A Model and Selected Instances of Green and Sustainable Software

Markus Dick, Stefan Naumann, Norbert Kuhn

A Model and Selected Instances of Green and Sustainable Software

Markus Dick, Stefan Naumann and Norbert Kuhn

Trier University of Applied Sciences, Umwelt-Campus Birkenfeld, Campusallee,
D-55768 Hoppstädten-Weiersbach, Germany
m.dick@umwelt-campus.de, s.naumann@umwelt-campus.de, n.kuhn@umwelt-campus.de

Abstract. The power consumption of ICT is still increasing. To date it is not clear if the energy savings through ICT overbalance the energy consumption by ICT or not. Where manifold efforts of Green IT address the environmental aspects of sustainability considering computer hardware, there is a lack of models, descriptions, or realizations in the area of computer software. In this paper we present some results that form the basis towards a definition of the term "Sustainable Software", an outline of a process and lifecycle model for "Green and Sustainable Software Engineering", and six concrete instances of this model for practitioners.

Keywords: Sustainable Software, Green Software Engineering, Green IT, Sustainable Development.

1 Introduction

It is well known that global warming, greenhouse gas (GHG) effects, climate change and sustainable development (SD) are key challenges of the 21st century [1]. Information and Communication Technology (ICT) takes an important role within these challenges. On the one hand, ICT can optimize material flows and therefore reduces energy consumption [2]. But ICT itself is consuming more and more energy: the energy consumption of IT especially that of the Web is still increasing [3]. E.g. the estimated power consumption of data centres in the U.S. was increasing from 28 billion kWh in 2000 to 61 billion kWh in the year 2006 [4] and in the world from 58 billion kWh in 2000 to 123 billion kWh in 2005 [5].

Up to now, several publications have examined the relationship between the field of sustainability and ICT. They discuss the impact of ICT on the environment [6] or consider the balance between energy savings and energy consumptions by ICT [7]. Especially, to date it is not clear if energy consumption by ICT is greater or smaller than energy savings by ICT, e.g. via more efficient processes or simulation of scenarios.

In our paper we present an approach for a model of green and sustainable software that tries to handle both challenges: Reducing the energy consumption of ICT itself and using ICT to contribute to other goals of "Sustainable Development". From a

theoretical point of view, our approach can be classified into a new research field called "Sustainability Informatics" [8].

2 What is Sustainable Software?

As a first step on our way to a general definition of the term “Sustainable Software”, we interviewed five lecturers and practitioners of computer science and asked them about their understanding of Sustainable Software. Of course, this sample size is too small for a substantial analysis, however it gives some good indications of what Sustainable Software might be. The answers lead to the following categories that sharpen the picture towards a definition: software design and programming, need-based development, project management, special features, possible applications, and resource saving.

The category “software design and programming” subsumes software quality aspects that comply with software quality needs according to ISO/IEC 25000 [9], like e.g. portability, reusability, extensibility, adaptability, modularization, documentation, or readable source code which makes future extensions of software products easier and less error-prone, but also an expected long physical life time. Modularization depends directly on the need to deliver only those functions that are necessary for a user. Another point is the possibility of making software products comparable with respect to sustainability e.g. power consumption or resource efficiency during runtime. This may be accomplished through an eco-label similar to the EU energy label [10].

The category “need-based development” subsumes efficiency criteria like processing time, memory requirements and network load which are also quality needs as already mentioned above. Some interviewees addressed the function-richness of modern standard software like e.g. text processors. Standard users rarely tap the full potential of such applications. Thus, sustainable software should only be delivered with functions that are used by a specific user.

The category “project management” addresses distributed software development, supporting paperless offices with document management systems or pursuing company visions that touch sustainability.

The category “special features” contains items like smart techniques and drivers, ergonomic user adjustments or even health promoting interventions. Smart techniques and drivers should anticipate intelligent (and therefore sustainable) user behaviour. Examples are device drivers that do not switch devices on prior to their usage, or printer drivers that ask if two pages should be printed on one sheet of paper when printing in concept or economy mode. An example for a health promoting intervention may be a program that invites users to perform gymnastic exercises when they worked nonstop for some time.

