effects of software in isolation (Van Bokhoven & Bloem 2013).

Figure 1 shows an example of a setup for measuring the energy consumption of software (Kern et al. 2013). In this case, the System Under Test (SUT) is divided into five components: hardware, operating system, runtime environment, interpreter/compiler, and the application program. In order to compare the electricity consumption of different applications, for example, the other four components must be kept constant for methodological reasons. This makes the result extremely dependent on context: a statement such as "program A uses more energy than program B" would be justified only if "executed by hardware P, under operating system version Q, in runtime environment R, and with interpreter S" were added.

While the problem of context dependency also arises when evaluating other products in terms of their resource consumption, the multitude of influencing variables in the case of software products poses a particular challenge.

Figure 1: Example of a setup for measuring the energy consumption of software



Source: Kern et al. (2013; p. 91)

[17] Bozzelli et al. (2013) provide an overview of the literature on the relevant metrics. Complex energy consumption models are necessary for estimating energy consumption in cloud environments (Chen et al., 2012).

The problem is exacerbated if the interaction of the components in networks is included, which is the norm today. For this reason, studying the energy consumption (or more generally, the resource consumption) of software is extremely complex and implies breaking new ground in terms of methods.

Figure 1 shows a “workload generator (WG)” whose task is to generate a workload for the SUT like a real user would. The choice of usage patterns or usage scenarios, which is automated by the WG, is to be considered an approximate solution to the problem of defining functional units discussed above.

3.1.3 Allocation problems in the case of fluctuating load

If several software components use capacity of the same hardware component, then the problem of allocation arises, as is often the case in LCA studies. Viewed in terms of production management, this is an example of co-production, as one hardware component produces various products simultaneously: computing power for software 1, computing power for software 2, etc. In other words, resources used by hardware must be allocated, according to an appropriate formula, to the various software products (to be precise, to the defined functional units which are delivered by the software products) using the hardware.

The problem of allocation becomes particularly obvious when it comes to hardware in network nodes, e.g. routers, as well as servers. In these cases, an individual software product running on an end-user device is usually responsible for an extremely small fraction of the total load handled by the router or server.

Allocation can be based either on the fraction of the *currently used hardware capacity* or on the *total hardware capacity available*. Usually, the latter option is used; otherwise, reserve capacity would not be allocated at all except for peak usage times.

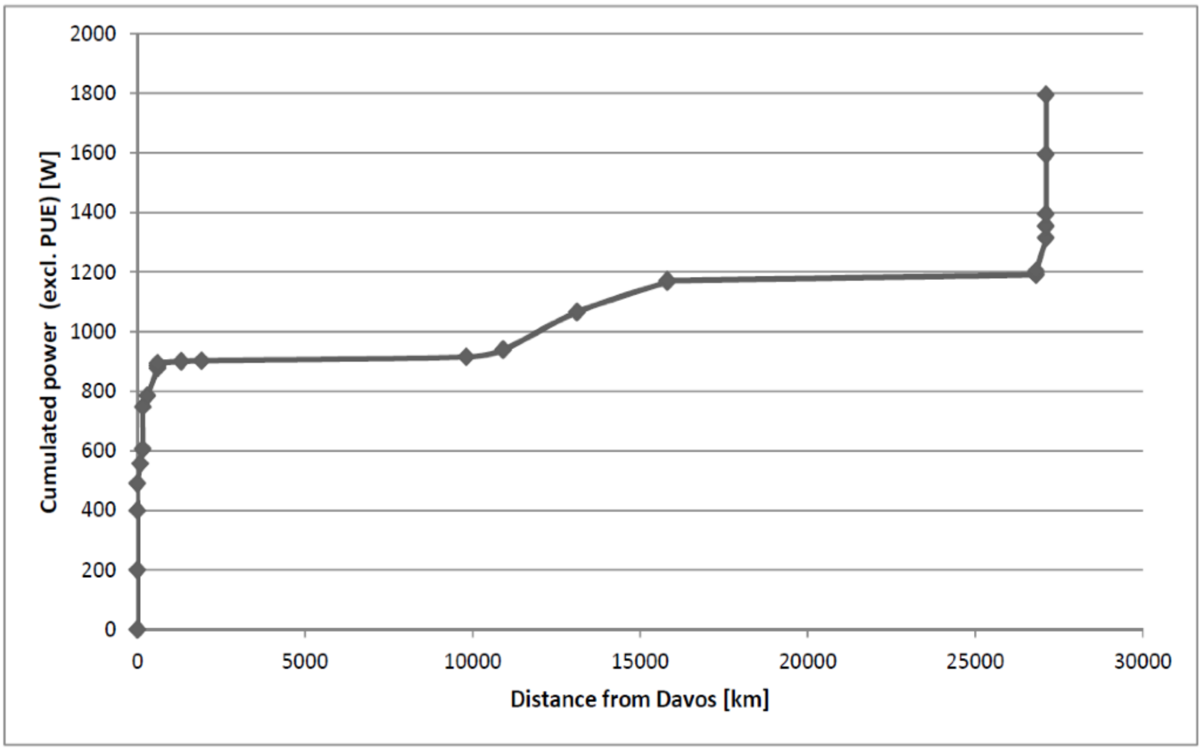
The reason why this is still an unsolved methodological problem shall be elucidated using the example of the energy consumption measured for a video conferencing Internet connection between Switzerland and Japan. Coroama et al. (2013) determined the power consumption and the capacity utilization of all the network nodes and network connections between the conference center in Davos and the University of Nagoya (4 simultaneous bidirectional full HD channels). The data traveled a distance of 27,117 km in 24 hops. Figure 2 shows cumulative electrical power consumption for the entire distance, from Davos to the destination. The most striking result is that consumption is highest near the two end nodes and that hardly any energy is used to cover the long distances (the deep sea fiber optic cables through the Atlantic and the Pacific appear as practically horizontal segments in the figure).

