EnviroInfo 2010 (Cologne/Bonn)
Integration of Environmental Information in Europe
Copyright © Shaker Verlag 2010. ISBN: 978-3-8322-9458-8

Enhancing Software Engineering Processes towards Sustainable Software Product Design

Markus Dick, Stefan Naumann
Trier University of Applied Sciences, Umwelt-Campus Birkenfeld
Campusallee, D-55768 Hoppstädten-Weiersbach
m.dick@umwelt-campus.de
s.naumann@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 propose a generic software development process enhancement that has the potential to integrate the consideration of sustainability aspects into arbitrary software development methodologies.

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 (Brundtland 1991). Information and Communication Technology (ICT) can take an important part within these challenges. On the one hand, ICT can optimize material flows and therefore reduces energy consumption (Hilty 2005). On the other hand, ICT itself is consuming more and more energy (Erdmann et al. 2004).
Up to now, several publications examine the relationship between the field of sustainability and ICT. They discuss the impact of ICT on the environment (Göhring 2004) or consider the balance between energy savings and energy consumptions by ICT (Coroama, Hilty 2009). Especially, to date it is not clear, whether energy consumption by ICT is greater or smaller than energy savings by ICT, e.g. because of more efficient processes or simulations of scenarios. Consequently, considering problems like climate change, the reduction of energy and resource consumption, which is caused by software, is necessary. Therefore approaches and solutions for the development and usage of sustainable software are essential. However, famous software engineering textbooks (e.g. Sommerville 1998 or Balzert 1998), do not cover issues about how to integrate environmental or sustainability aspects into software design and development.
Facing these challenges, our paper proposes a generic extension for software development processes that aims at an integration of sustainability issues in arbitrary software development processes and hopefully leads to more sustainable software products. The exemplary tools, checklists, and guidelines that assist professionals with the application of our model are currently focusing on the environmental pillar of sustainability, e.g. energy consumption. However, if this model proves to be successful, there may be a higher diversity of assisting tools and guidelines in the future, so that possibly the social and economic pillars can be covered as well.

2. Related Work

Up to now, most available publications focus on the sustainability aspects of hardware by the means of Green IT, which addresses mainly the environmental pillar of sustainability. Only few publications exist, that focus especially on software and its contributions to or its impacts on sustainability. Mocigemba (2006) introduced a Sustainable Computing Concept that also regards the software level of computing. It considers issues relevant to software production processes, like e.g. licensing, user participation, software recycling and software patents. Käfer (2009) presented conceptual and architectural issues, concerning software energy consumption and ideas for incorporating energy efficiency issues into software projects. Amsel and Tomlinson (2010) presented a software tool called “Green Tracker” that estimates the energy consumption of already deployed software in order to assist users in making decisions about the software they use. Dick et al. (2010) presented principles and suggestions that enable web developers, administrators and users to develop, operate, and use websites in a manner, so that energy consumption and data transfer is minimized.

3. Lifecycles of Software Products

Software lifecycle models or product lifecycle models, commonly known from the business context, are not suitable to identify phases in the life of products that have effects on sustainability. These lifecycle models focus on software development or maintenance aspects (e.g. development plans, release plans, service releases), and on business aspects (e.g. sales volumes, profitability).

Hence, we propose a software lifecycle model that is inspired by Life Cycle Thinking (abbr. LCT), which is also highlighted in terms like “from cradle to grave”, and is based on a product lifecycle that is outlined in ISO/TR 14062 (Deutsches Institut für Normung 2003). „[LCT] covers the extraction of raw materials, the production process, and the distribution, use, recycling, and finally disposal of products.“ (Tischner et al. 2000, page 13). This lifecycle description, which can be applied to material products, like e.g. dish washers, cars, and computer hardware, must be adapted to comply with immaterial software products (see Figure 1). Hence, the lifecycle phases, which do not fit on immaterial products, like the extraction of raw materials and recycling are removed. The model has two objectives:

Assign criteria (i.e. material, energy, data transfer) that arise from impacts on sustainability, which are caused by the software product, to stages of the product’s lifecycle.

Provide starting points for activities that optimize these impacts (whether negative or positive) as early as possible.

These activities intend to lead to more sustainable software products. The outline of the model, as described below, focuses mainly on environmental aspects of sustainability. Extensions to the other pillars of sustainability are planned for the future.