The category “possible applications” addresses the visualization of energy and resource consumption, the reduction of data redundancy, or software that supports the reduction of power and resource consumption indirectly.

Resources can be saved directly and indirectly. The corresponding answers are subsumed under the category “resource saving”. Direct effects mentioned are: alterna-

tive data mediums, lower transport and retail packaging efforts or low hardware requirements, so that the software runs even on hardware that is out of date. This means, that new software products should not lead to the need to replace existing hardware with newer hardware, because of the overall recycling (old hardware) and production (new hardware) expenditure. Indirect effects mentioned are: socially acceptable working conditions, more efficient route planning systems, and more efficient warehouse management systems. More efficient route planning systems can result in lower transport rates and distances and therefore save fossil energy in form of e.g. fuel. Efficient warehouse management systems can lead to smaller warehouses and thus result in fewer sealed surfaces.

These results show that sustainable software focuses not only on a reduction of the consumption of natural resources and energy in the sense of the environmental dimension of sustainability, but also on the social and economic dimensions. This leads to a definition of Sustainable Software as follows:

Definition 1: Sustainable Software is software whose direct and indirect negative impacts on economy, society, human beings, and the environment resulting from development, deployment, and usage of the software is minimal and/or has a positive effect on sustainable development.

This definition and some aspects of the interviews are taken into consideration in order to initially form a model for Green Software Engineering as one part of a model for Sustainable Software Engineering.

3 A Model for Green and Sustainable Software Engineering

The model for Green and Sustainable Software Engineering comprises two general components: the process and lifecycle model, and guidelines and checklists. The model (for an overview see Fig. 1) delimits Green and Sustainable Software Engineering from other software engineering methodologies and includes guidelines and checklists. The process model identifies tasks or activities during a software product's lifecycle, which are relevant for the sustainability valuation of the product. The guidelines and checklists support actors with different professional levels in applying green or sustainable techniques in general when developing, administrating or using software products.

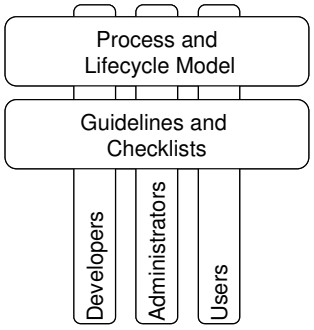

Fig. 1. A model for Green and Sustainable Software Engineering.

3.1 Process and Lifecycle Model

The proposed process and lifecycle model is based upon ISO 12207 ("Software life cycle processes") [11] and ISO 14040 ("Life cycle assessment") [12]. Its objective is to provide starting-points for activities that allow an assessment of the sustainability relevant consequences resulting from the usage of the software product during its whole lifecycle (see Fig. 2). These activities are intended to lead to more sustainable software products. The early draft of the model presented below, focuses mainly on the environmental aspects of sustainability. Extensions to the other pillars of sustainability are planned for the future.

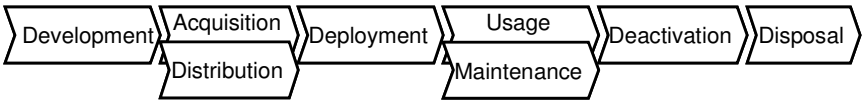

Fig. 2. Process and lifecycle model.

The software development processes applied during the *development* lifecycle phase are extended with continuous reflection meetings and assessment activities. These are targeted at assessing direct and indirect environmental impacts, which result from the software development process itself and are expected to arise from the future use of the software product under development. The outcomes of the assessments and reflection meetings should be used to take action towards more sustainable software products e.g. by increasing resource efficiency (which leads possibly to a reduction of the power consumption during program execution), reducing hardware requirements of the software product, avoiding business trips for meeting the development team, or involving prospective users tightly in the development process.