This finding can be explained in part by allocation. Near the end nodes, there are nodes whose load was less than 3 % at the time of measurement (some less than 1 %). In contrast, a very large number of data traffic streams are combined in the Internet backbone, which makes capacity utilization more constant and reduces the difference between average and maximum capacity utilization. In contrast, total reserve capacity in the network nodes with low usage, which is available for rare peak usage times, is allocated to very low average capacity utilization according to the allocation scheme mentioned above.

If one takes this situation as given and considers the amount of data transmitted, one can derive that transmitting one gigabyte across this distance requires less than 0.2 kWh, assuming an estimated PUE value for the nodes and connections (Coroama et al., 2013). Compared to previously published figures, this is relatively low, even though most of it is due by way of calculation to the extremely low capacity utilization of some in-house and region network nodes.

Assuming that 1,000 such video conferences were taking place simultaneously in Davos, the figure 0.2 kWh/GB would no longer apply, because the capacity utilization of the regional nodes would increase significantly. The energy intensity per conference would be significantly lower for this reason; in other words, one would *overestimate* energy consumption if one were to continue using 0.2 kWh/GB in calculations. Only at the point at which capacities were increased would more electricity be used in absolute terms.

Figure 2: Cumulative electrical power consumption of an HD video conferencing connection



Source: Coroama et al. (2013)

Allocation problems also occur in peer-to-peer (P2P) networks in which the individual users make hardware capacity available to the network. For example, Skype telephone calls are transmitted via individual users with free capacity (so-called supernodes), who do not notice that this is occurring. This increases the supernode's utilization of its own hardware capacity, and if allocated in this way, the resource efficiency of *other* programs running simultaneously on that hardware is improved by way of calculation. Independently of this effect, the question arises whether one should allocate the load due to *other people's* Skype calls to *one's own* Skype calls. The results depend on how these questions are decided.

These examples are intended to illustrate that the results of resource estimates, which are (necessarily) dependent on decisions concerning allocation, should not be interpreted independently of their context.

This methodological difficulty makes the question of assessing cloud computing in terms of resources a primary problem of estimating capacity utilization of networks and servers and selecting allocation rules.

3.2 Existing approaches to solutions

3.2.1 Concentrating on individual aspects which can be measured

Some authors deal with the methodological problems by focusing on individual aspects which can be measured:

- There are a number of programs that use PC software to determine average capacity utilization of processors, using standardized benchmarks; these programs are intended to help users select the most energy-efficient among several software products (see, e.g., Amsel und Tomlinson 2010).
- Wilke (2012) measured local energy consumption of smartphone apps with the goal of creating the basis for an energy label. He compared apps with similar functions (e.g. e-mail clients) and defined simple use cases as functional units.[18] Two important factors contributing to natural resource use—production of the smartphone and the data traffic it induces in the cellular network—were not taken into consideration, as Wilke's work was focused primarily on battery service life.
- Naumann et al. (2008) introduced the Power Indicator (called "Green Power Indicator" today), an add-on for the web browser Firefox, which shows whether a website is running on a server operated with electricity generated from renewables. The add-on accesses a centralized list of "eco-providers."
- Zapico et al. (2010) created Greenalytics, a technique for automatically estimating the energy consumption of websites for clients, networks, and servers. However, the estimates are very rough and are based mainly on data volume and access statistics. According to Grosskop and Visser (2013), this does not enable measurement of energy consumption.

3.2.2 Efforts toward standardization

Both the problem of defining functional units and the problems of measurement and allocation can be partially alleviated by means of standardization, because it achieves comparability of results at least.

In individual areas of testing software products, standardized usage scenarios could take on a role similar to that of standardized driving cycles for exhaust and fuel consumption tests of automobiles. It is debated, however, how realistic they are[19]; as a result, it appears that because of the even greater complexity of software products, broad consensus about "software driving cycles" would appear to be difficult to achieve among various manufacturers, the scientific community, public agencies, as well as consumer and environmental organizations. While it can be assumed implicitly in the case of a car that its function is to travel a given distance, in the case of software, even the selection of the function(s) on which the testing cycle is based is a normative decision with the corresponding potential for conflict.

[18] e.g. "Check Inbox," "Read Mail," "Open Attachment." Significantly higher energy consumption was noted for those products showing advertising (Wilke 2012).

[19] This is apparent even on the German Wikipedia page on "driving cycles": "The standardized driving cycles represent average profiles as a basis for comparison of different vehicles. They frequently do not match the customer's user profile …" (Wikipedia 2013a).

Dick et al. (2011) define load profiles for ICT systems according to ISO/IEC 14756.[20] These profiles are randomized, i.e. the order and duration of user actions are selected at random within a given framework. Average energy consumption of the profile in a given configuration of the ICT system can be determined by repeating the profile numerous times. If such load profiles are to be realistic, they must be oriented toward "typical workloads" in the intended area of application in each individual case (Dick et al. 2011; p. 292). Kern et al. (2013) call for the definition of usage scenarios for standard software in order to enable systematic comparison of the energy consumption of different software products and configurations. Such usage scenarios would then also form the basis for the load profiles which are measured. Standards for usage scenarios are not yet on the horizon. Business process models (which are defined in BPMN[21], for example) could serve as the basis for load profiles.

The International Telecommunications Union (ITU) prepared an "Assessment framework for environmental impact of ICT" (ITU 2012a). It is a part of the "ITU Toolkit on Environmental Sustainability for the ICT sector," which was developed by ITU-T in collaboration with more than 50 organizations and ICT companies with the goal of defining environmental sustainability requirements for the ICT sector ("Environmental Sustainability Requirements"; ITU 2012b).

The "assessment framework" gives an overview of the existing norms and standards, including the ITU-T standard L.1410, "Methodology for environmental impact assessment of ICT goods, networks and services," which is based on the ISO standards for LCA. It specifies a methodology for estimating the environmental impacts of ICT systems (ITU 2012c).

Of the document's 85 pages, 2 pages in the main text and 2 pages in the annex are devoted to the topic of software. This demonstrates that software is not yet firmly established as a subject of environmental standards. The ITU-T norm focuses on regulating the estimation of impacts of the *production* of software and the allocation of the production stage (divided by the number of licenses sold).