The first lifecycle phase is the *Development* phase, which is in the focus of this paper. Here, several well-organized methods and tools are applied during the software development process. These enable participating actors to assess impacts on sustainability that result from the software product over its whole lifecycle. Furthermore it allows them to take action to improve the software product in order to optimize its impacts. Hopefully, this leads to a more sustainable software product design. The *Distribution* phase is relevant for both, standard and custom software. It takes impacts on sustainability into account that result from the production of the data medium, the packaging, or the download of software packages. The next lifecycle phase is the *Acquisition* phase. Here, *acquisition* means that one evaluates different standard software products, chooses one that fits the needs best and purchases it from a software retailer. During software selection, persons responsible should also, beside technical, functional or licensing criteria, consider sustainability criteria. The *Deployment* lifecycle phase considers aspects that are relevant for administrators during deployment of software products. The *Usage* and *Maintenance* phase considers direct and indirect sustainability effects, which evolve from the utilization of the software product. The *Deactivation* phase considers aspects, which become relevant if software products are taken out of service. The *Dis-*

posal phase takes impacts on sustainability into account that result from the disposal of the data medium and packaging.

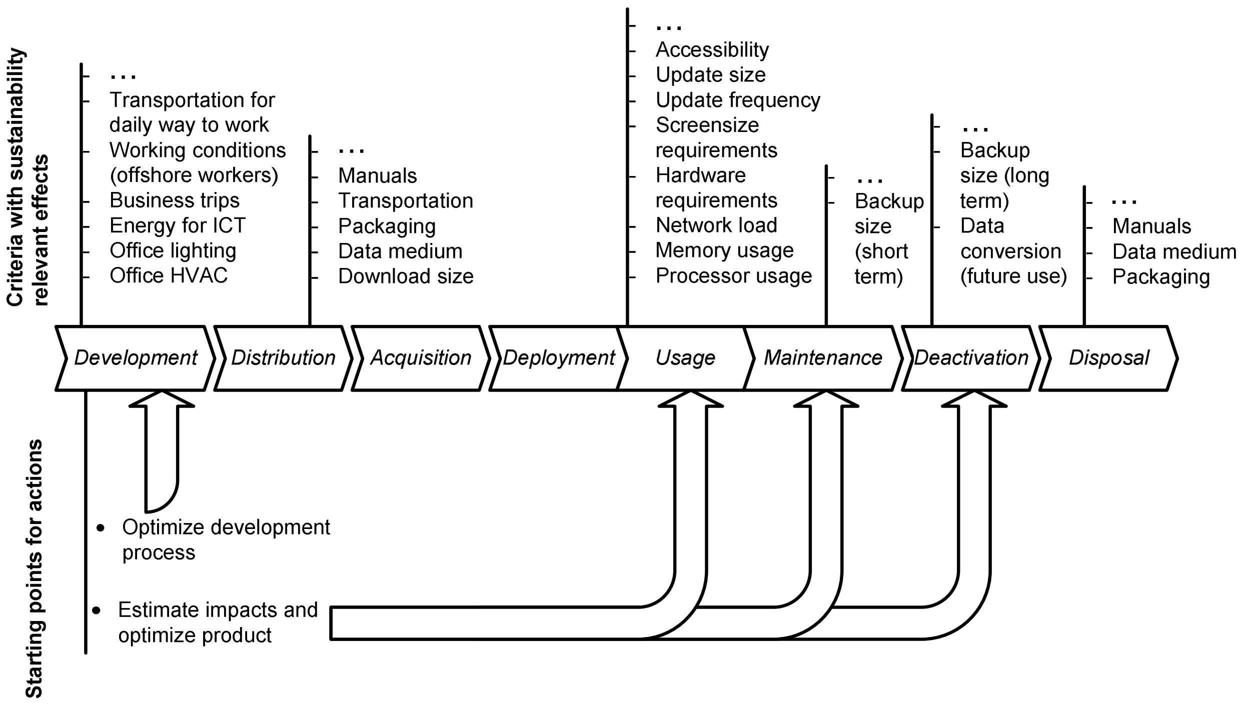

Figure 1: Life Cycle Thinking inspired lifecycle for Software Products (shows exemplary criteria and development phase focusing starting points for actions)

The future impacts on sustainability that are expected from post development phases and especially from the usage phase have already to be considered by actors during the development phase in order to be optimized. Hence, during development, actors have to anticipate and estimate first, second and/or third order impacts on sustainability or even rebound effects. In addition, actors are advised to measure the impacts on sustainability that result from the software development process itself, and to establish a continuous improvement process that optimizes the impacts. These findings lead to the model described in the following sections.