In the next process phase a distinction is drawn between *acquisition* and *distribution*. Here, acquisition means that one evaluates different standard software products, chooses one that fits the needs best and purchases it from a software retailer. A sus-

tainable software selection process should consider selection criteria that take account of direct and indirect impacts on the environment resulting from use of the software product. The acquisition process part can generally be omitted when developing custom software products, because the sustainability aspects that are corresponding to selection criteria can there be considered as software requirements. The distribution process part is relevant for both standard and custom software. Here, it is necessary to consider whether the software product should be delivered as a download or on a physical data medium. If the software is delivered on a data medium, it is also necessary to decide which transport and sales packaging is necessary and justifiable against the background of ecological sustainability. In making the decision, one has to consider the impacts of the corresponding production chains and also the anticipated impacts of the necessary recycling processes. Even though the impacts of recycling/disposal are already considered in the decisions in this phase, the recycling costs themselves, like resource consumption, are assigned to the disposal phase.

The *deployment* lifecycle phase considers aspects that are relevant for administrators during deployment of software products. This includes: data centre virtualization, configuration of the runtime environment necessary to run the software product and appropriate initial configuration of the software product. These activities do not take effect until the usage phase.

The *usage* and *maintenance* phase considers direct and indirect sustainability effects that evolve from the utilization of the software product. The impact on sustainability that results from this phase, is already considered during the development phase by actors such as the software programmers. In this model, the term "maintenance" does not include the programming activities that are necessary to correct faults after the software product has been delivered. On the contrary, in this context it refers to the support and maintenance offered by administrators to end users, e.g. in organizations which run a data centre. In order to reduce the power consumption of computer systems, users can configure programs, or the energy management of the operating system properly, so that the overall power consumption is minimized. This also includes configuring large caches in web browsers, so that the necessary data transfer, and thus power consumption, is minimized. Another example is the visualization of the energy consumption of working groups, which also has an educational formation effect that is considered to be relevant in developing an awareness of sustainability and ecologically relevant questions. In organizations, where computer users act with low or even no administrational rights on their computer systems, the responsibility to configure the computer systems in an energy conserving way is taken over by the administrators. In organizations that implement service desks according to the IT Infrastructure Library as single points of contact to end users, the mission of the service desks is also to advise the end users proactively [13]. This means that service desks could also advise users proactively on software configurations regarding energy conservation and sustainability issues.

The *deactivation* phase considers aspects that become relevant if software products are taken out of service. An example would be backups of files that have proprietary file formats and can therefore possibly not be processed with newer or competing software products. In this example, one must decide if the old files have to be converted to new data formats, e.g. if copies must be kept due to legal issues, or if the files are obsolete and can therefore be deleted. The deletion of these files releases

backup resources, which can be used for other backup purposes and thus save energy and natural resources, because there is possibly no need to buy new backup storages.

The *disposal* phase takes impacts on sustainability into account that result from the disposal of the data medium and packaging. This includes energy and resource consumption that is necessary for recycling. In this phase, no further actions or considerations take place, because the relevant actions and considerations are already taken in the distribution phase. Hence, only the impacts of disposal and recycling are assigned to this phase.

3.2 Guidelines and Checklists

Guidelines and checklists represent the implementable parts of the model and are therefore a constitutive component of it. At large, they are forming an open knowledge base, which enables the consideration of future developments, trends, and evolving technical expertise. The guidelines and checklists provide tips and helpful hints on how to develop, use, provide, and maintain software products in a sustainable way. Therefore, the guidelines and checklists are aligned to the activities and product scenarios of the lifecycle model described in the preceding section. The knowledgebase should regard the different technical levels of its users according to roles shown in the reference model (see Fig. 1). Appropriate basic roles are: developer, administrator, and user. In order to find the best guidelines and checklists for an actor it is necessary to create specialized roles like: web developer, web author, requirements engineer, software architect etc. as sub roles of the developer role [14].

4 Selected Instances of Guidelines and Software Tools

In the following sections we present six representative instances of our model, realized as software tools or guidelines.

4.1 For Developers

The first two examples are addressing the development lifecycle phase of our model and are thus relevant to developers. Although these examples are applied during the development lifecycle phase, they do not take effect until the usage phase.