The problem of defining functional units in the case of software is mentioned in an example on word processing: "The function experienced by a user of a word processor program is to deliver word processing of documents electronically. The corresponding functional unit could then be the number of pages processed per time unit (e.g., one hour) during the operational lifetime (e.g., three years). Finally, the reference flow is defined as one unit of word processing software (distributed e.g., in a CD with packaging)" (ITU 2012c; p. 8). This example, too, focuses on the impact of the production stage of the software; even its traditional distribution on CDs is considered. In contrast, the impacts of software due to usage of hardware capacities in the usage stage are not discussed.

The functional unit for ICT services is generally proposed to be usage during one year ("annual service use"). The authors point out that realistic usage scenarios must be defined for this functional unit and that network capacity must be allocated, yet concrete suggestions for solving this problem are lacking (ITU 2012c; p. 9).

[20] ISO/IEC 14756 describes a procedural model for measuring software performance. Classic performance characteristics of ICT, such as execution time and throughput, are measured for given load profiles. This norm has no specific reference to software resource efficiency, but its approach for defining and using load profiles as well as conducting measurements can be applied.

[21] Business process model and notation, an Object Management Group standard for describing business processes (BPMN 2013).

The GHG Protocol, an initiative founded by WRI and WBCSD in 1998, develops standards for gathering data about and reporting on companies' greenhouse gas emissions. At the recommendation of ICT firms, the GHG Protocol, in collaboration with The Carbon Trust and the Global eSustainability Initiative (GeSI), developed specific guidelines for the ICT sector ("ICT Sector Guidance," GHGP 2013a, Stephen & Didden 2013).

"ICT Sector Guidance" covers the following topics: telecommunications services, desktop-managed services, cloud and data center services, hardware, software, avoiding transportation (GHGP 2013b). The chapter on software is currently available only in draft form (Version 2.9, GHGP 2013c). The most important aspects for the question at hand will be presented in the following.

In contrast to ITU-T, the GHG Protocol focuses on the impacts of software in the usage stage: "Up to 90 % of the energy used by ICT hardware can be attributed to the application software running on it, and the design of software can have a significant impact on the amount of energy used. It is therefore important that software designers carefully consider the energy use of the software, and design software efficiently to reduce the energy use. Examples of where better software design can reduce the energy use are: optimizing the CPU usage; optimizing the disk IO usage; optimizing remote calls such as database calls, and web accesses." (GHGP 2013c; p. 4)

The central contribution of the chapter on software is the detailed suggestions on how to approach preparing, conducting, and evaluating electricity consumption tests for systems and application software, including the libraries used. Using benchmarks and correctly taking energy management into account by the operating system or the application software are considered particularly important. Testing of remote devices and virtual machines are also considered.

No concrete suggestions are made for benchmarks as this standard describes the basic methodology. In so doing, the standard does make a substantial contribution to solving the problem of measurement (section **Error! Reference source not found.**), but implicitly presupposes that meaningful and accepted functional units can indeed be defined (section **Error! Reference source not found.**); after all, the definition of benchmarks must be founded upon them. It is not yet possible to determine to what extent the standard can contribute to solving problems of allocation in the case of low capacity utilization (section **Error! Reference source not found.**).

3.2.3 Comparison of functionally similar software products

As software products are usually not functionally equivalent (see section 3.1.1), comparing products that are merely similar to one another may be an alternative. The comparison can then be limited to the functions common to various software products, but it requires a realistic usage scenario.

Examples of such studies on energy consumption of software products in the usage stage include:

- Kern et al. (2011) compared the web browser Mozilla Firefox with Microsoft Internet Explorer, and Open Office Writer with Microsoft Word.
- Commissioned by Microsoft, Roth et al. (2013) compared the web browsers Mozilla Firefox, Google Chrome, and Microsoft Internet Explorer, and arrived at a different result than Kern et al. (2011).

- Williams and Tang (2013) compared the Microsoft products Outlook, Excel, and Word with their web-based equivalents from Office 365 running in the cloud (see also section 2.1.4).

Functional similarity is greatest when different configurations of the same product are compared. This can reveal relevant energy-saving potentials, as Dick et al. (2011) showed using the example of a content management system: a comparison of configurations with and without a hard disk cache showed that energy consumption was significantly lower in the former case.

3.2.4 Comparing a software product with itself over time

Another possibility is to compare different releases or versions of the same software product with each other in terms of resource use.

If a newer version requires greater hardware capacity (e.g. memory, CPU performance, network bandwidth) than an older one, this creates an incentive to replace operational hardware. If analysis focuses on this obsolescence effect, then measurement and allocation problems are partially alleviated, as it is easier to determine the use of hardware capacity than the fraction of energy consumption by hardware resulting from that use.

However, the older and newer software versions are usually not functionally equivalent, as new versions often fulfill additional functional requirements.

In their comparison of three versions of the Microsoft Windows operating system and the corresponding versions of Microsoft Word, Hilty et al. (2006) determined that Windows NT, which was still new at the time of the study, required more total processing time than Windows 2000 to carry out the same functions, even if it was running on hardware that was twice as powerful.[22]

The test was based on file management and word processing tasks performed by 42 subjects. Execution time for the tasks and processor usage were measured as they were carrying out the tasks. Manufacturers generally justify the fact that new software versions inevitably overcompensate for increasing hardware performance by pointing to the new requirements to be fulfilled by the software. However, it cannot be proven that this relationship is inescapable, and resource efficiency itself could be treated as a high-priority requirement (see also the following section 3.2.5).

3.2.5 Green software engineering

A further approach for a solution lies in expanding the perspective of measurable characteristics of the software product to include the process that generates and maintains the product. Then, the question is no longer “What is green software?” but “What is green software engineering?”

Software developers are accustomed to thinking in terms of requirements and developing systems that fulfill them. They differentiate between functional requirements (that describe what the system to be developed is supposed to do) and non-functional requirements that describe qualitative aspects of the system. Typical non-functional requirements in software engineering include security, serviceability, and expandability. As non-functional

[22] This example confirmed Wirth’s Law: “Software is getting slower more rapidly than hardware becomes faster” (as cited in Grosskop & Visser 2013; p. 101).

requirements are characteristics of software architecture, they must be taken into account even in the early stages of software development, and they influence the entire process. At a later stage, it is easier to add functions than to fulfill non-functional requirements.