4. Sustainable Software and Sustainable Software Engineering

For the purpose of this paper, we define the terms *Sustainable Software* and *Sustainable Software Engineering* as follows:

Sustainable Software is software, whose impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software are minimal and/or which have a positive effect on sustainable development.

Sustainable Software Engineering is the art of defining and developing software products in a way so that the negative and positive impacts on sustainability that result and/or are expected to result from the software product over its whole lifecycle are continuously assessed, documented, and optimized.

5. A Generic Model for Sustainable Software Engineering

5.1 General Models for Software Engineering

Software Engineering knows several systematic process models that allow a methodical and well-organized development of software products. Typical process phases are: requirements analysis, design, implementation, testing, operation and maintenance. At a first glance, most of these process models look very different. However, when analyzing the different activities within these models, one can find activities that belong to the phases mentioned above, even though the phases are named differently, occur in a different order, or are iterated. By lifting the constraints regarding the sequential order or iterations in the waterfall model (Royce 1970), it is possible to represent most models through a general one. This general model executes phases in parallel, allows feedback between phases, and iterating phases if necessary. Hence, we are using this general model to show the integration of our enhancements in development processes, without limiting our approach to it.

5.2 Model Overview

The generic process enhancement model for sustainable software development is outlined in Figure 2. For this overview, we assume concurrent phase execution and a simple phase order without iterations, as depicted in the activity box. The model itself belongs to the development lifecycle phase of the software product lifecycle model (see Figure 1). In contrast to our software product lifecycle model, this model takes an organizational perspective to look at the development phase of a software product. Hence, it models processes and activities that should be applied in organizations.

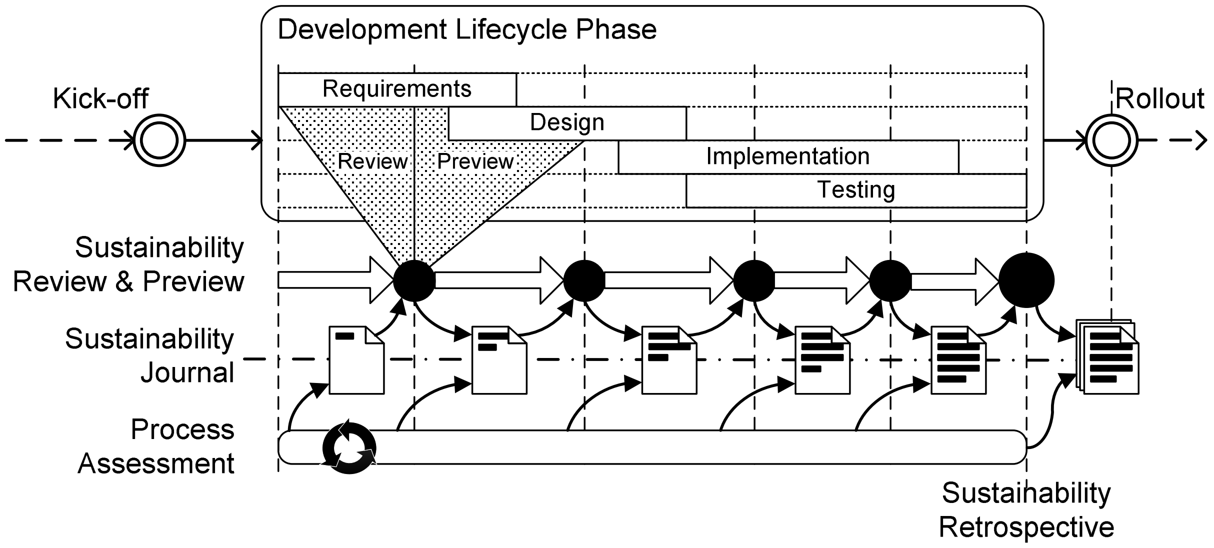

Figure 2: Process Enhancement Model for Sustainable Software Engineering

The introduced innovative enhancements are (see Figure 2): *Sustainability Reviews & Previews*, *Process Assessment*, *Sustainability Journal*, and *Sustainability Retrospective*. Additionally, these elements are accompanied and supported by assisting tools, checklists, guidelines as well as educational material. These enhancements cover impacts, which result from the production process of a software product (the development lifecycle phase) and impacts, which result from using the software product (other lifecycle phases to the right of the development phase). Process Assessment covers the first, whereas Sustainability Reviews & Previews cover the latter. Finally, the Sustainability Retrospective brings both efforts together, so that impacts over the whole lifecycle of the software product are covered.