Optimization Guideline for Graphical Design Elements in Websites. Today, 54% of an average website is made up of graphics [15]. Hence, optimizing graphical design elements is a promising way to minimize the transferred data volume and thus save power because of less network load. Graphical design elements are, unlike photographs or charts, images which are used to design borders, tabs, buttons, or as a logo identifying a company or an organization. In comparison to photographs or charts, graphical design elements are smaller in size and generally have a limited number of colours. This means that reducing the number of colours within an image

from RGB living colour to a colour-palette with a limited number of colours reduces the filesize significantly (see Table 1). These images should be saved in the Portable Network Graphics (PNG) format, which, in contrast to the JPEG format, supports indexed palettes. Compared to the GIF format, the PNG format has a better compression algorithm, which results in a smaller filesize [15]. Furthermore it is possible to remove text from a logo. The logo without text is then put as a background image behind the text implemented directly with HTML. In the special example given in table 1, the savings that can be achieved by switching from JPEG to PNG with an indexed palette are for both cases (with text and without text) approx. 84%. Furthermore, the removal of the text results in savings of approx. 50% to 60% in cases of PNG and JPEG. The total saving, when converting from JPEG (with text) to PNG (without text) is approx. 92%. So it is recommended to optimize graphical design elements like those described above, in order to reduce the network load and thus power consumption of a website.

Table 1. Filesizes of a graphical design element (see Fig. 3) in different optimization levels (JPEG files at 85% quality level; all values in bytes)

	File format	RGB living colour	12 colour palette	7 colour palette
Filesize with text	JPEG 85%	8,152	n/a	n/a
	GIF	n/a	2,345	2,248
	PNG	9,036	1,320	1,290
Filesize without text	JPEG 85%	3,895	n/a	n/a
	GIF	n/a	1,528	1,431
	PNG	3,425	666	639

Fig. 3. Left: Logo with RGB living colour; Right: Logo with 7 colour palette without text.

Website Optimization Suggestions Integrated in Development Tools. Optimizing the text based source files that make up a website is a promising way to reduce the network load generated by it. One possibility is to minimize CSS files. An exemplary tool that supports minimization is CSSTidy (http://csstidy.sourceforge.net/). It removes unnecessary characters (e.g. whitespaces and line breaks), unnecessary values and attributes (e.g. default values), combines separate single properties to shorthand notations (e.g. margin, border, font), and uses abbreviations. There are many other possibilities to minimize CSS [14].

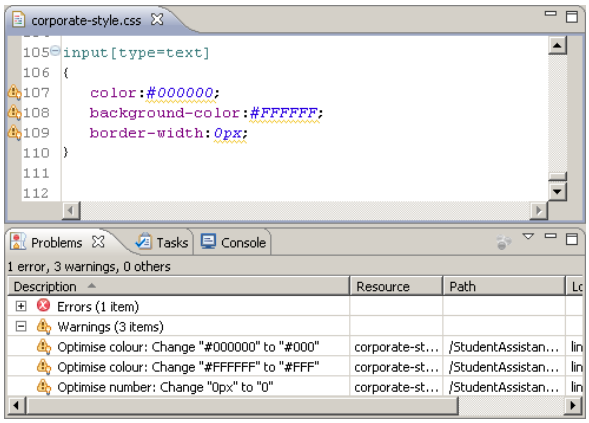

Fig. 4. Integration of network load reducing optimization suggestions for CSS in Eclipse.

For web developers, it is not easy to follow all these different suggestions during development. Hence, appropriate situational assistance tools integrated in development environments are essential to simplify matters. Figure 4 depicts our prototypical integration of the above mentioned tool CSSTidy in the well known integrated development environment Eclipse (http://www.eclipse.org). After saving the CSS file under work, the developer gets optimization feedback as warning hints next to the affected code lines. These hints are additionally listed in the problems view, beneath the editor view. In contrast to the original CSSTidy tool that reformats the code of the processed files, our Eclipse Plug-in prototype uses only the processing log output of CSSTidy to generate its hints. Thus developers are free to decide whether or not they want to follow these hints and hence retain full control of the source code.