If the task is to successfully develop “green” software products, then two problems must be solved:

1. Defining non-functional requirements that specify what “green,” “energy efficient”, “resource conserving,” or more generally “sustainable” mean in terms of qualitative characteristics of a software product.
2. Developers taking these requirements into account across the entire process of software development.

Then it would be possible to label software products as to whether they are the result of a process that takes these requirements into account, or to certify the process as such.

1. Definition of non-functional requirements for “green” or sustainable software:

Initial ideas for non-functional requirements are formulated here, based on the previous chapters and the literature cited:

- Demand adaptivity (as described in section 2.1.5)
- New versions do not make greater demands concerning memory, CPU performance, and bandwidth unless absolutely necessary for additional functions.
- Basic functions can still be executed on older hardware in the long term.
- User-oriented configuration options for energy-saving modes.
- Power awareness, optimum management of hardware concerning energy consumption; server software should also take the energy used by the client into account and should in no case hinder or discourage turning off end-user devices or local power management.
- “Power-down-friendliness”: software should not animate people to leave hardware turned on all the time.
- Support for data formats that are economical in terms of bandwidth and memory.
- Support for open standards for data formats (no customer lock-in via formats).
- Flexibility in terms of useable peripheral equipment (minimizing requirements to purchase new equipment).
- Undesired advertising can be turned off.

2. Process-oriented aspects of green software engineering

Naumann et al. (2011, 2013) developed the reference model GREENSOFT for green software engineering. Covering the entire software life cycle, it provides for periodic “sustainability reviews and previews” during the development stage (Dick 2010a) or using the approach of “continuous integration” to conduct energy efficiency measurements even during the development process (Dick et al., 2013), among other things.

Manuals and checklists for developers are mentioned frequently as a tool to support green software engineering, but to date, concrete suggestions are few. Dick et al. (2010b, 2010c) formulated guidelines for “green web engineering,” Microsoft (2010) gave practical power management tips for programming applications on Windows platforms.

Consolidation and further development of guidelines for developers on the basis of a catalogue of non-functional requirements has yet to occur. The new workshop series “Energy Aware Software Engineering and Development” collected existing manuals and checklists in order to make them available to the public (EASED 2013).

4 Recommendations for action

In this section, recommendations for action are formulated whose goal is to minimize use of natural resources caused indirectly by software products.

4.1 Need for research and standardization

4.1.1 Development of methods and standards

The need for action is greatest at the interface between independent application-oriented research on one side and standardization on the other. Here, it is the task of research to analyze and solve methodological problems. Then, it is the task of standardization to operationalize science-based methods for practical use and to create a basis for comparability when they are implemented. Two measures to this end are recommended in the following.

Standardized usage scenarios as the basis for software tests

Usage scenarios describe a typical workflow of using a software function, forming an important basis for defining load profiles and benchmarks when conducting comparison tests. Such scenarios are needed to compare not only different software products, but also different versions and configurations of the same product. When developing standardized usage scenarios, one can build upon initial existing approaches (section 3.2.2).

Definition of non-functional requirements for sustainable software

There is a need for research to systematically develop qualitative criteria that define the idea of sustainable software more precisely. These criteria are to be understood as non-functional requirements for software engineering.

At the same time as these requirements are being developed, it is necessary to create guides or tools for practical work that support incorporating them during the process of software development, including additions to existing procedural models, best-practice guidelines, and checklists in particular.

4.1.2 Periodic data collection

Periodic data collection in the two following areas is recommended for monitoring purposes as long as the current growth dynamics persist.

Monitoring energy consumption by cellular networks and its causes

If, as is to be expected, energy consumption grows rapidly because of the increase in mobile Internet access, measures to promote pricing appropriate to causation in the realm of ICT should be considered in general (e.g. no flat rates for especially energy-intensive or especially CO_2-intensive data traffic).

Monitoring the market for web- and cloud-based applications in terms of resource efficiency

Periodic screening could help identify particularly favorable and unfavorable products and configurations, whereby it is important to take resource use through the entire system providing the service into account. Such screening serves to continuously update the configuration recommendations to users (see section 4.2.2).

4.2 Consumer-oriented measures

4.2.1 Awarding the “Blue Angel” environmental label for software

Although it is true that significant methodological problems are yet to be solved on the way to a “Blue Angel” environmental label for software (see Need for research and standardization, section 4.1), sub-areas can be identified which would be suitable for such a measure in the medium term, namely:

- traditional websites
- web-based applications.

These products require estimating resource use by networks and servers for usage scenarios yet to be defined.

It is difficult to define quantifiable product characteristics for a “Blue Angel” label for locally installed software products as the diversity of functions and the variability of forms of usage are too great. For this reason, qualitative product-related criteria, such as demand adaptivity, support for resource-saving data formats, and energy management on the end-user device are to be considered as well (see also sections 3.2.5 and 4.1.1).

In addition, application of process-oriented criteria for software development is recommended, for example following (yet to be developed) best-practice guidelines for sustainable software development (section 3.2.5).

4.2.2 Providing information

Because of the existing methodological difficulties, it is very difficult to provide objective and robust recommendations for end users of software based on the current state of research: recommendations for or against certain products or types of products would be valid only under severely limiting conditions which would be difficult to communicate due to their complexity.

Recommendations concerning resource-efficient configuration of common software products are an exception. In this case, it would not be products, but configurations of the same product that would be compared. The German Federal Environment Agency could harvest the low hanging fruit in ICT resource efficiency by developing and publishing such recommendations.

Developing and periodically updating configuration recommendations

The following approach is recommended:

1. Selecting a common software product (or a common combination of application and systems software) that can be configured in order to optimize resource efficiency.

2. Conducting comparison tests of the configurations of the software product (same product, same version, different configurations) to measure the hardware capacities used and the electricity consumed.
3. Identifying settings that significantly affect the measured variables. Potentially existing recommendations by manufacturers or user communities could be tested here.
4. Developing or updating guides for users and system administrators that recommend energy-saving settings, for example.