4.2 For Administrators

The next two examples address the deployment lifecycle phase of our model and therefore are relevant to administrators. Although, the examples are applied during the deployment phase they take effect to ecological sustainability mainly not until the usage and maintenance phase.

Caching and Compression. Today, approx. 80% of the web users have their browsers configured for caching [16]. Therefore supporting the caching strategies of web browsers is ecologically worthwhile, because this will significantly decrease the amount of transferred data and thus power consumption. Since the configuration of the user's web browser cannot be affected by administrators, they have to focus on the server-side configuration aspects of caching. Caching in HTTP/1.1 is designed to reduce the need to send requests to servers (expiration mechanism) and the need to send full responses back to clients (validation mechanism). The latter does not reduce

the amount of HTTP requests but it reduces the payload of the HTTP responses that are sent back to the client and thus addresses network bandwidth reduction [17].

In addition to caching, modern web browsers usually support some kind of data compression. This reduces response sizes and transfer times, which results in lower power consumption. A web server may compress the content using one of the compression methods that a browser claims to support [17]. On the contrary, compressing content on the fly, depending on the capabilities of the user's web browser, results in higher processor load, which in turn leads to higher power consumption. Hence, compressible files should be stored in a compressed and an uncompressed version.

In order to reduce the total amount of HTTP requests and HTTP payload sizes we suggest that administrators of web servers configure the cache support and the compression support properly, so that power consumption is reduced effectively.

Energy-Saving Software for Hardware and Data Centres. So far, several data centre providers exist that operate their processing services with renewable energy. Additionally, administrators can apply the newest techniques regarding Green IT like virtualization strategies. According to [18] it seems to be possible to save 4.1 megatons of carbon dioxide emissions by virtualization and by optimizing heating, ventilating and air conditioning in German data centres.

Unfortunately, introducing virtualization strategies in data centres carries the risk that persons responsible tend to deploy server based applications like e.g. web services or web based content management systems to distinct virtual machines. One advantage is that these services do not interfere with each other anymore when applying software patches or updates, because they run on separate systems, which makes them at a first glance easier to maintain. But from a certain point in time this possibly leads to an additional workload for administrators, because they may have to maintain even more machines than before virtualization was introduced.

On the contrary, virtualization software consumes processing time to manage the different virtual machines, and this leads to some overhead. Virtualization of peripheral devices is much more resource intensive than CPU virtualization. It was shown that the throughput of virtualized network devices is only 75% down to 50% compared to non-virtualized devices, depending on applied virtualization strategies [19].

This shows, that introducing virtualization strategies in data centres can easily lead to rebound effects because it is so easy to run a new server for a service or software instead of deploying them on existing ones. Hence, it is suggested for administrators to use virtualization techniques, because it reduces power-consuming servers that are hardly working to capacity, but administrators should resist starting new virtual servers for every new service or software, because that may lead to rebound effects resulting from processing overheads caused by virtualization.

4.3 For Users

Our proposed model includes concepts for different user types. Beyond the possibilities for developers and administrators, end users also influence a sustainable software approach. In this section, we describe two different examples implementing the usage lifecycle phase of our model, which is also relevant for end users.

Generating User-specific Interactive Documents (GUIDO). In looking at Sustainable Development, not only questions regarding the environment are relevant, the social and economical dimensions have to be considered. Regarding the accessibility of information systems and governmental processes it is necessary, also for handicapped people to have access to information and to be able to participate in formal workflows. The GUIDO-System [20] depicted in Figure 5 helps users by simplifying access to official forms. By generating a generic structure of forms like official applications or notifications, the GUIDO-System provides a presentation on the user side, which takes special handicaps like visual impairment into account. By means of special profiles – the user profile and the device profile – GUIDO is able to convert arbitrary forms and documents into a user-specific presentation. Thus the GUIDO-System contributes to the social dimension of sustainability as it enables impaired people to participate in official decision and management processes within modern information societies without the need of assistance from other people.