The goal of this measure is also to exploit in practice the characteristics demanded of software products, such as demand adaptivity and energy management, in order to actually conserve resources.

4.3 Recommendations for software developers

4.3.1 Manuals, best-practice guidelines, checklists

Promising approaches such as the GREENSOFT model (Naumann et al. 2011) notwithstanding, the current state of research does not yet permit formulation of tools such as manuals, best-practice guidelines, or checklists for software development.

However, there are numerous tips and practices for energy-efficient programming at least in one specific area: the development of apps for mobile end-user devices, as maximizing battery service life is an important requirement here. Accordingly, the only factor taken into account is local energy consumption. The new workshop series EASED (Energy Aware Software-Engineering and Development) consolidates and publishes the knowledge generated here, which is transferable to other areas of software development to some limited extent. Software developers are advised to follow the activities of the EASED workshops as well as research in the field of green software engineering in general and to participate actively in expanding the body of knowledge and experience.

4.4 Training and professional development

4.4.1 Teaching materials on resource aspects of software architectures for computer technology education

The development of teaching materials on the topic of resource use by software is recommended with the goal of including these subjects in computer science and information systems programs at academic universities, universities of applied sciences, and in vocational training for IT specialists. Other relevant study programs include image processing and computer visualization, media informatics, web design, game development, and game design.

4.4.2 Professional development on resource aspects of ICT use for businesses and public institutions

Businesses and public institutions can make significant contributions to resource conservation by undertaking measures in the field of software, and can reduce costs as well.

Because of the large number of local servers operated by small and medium-sized enterprises and schools (as small to medium-sized organizations operating ICT; see section 4.4.3, however), their potential to save energy and resources is large.

Development of appropriate teaching materials for professional development in businesses and public institutions is recommended, and they should be integrated in existing professional development programs.

Including federal agencies in these efforts is recommended as well because of their considerable equipment needs and their purchasing power in the ICT market.

4.4.3 Support for computer science teachers at schools

When teaching computer science in schools, it is important in general to focus on principles of computer science and not to teach specialized topics that quickly go out of date. This must also be taken into account when integrating environmental and resource-oriented topics in curricula for computer science (or other subjects relating to ICT).

4.4.4 Competitions for resource-efficient software

Annual competitions in which students and developers working in the field can submit resource-saving improvements to existing open source software products are recommended. They would permit any developer to participate even without comprehensive specialized knowledge about the area in which a software product is to be used and to concentrate, for example, on individual algorithms in a module.

In addition to raising awareness on the part of the participants and the interested public about resource questions relating to software, good solutions could be disseminated quickly because of the openness of OSS and could also be taken on in other software products.

Bibliography

AC4DC (n.d.): Bisher unveröffentlichte Angaben aus dem Projekt AC4DC, an dem die Autoren der vorliegenden Studie beteiligt sind, http://www.AC4DC.de; last access 24.07.2013

Accenture (n.d.): Application Services for Open Source Software: Service-Überblick. http://www.accenture.com/de-de/Pages/service-technology-systems-integration-open-source-overview.aspx .last access 29.6.2013

Amsel, N., Tomlinson, B. (2010): Green Tracker: a tool for estimating the energy consump-tion of software. In: CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems. ACM, New York, S. 3337–3342

Baumeister, J. (2012): Server-Management-Suites: Management-Software der Server-Hersteller im Vergleich. In Tecchannel, 18.11.2009. http://www.tecchannel.de/server/hardware/2023817/server_management_software_das_bieten_die_besten_verwaltungs_suites/index6.html last access 31.05.2013.

Bayer, M. (2009): Hardware, Software und Prozesse: 11 Ratschläge, Storage billiger zu machen. In: CIP, 04.05.2009. http://www.cio.de/knowledgecenter/storage/887974 last access 17.12.2012.

BITKOM (2010b): Green-IT-Allianz, Ergebnisstand AG3 - Software und Green IT. Internes Arbeitspapier.

BITKOM (n.d.): Open Source Software: Rechtliche Grundlagen und Hinweise. Leitfaden (Version 1.0). http://www.bitkom.org/files/documents/bitkom_publikation_oss_version_1.0.pdf last access 29.6.2013.

BMU (2012): Ressourcen schonende Speicherlösung für Rechenzentren. http://www.bmu.de/bmu/presse-reden/pressemitteilungen/pm/artikel/ressourcen-schonende-speicherloesung-fuer-rechenzentren/ last access 14.01.2013.

Bokhari, S. H., Rehman, R. (1999): Linux and the Developing World, IEEE Software 16(1), S. 58-64.

Bozzelli, P., Gu, Q., Lago, P. (2013): A systematic literature review on green software metrics. Technical report.

BPMN (2013): Business Process Modelling and Notation. Hompeage. http://www.bpmn.org/ last access 20.6.2013.

Bräuner, H. (2005): Linux im Rathaus – Ein Migrationsprojekt der Stadt Schwäbisch Hall, in (Bärwolff et al., 2005), S. 37-50, http://www.opensourcejahrbuch.de/download/jb2005/ last access 30.6.2013.

Chen, F., Schneider, J.-G., Yang, Y., Grundy, J., He, Q. (2012): An Energy Consumption Model and Analysis Tool for Cloud Computing Environments. 1st ICSE Workshop on Green and Sustainable Software (GREENS 2012), Zurich, Switzerland, 3rd June 2012, S. 45-50.

Coroama, V., Hilty, L. M., Heiri, E., Horn, F. (2013): The Direct Energy Demand of Internet Data Flows. Journal of Industrial Ecology. DOI: 10.1111/jiec.12048

Creber, C. (2004): Die Bedeutung von Open Source in der Geschäftsstrategie von IBM, in (Picot und Doeblin, 2004), S. 111-116.

Dick, M., Drangmeister, J., Kern, E., Naumann, S. (2013): Green Software Engineering with Agile Methods. 2nd ICSE Workshop on Green and Sustainable Software (GREENS 2013), San Francisco, CA, USA, May 20, 2013.