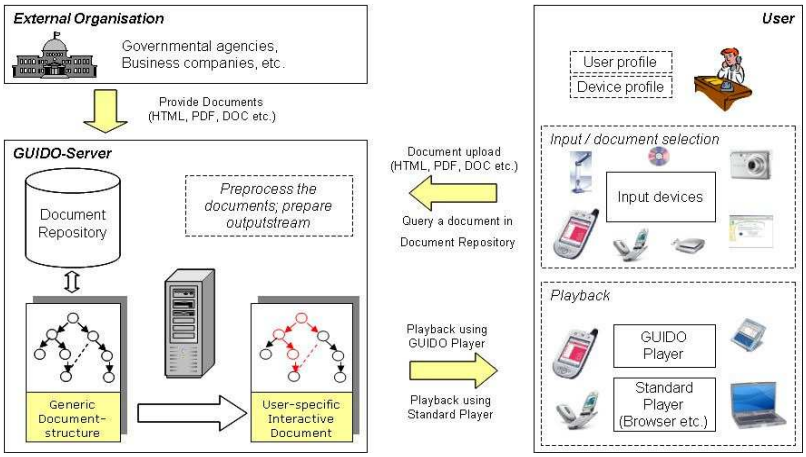

Fig. 5. System Architecture and Workflow of the GUIDO-System [20].

Fig. 6. Screenshot of the Power Indicator application [21].

Visualizing Power Quality of Websites. As mentioned, in order to achieve a "greener" Internet, one first step is to visualize for web users whether a website is hosted with renewable energies or not. Our Firefox add-on "Power Indicator" visualizes the green power state of a website via a small icon (see Fig. 6). The application is based on a database which contains web providers and their power supply companies.

5 Conclusion and Outlook

Today, the three pillars of sustainability: economy, society, and environment are far from being balanced [1]. The environmental pillar especially, is still underrepresented. Additionally, to date it is not clear whether energy consumption by ICT is greater or smaller than energy savings by ICT. Where manifold efforts of Green IT address the environmental pillar considering computer hardware, hardly any models, descriptions or realizations exist in the area of computer software so far. In order to fill this gap, it seems to be necessary to foster corresponding sustainability efforts in the field of software engineering. This can be achieved with appropriate models, descriptions and tools. Therefore, we presented in our paper the results of a few expert interviews that form the basis for a definition of the term "Sustainable Software", a process and lifecycle model for "Green and Sustainable Software Engineering", and six concrete instances of this model for practitioners.

The model comprises the process and lifecycle model component and the guidelines and checklists component. The process and lifecycle model is based on common software lifecycle process standards and common lifecycle assessment standards. It is intended to provide starting points for activities that lead to software products that are developed, provisioned and used with respect to sustainability issues. Guidelines and checklists are applicable representations of the model regarding various process steps. The presented instances of the model consider different phases and actors during the lifecycle.

Our next steps are to detail and broaden our model. More details can help in the finding of solutions for more special problems regarding energy efficiency. Broadening comprises such as barrier-free techniques as the idea of sustainable development also contains aspects of sociability and free access to common goods.

Acknowledgments. This paper evolved from the research and development project "Green Software Engineering" (GREENSOFT) sponsored by the German Federal Ministry of Education and Research under reference number 17N1209.

References

1. World Commission on Environment and Development (ed.): Our Common Future, 13th edition. Oxford University Press, Oxford (1991).
2. Fichter, K.: Sustainable Business Strategies in the Internet Economy. In: Hilty, L.M., Gilgen, P.W. (eds.) Sustainability in the Information Society, Proceedings of the 15th In-

ternational Symposium Informatics for Environment Protection, Zurich. Metropolis Verlag, Marburg (2001).
3. Erdmann, L., Hilty, L.M., Goodman, J., Arnfalk, P.: The Future Impact of ICTs on Environmental Sustainability. Technical Report Series EUR 21384 EN. IPTS, Brussels (2004), http://ftp.jrc.es/EURdoc/eur21384en.pdf.
4. U.S. EPA (U.S. Environmental Protection Agency): Report to Congress on Server and Data Center Energy Efficiency, Public Law, pp. 109-431, August 2 (2007).
5. Koomey, J.G.: Estimating total Power Consumption by Servers in the U.S. and the World. Final report, February 15 2007. Analytics Press, Oakland (2007), http://enterprise.amd.com/Downloads/svrpwrusecompletefinal.pdf.
6. Göhring W.: The Memorandum "Sustainable Information Society". Proceedings 18th International Conference Informatics for Environmental Protection, EnviroInfo, Geneva (2004), http://www.wolf-goehring.de/OverMemoSIS.pdf.
7. Coroama, V., Hilty, L.M.: Energy Consumed vs. Energy Saved by ICT – A Closer Look. In: Wohlgemuth, V., Page, B., Voigt, C. (eds.) Environmental Informatics and Industrial Environmental Protection: Concepts, Methods and Tools. 23rd International Conference on Informatics for Environmental Protection, vol. 2, pp. 353-361. Shaker Verlag, Aachen (2009).
8. Naumann, S.: Sustainability Informatics – A new Subfield of Applied Informatics? In Möller, A., Page, B., Schreiber, M. (eds.) EnviroInfo 2008. Environmental Informatics and Industrial Ecology, 22nd International Conference on Environmental Informatics, pp. 384-389. Shaker Verlag, Aachen (2008).
9. International Organization for Standardization (ed.): ISO/IEC 25000:2005 Software Engineering—Software Product Quality Requirements and Evaluation (SQuaRE)—Guide to SQuaRE. International Organization for Standardization, Geneva (2005).
10. The EU Energy Label, http://www.energy.eu/focus/energy-label.php.
11. International Organization for Standardization (ed.): ISO/IEC 12207:2008 Systems and software engineering - Software life cycle processes. International Organization for Standardization, Geneva (2008).
12. International Organization for Standardization (ed.): ISO 14040:2006 Environmental management - Life cycle assessment - Principles and framework. International Organization for Standardization, Geneva (2006).
13. Office of Government Commerce (ed.): Service Support: ITIL; [the key to] managing IT services. TSO (The Stationery Office), London (2005).
14. Dick, M., Naumann, S., Held, A.: 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. (eds.) WEBIST 2010: Proceedings of the 6th International Conference on Web Information Systems and Technologies, April 7-10 2010, Valencia, Spain, vol. 1, pp. 48 - 55. INSTICC Press, Setúbal (2010).
15. King, A.: Website Optimization. O'Reilly Media, Sebastopol (2008).
16. Theurer, T.: Performance Research, Part 2: Browser Cache Usage - Exposed! http://www.yuiblog.com/blog/2007/01/04/performance-research-part-2/.
17. Fielding, R.; Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol - HTTP/1.1. Request for Comments 2616. The Internet Society (1999), http://tools.ietf.org/html/rfc2616.
18. The Boston Consulting Group: SMART 2020 Addendum Deutschland: Die IKT-Industrie als treibende Kraft auf dem Weg zu nachhaltigem Klimaschutz (2009), http://www.gesi.org/Media/GeSINewsFullStory/tabid/85/smid/503/ArticleID/43/reftab/37/t/The%20Need%20For%20a%20SMART%20Alliance/Default.aspx.
19. Menon, A., Santos, J. R., Turner, Y., Janakiraman, G., Zwaenepoel, W.: Diagnosing Performance Overheads in the Xen Virtual Machine Environment. In: VEE '05: Proceedings

of the 1st ACM/USENIX International Conference on Virtual Execution Environments, June 11-12 2005, Chicago, USA, pp. 13-23. ACM, New York (2007).
20. Richter, S., Kuhn, N., Naumann, S., Schmidt, M.: Enhancing Accessibility to E-Government Processes. International Journal of Information Communication Technologies and Human Development, Special Issue: ICT and E-Governance, 1(2), 28-47 (2009).
21. Naumann, S., Gresk, S., Schäfer, K.: How Green is the Web? Visualizing the Power Quality of Web-sites. In: Möller, A., Page, B., Schreiber, M. (eds.) EnviroInfo 2008. Environmental Informatics and Industrial Ecology, 22nd International Conference on Environmental Informatics, pp. 62-65. Shaker Verlag, Aachen (2008).