Dick, M., Kern, E., Drangmeister, J., Naumann, S., Johann, T. (2011): Measurement and Rating of Softwareinduced 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, S. 290–299.

Dick, M., Naumann, S. (2010a): Enhancing software engineering processes towards sustainable software product design, in: K. Greve, A.B. Cremers (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. Shaker, Aachen, 2010, S. 706–715.

Dick, M., Naumann, S., Held, A. (2010b): 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: Filipe, J., Cordeiro, J. (Hrsg.). WEBIST 2010 - Proceedings of the Sixth International Conference on Web Information Systems and Technologies, Volume 1, Valencia, Spain, April 07-10, 2010, 2 volumes, INSTICC Press, Setúbal, S. 48–55.

Dick, M., Naumann, S., Kuhn, N. (2010c): A Model and Selected Instances of Green and Sustainable Software. In Berleur et al. (2010) S. 248-259.

Diedrich, O. (4.2.2009): Trendstudie Open Source. Wie Open-Source-Software in Deutschland eingesetzt wird. In Heise Open Source. http://heise.de/-221696 . last access 29.6.2013.

Drenkelfort, G., Pröhl, T., Erek, K. (2013): Energiemonitoring von IKT-Systemen. Kennzahlen. Projektberichte IKM| 3. Universitätsverlag der TU Berlin.

EASED (2013): 2nd Workshop EASED@BUIS 2013 – Energy Aware Software-Engineering and Development – Proceedings. Bunse, C., Gottschalk, M., Naumann, S., Winter, A. (Hrsg.). Carl von Ossietzky Universität Oldenburg. OLNSE Number 4/2013.

Ehmann, S., Hintemann, R. (2004): Leitfaden zum Thema Information Lifecycle Management. In Competence Site. http://www.competence-site.de/location-intelligence-geomarketing-gis/Leitfaden-zum-Thema-Information-Lifecycle-Management last access 17.12.2012.

EMC (2013): Deduplication Solutions. http://germany.emc.com/backup-and-recovery/deduplication.html last access 10.05.2013.

Emmenegger, M. F., Frischknecht, R., Stutz, M. (2006): Life Cycle Assessment of the mobile communication system UMTS – towards eco-efficient systems, International Journal of Life Cycle Assessment, 11, S.265-276.

Enterprise Management Associates: 2006, EMA study: Get the Truth on Linux Management, Website, http://www.thalix.com/files/EMA_Levanta-Linux_RR.pdf last access 1.7.2013.

Erek, K., Drenkelfort, G., Pröhl, T. (2013): Energiemonitoring von IKT-Systemen. State-of-the-Art von Energiemonitoringsystemen. Projektberichte IKM| 2. Universitätsverlag der TU Berlin.

Ernst&Young (2011): Open Source Software im geschäftskritsichen Einsatz. http://www.ey.com/Publication/vwLUAssets/Open_Source_Software_im_geschaeftskritischen_Einsatz/$FILE/Open_Source_Software_DE.pdf . last access 29.6.2013.

Gartner (4. Mai, 2013): Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013. Pressemitteilung. http://www.gartner.com/newsroom/id/2482816 last access 18.07.2013.

Geer, D. (2008): Reducing the Storage Burden via Data Deduplication. In Computer, Volume 41, Issue 12, December 2008, S. 15-17.

GHGP (2013a): GHG Protocol Product Life Cycle Accounting and Reporting Standard ICT Sector Guidance, 26. Januar 2013, http://www.ghgprotocol.org/feature/ghg-protocol-product-life-cycle-accounting-and-reporting-standard-ict-sector-guidance

GHGP (2013b): GHG Protocol Product Life Cycle Accounting and Reporting Standard ICT Sector Guidance, Chapter 1, Introduction and General Principles, Draft v1.3, 26. Januar 2013, http://www.ghgprotocol.org/files/ghgp/GHGP-ICT-Introduction-Chapter-v1-3-26-JAN-2013.pdf

GHGP (2013c): GHG Protocol Product Life Cycle Accounting and Reporting Standard ICT Sector Guidance , Chapter 7, Software, Draft v2.9, http://www.ghgprotocol.org/files/ghgp/GHGP-ICT-Software-v2-9-26JAN2013.pdf

Grosskop, K., Visser, J. (2013): Identification of Application-level Energy Optimizations. In Hilty et al. (2013), S. 101-107.

He, Q., Li. Z., Zhang, X. (2010): Data deduplication techniques. In Future Information Technology and Management Engineering (FITME), 2010 International Conference on (Volume 1). Date of Conference: 9-10 Oct. 2010, S. 430-433.

Heinrich, H., Holl, F.-L.. Menzel, K., Mühlberg, J. T., Schäfer, T., Schüngel, H. (2006): Metastudie: Open-Source-Software und ihre Bedeutung für Innovatives Handeln. In Holl, F.-L. (Hrsg.) Entwicklungen in den Informations- und Kommunikationstechnologien. Band 1. http://www.bmbf.de/pubRD/oss_studie.pdf. last access 28.6.2013.

Heise Online (2011b): Nokia forciert ortsbezogene Anwendungen für Windows Phone 7. In Heise Online. http://heise.de/-1272416 last access 31.05.2013.

Heise Online (2012): Studie: Open-Source-Software qualitativ besser als proprietäre Entwicklungen. In HeiseOnline. http://www.heise.de/developer/meldung/Studie-Open-Source-Software-qualitativ-besser-als-proprietaere-Entwicklungen-1440788.html last access 4.3. 2012.

Heise Open Source (2008): Open Source ist überall. In Heise Online. http://heise.de/-217214 last access 29.6.2013.

Heise Open Source (2012a): Immer mehr Linux auf dem Desktop. In HeiseOnline. http://www.heise.de/open/meldung/Immer-mehr-Linux-auf-dem-Desktop-1404775.html last access 3.3.2012.

Heise Open Source (2012b): Anteil der Freizeit-Kernel-Hacker sinkt. In Heise Online. http://heise.de/-1500629 last access 29.6.2013.

Hilty, L. M., Aebischer, B., Andersson, G., Lohmann, W. (Hrsg.) (2013): ICT4S – ICT for Sustainability. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16, 2013. ETH. http://e-collection.library.ethz.ch/eserv/eth:6558/eth-6558-01.pdf last access 31.05.2013.

Hilty, L. M., Köhler, A., von Schéele, F., Zah, R., Ruddy, T. (2006): Rebound Effects of Progress in Information Technology. Poiesis & Praxis: International Journal of Technology Assessment and Ethics of Science, 1 (4), S. 19-38.

Hilty, L.M., Lohmann, W. (2011): The Five Most Neglected Issues in "Green IT" In CEPIS UPGRADE 12: 4, S. 12-15.

Hilty, L.M., Oertel, B., Wölk, M., Pärli, K. (2012): Lokalisiert und identifiziert. Wie Ortungstechnologien unser Leben verändern. TA-Swiss. ISBN 978-3-7281-3460-8.

Hintemann, R., Fichter, K. (2012): Energieverbrauch und Energiekosten von Servern und Rechenzentren in Deutschland - Aktuelle Trends und Einsparpotenziale bis 2015, Berlin. www.borderstep.de last access 31.05.2013.

Hintemann, R., Fichter, K. (2013): Server und Rechenzentren in Deutschland im Jahr 2012, Berlin 2013. http://www.borderstep.de/pdf/Kurzbericht_Rechenzentren_in_Deutschland_2012__09_04_2013.pdf last access 14.05.2013.

ITU (2012a): Assessment Framework for Environmental Impacts of the ICT Sector (September 2012), http://www.itu.int/dms_pub/itu-t/oth/4B/04/T4B0400000B0008PDFE.pdf, last access 12. Mai 2013.

ITU (2012b): Toolkit on environmental sustainability for the ICT sector (September 2012), http://www.itu.int/dms_pub/itu-t/oth/4B/01/T4B010000060001PDFE.pdf last access 12. Mai 2013.

ITU (2012c): L.1410: Methodology for the assessment of the environmental impact of information and communication technology goods, networks and services, http://www.itu.int/rec/T-REC-L.1410-201203-I/en , März 2012, last access 16. Mai 2013.

Kern, E., Dick, M., Johann, T., Naumann, S. (2011): Green Software and Green IT: An End Users Perspective. In Golinska, P., Fertsch, M., Marx-Gomez J. (Hrsg.). Information Technologies in Environmental Engineering Environmental Science and Engineering. New Trends and Challenges. Berlin, Heidelberg: Springer, S. 199-211. http://link.springer.com/book/10.1007/978-3-642-19536-5

Kern, E., Dick, M., Naumann, S., Guldner, A., Johann, T. (2013): Green Software and Green Software Engineering – Definitions, Measurements, and Quality Aspects. In: Hilty et al. (2013), S. 87-94.

Lanline (2012): Ganzheitliches IT- und Facility-Management optimiert Leistung im RZ. In Lanline (online), 25.10.2012. http://www.lanline.de/fachartikel/ganzheitliches-it-und-facility-management-optimiert-leistung-im-rz.html last access 18.12.2012.

Masanet, E., Shehabi, A, Ramakrishnan, L., Liang, J., Ma, X., Walker, B., Hendrix, V., Mantha, P. (2013): The Energy Efficiency Potential of Cloud-Based Software: A U.S. Case Study. Lawrence Berkeley National Laboratory, Berkeley, California. http://crd.lbl.gov/assets/pubs_presos/ACS/cloud_efficiency_study.pdf; last access 20.8.2013

Meyer, D.T., Bolosky, W.J. (2012): A study of practical deduplication. In Transactions on Storage (TOS), Volume 7 Issue 4. January 2012, Article No. 14. http://delivery.acm.org/10.1145/2080000/2078864/a14-meyer.pdf?ip=130.60.155.223&acc=ACTIVE%20SERVICE&key=C2716FEBFA981EF1DD5891E815377FF4A1D6D263689CD484&CFID=219314061&CFTOKEN=74874759&__acm__=1369323403_e879f7afe03eda700773f4fa34cf0fce last access 31.05.13.

Microsoft (2010): Energy Smart Software.http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Energy-Smart_SW.mspx last access 20.6.2013.

Müller, K. (2013): Persönliches Gespräch von R. Hintemann mit K. Müller zu Softwarelösungen im Rechenzentrum am 11.01.2013.

Naumann, S. (2013): Telefonisches Gespräch von L. M. Hilty mit Stefan Naumann, Professor an der Hochschule Trier, am 13.05.2013.

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 1 (2011), S. 294-304.

Naumann, S., Gresk, S., Schäfer, K. (2008): How green is the web? Visualizing the power quality of websites, in: A. Möller, B. Page, M. Schreiber (Hrsg.), Environmental Informatics and Industrial Ecology, 22nd International Conference on Informatics for Environmental Protection, EnviroInfo 2008, 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, Shaker, Aachen, 2008, S. 62–65.

Naumann, S., Kern, E., Dick, M. (2013): Classifying Green Software Engineering - The GREENSOFT Model. In Bunse, C., Gottschalk, M., Naumann, S., Winter, A. (Hrsg.): Proceedings of the 2nd Workshop Energy Aware Software-Engineering and Development (EASED@BUIS). OLNSE Number 4/2013, S. 13-14. http://www.se.uni-oldenburg.de/documents/olnse-4-2013-eased.pdf last access 31.05.2013.

Nebel, W., Hoyer, M., Schröder, K., Schlitt, D. (2009): Untersuchung des Potentials von rechenzentrenübergreifendem Lastmanagement zur Reduzierung des Energieverbrauchs in der IKT, Studie für das Bundesministerium für Wirtschaft und Technologie, OFFIS, Dezember 2009.

Paulson, J. W., Succi, G., Eberlein, A. (2004): An Empirical Study of Open-Source and Closed-Source Software Products, IEEE Transactions on Software Engineering 30(4), S. 246-256.

Pelkmann, T. (2010): Deduplizierung - wie geht das? In Computerwoche online, 29.11.2010. http://www.computerwoche.de/a/deduplizierung-wie-geht-das,2358569 last access 17.12.2012.

Powertop (n.d.): https://01.org/powertop/ last access 18.07.2013.

Rebitzer, G., Ekvall, T., Frischknecht, R., Hunkeler, D., Norris. G., Rydberg, T., Schmidt, W.-P., Suh, S., Weidema, B.P., Pennington, D.W. (2004): Life cycle assessment Part 1: Framework, goal and scope definition, inventory analysis,and applications. Environment International. 30(2004), pp. 701-720.

Reder, B. (2012): Data Center Infrastructure Management - Mit DCIM das RZ in den Griff bekommen. In Computerwoche online, 08.10.2012. http://www.computerwoche.de/a/mit-dcim-das-rz-in-den-griff-bekommen,2514482 last access 20.12.2012.

Renner, T., Vetter, M., Rex, S., Kett, H. (2005): Open Source Software. Einsatzpotenziale und Wirtschaftlichkeit. Eine Studie der Fraunhofer-Gesellschaft. Fraunhofer IRB Verlag, Stuttgart und Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO, Stuttgart. http://wiki.iao.fraunhofer.de/index.php/Open_Source_Software last access 28.6.2013.

Rieß, U (2012): Effizienz ohne Deduplizierung? In speicherguide.de, 01.06.2012. http://www.speicherguide.de/backup-recovery/disk-backup/effizienz-ohne-deduplizierung-15576.aspx last access 17.12.2012.

Roth, K., Patel, S., Perkinson, J. (2013): The Impact of Internet Browsers on Computer Energy Consumption. Final Report to Microsoft. Fraunhofer Center for Sustainable Energy Systems.

Rüdiger, A. (2011): Storage-Kosten: Zehn Tipps zum Sparen beim Speichern. In Computerwoche online, 26.03.2011. http://www.computerwoche.de/a/zehn-tipps-zum-sparen-beim-speichern,1911740 last access 17.12.2012.

Scharnhorst, W., Hilty, L. M, Jolliet, O. (2006): Life cycle assessment of second generation (2G) and third generation (3G) mobile phone networks ENVIRONMENT INTERNATIONAL 32: 5. 656-675 JUL

SourceForge (n.d.): http://sourceforge.net/ last access 18.07.2013.

Stephens, A., Didden, M. (2013): The Development of ICT Sector Guidance: Rationale, Development and Outcomes. In Hilty et al. (2013), pp. 8-11.

Stobbe, L., Nissen, N. F., Proske, M., Middendorf, A., Schlomann, B., Friedewald, M., Georgieff, P., Leimbach, T. (2009): Abschätzung des Energiebedarfs der weiteren Entwicklung der Informationsgesellschaft Bearbeitungsnummer I D 4 - 02 08 15 - 43/08, Abschlussbericht an das Bundesministerium für Wirtschaft und

Technologie. Berlin: Fraunhofer IZM, 2009, 164 pp. urn:nbn:de:0011-n-1102312. http://publica.fraunhofer.de/eprints/urn:nbn:de:0011-n-1102312.pdf last access 31.05.2013.

Thoma, J. (2012a): Ubuntu 12.04 mit aktivierter Energiesparoption. In Golem.de. http://www.golem.de/news/sandy-bridge-ubuntu-12-04-mit-aktivierter-energiesparoption-1202-89883.html last access 3.3.2012.

Thoma, J. (2012b): Stromsparen nur mit Handarbeit. In Golem.de. http://www.golem.de/news/test-asus-zenbook-mit-linux-stromsparen-nur-mit-handarbeit-1202-89934.html last access 3.3.2012.

Unterseher, D. (2008): Data-Deduplication, Vortrag auf dem BITKOM-Arbeitskreis Speichertechnologien am 27.11.2008. http://www.bitkom.org/files/documents/Unterseher_Data_Deduplication_2008_11_27d.pdf last access 16.12.2012.

Van Bokhoven, F., Bloem, J. (2013): Pilot result Monitoring Energy usage by Software. In Hilty et al. (2013), pp. 108-115.

Vilsbeck, C. (2012): Virtualisierung, Cloud, Big Data, SSD: Storage-Trends: Speichertechnologien für 2012. In Tecchannel, 30.01.2012. http://www.tecchannel.de/storage/management/2038683/storage_trends_speichertechnologien_2012_virtualisierung_cloud_big_data_ssd/ last access 31.05.2013.

Wikipedia (2012a): Seite „Open Source“. In: Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/wiki/Open_Source last access 30. März 2012

Wikipedia (2012b): Seite „Freie Software“. In: Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/wiki/Freie_Software last access 30. März 2012

Wikipedia (2013a): Seite „Fahrzyklus“. In: Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/wiki/Fahrzyklus last access 30. Juni 2013

Wikipedia (2013b): Seite "Open Source Software in öffentlichen Einrichtungen". In Wikipedia. http://de.wikipedia.org/wiki/Open-Source-Software_in_%C3%B6ffentlichen_Einrichtungen last access 29.6.2013.

Wilde, H. (2013): Persönliches Gespräch von R. Hintemann mit H. Wilde zu Deduplizierungslösungen im Rechenzentrum am 14.01.2013.

Wilke, C. (2012): Energy Labels for Mobile Applications. Fakultät Informatik, Institut für Software- und Multimediatechnik. http://www.claaswilke.de/publications/workshops/EEbS2012.pdf last access 31.05.2013.

Williams, D. R., Tang, Y. (2013): Impact of Office Productivity Cloud Computing on Energy Consumption and Greenhouse Gas Emissions. In Environmental Science & Techology 47 (9), pp. 4333-4340.

Zapico, J. L., Turpeinen, M., Brandt, N. (2010): Greenalytics: a tool for mash-up life cycle assessment of websites. In: Proceedings of the 24th International Conference on Informatics for Environmental Protection (EnviroInfo 2010). Cologne/Bonn, Germany, Shaker, Aachen, 2010, pp. 754-763. ISBN: 978-3-8322-9458-8.