5.3 Sustainability Reviews & Previews

Sustainability Reviews & Previews take a look on the work done, assess outcomes according to sustainability issues, and develop measures, which are realized until the next Review & Preview, in order to optimize the sustainability of the software product under development.
These reviews take software aspects, like e.g. requirements, architecture, or coding into account that have impacts on sustainability. In a team facilitation approach, actors review and assess their artefacts (e.g. requirements, architecture, coding), in order to develop more sustainable solutions if necessary. That is the Review part. The Preview part are the solutions themselves and the estimation that these solutions will be more sustainable. It is clear that the focus of this team meeting depends on the software development methodology applied. In case of the general model used in Figure 2, the focus changes from Requirements to Design etc. However, in a methodology that works with short iterations instead of process phases (e.g. agile methodologies) the focus may be on the whole development cycle of an iteration (design, implement, test). Sustainability Reviews & Previews take place after one-half or two-thirds of a process phase. This enables actors to assess impacts and to realize optimization measures within the same process phase. In a process phase, there could also be multiple Reviews & Previews, depending on its length.
In Reviews & Previews, which take the role of a formative evaluation, not only software aspects, like e.g. architecture or coding, are taken into account, but also impacts that result from the development process itself. Hence, data collected by Process Assessment could be used to optimize the sustainability of the process. This applies especially, when changes in behaviour of actors can help to improve the sustainability of a specific development process. However, impacts that result from the development process and that can hardly be influenced by actors should not be considered in Reviews & Previews (e.g. heating or the insulation of the office building). These will be considered in the Process Assessment activity.

5.4 Process Assessment

The Process Assessment activity continuously quantifies the development process. According to the idea of lifecycle assessment, it builds up some parts of the lifecycle inventory. Therefore, different data from the development process is collected, which is necessary to assess the process' impacts on sustainability. It is not the objective of this activity to assess the complete software product, because this is the objective of Sustainability Reviews & Previews and the Sustainability Retrospective. However, guidance on how the LCA method described by the ISO 14040 series could be applied to software products and their production processes is currently not available. Hence, we propose as a first approach influencing factors that have impacts on environmental sustainability and should therefore be collected by the Process Assessment activity. Relevant data is e.g.: energy for heating, ventilation, and air conditioning of the offices, energy consumption of workstations, energy consumption of lighting, energy consumption of the necessary information and communication infrastructure, business trips and the used means of travel, consumed stationary and its quality (e.g. fair trade or environmental friendly products), pro rata impacts of common corporate departments, and pro rata impacts of means of transportation used by employees for their daily way to work.

5.5 Sustainability Retrospective

At the end of the development lifecycle phase, the Sustainability Retrospective sums up the collected data, assesses the impacts on sustainability of the software product, and looks for ways to improve the sustainability of upcoming development projects and their resulting software products.

The Sustainability Retrospective combines expected impacts assessed by Sustainability Reviews & Previews, which result from using the product, with impacts assessed by Process Assessment, which result from developing the product. The outcome covers impacts of the software product over its complete lifecycle in the sense of LCA. These assessment results should be reported to consumers/customers.
Further outcomes of the Retrospective are e.g. assessments and group reflections of impacts on sustainability of the developed software product and the development process, decisions for future projects, lessons learned, or best practices regarding sustainability issues of software products and development processes. Hence, the Sustainability Retrospective is some kind of summative evaluation of the software development process as well as of the resulting software product.

5.6 Sustainability Journal

The Sustainability Journal is the information hub of the process enhancement. It is a well-structured document, which evolves simultaneously with the software project. Its purpose is to document Sustainability Reviews & Previews, Process Assessment, the Sustainability Retrospective, and finally, after the project has finished, it reports the assessed impacts on sustainability.
Sustainability Reviews & Previews should be documented as short as possible, but with enough details, so that decisions taken can be easily reconstructed by stakeholders later on. The data collected by Process Assessment activities is reported twice in the journal. First, Process Assessment reports to Reviews & Previews and is logged, if decisions follow from it. Second, the data is summed up and logged, so that it is ready to be used as a lifecycle inventory for the lifecycle analysis of the development process. The results of the lifecycle analysis are documented in the Journal and presented during the Sustainability Retrospective. The outcomes of the Retrospective, e.g. assessments and group reflections of impacts on sustainability of the developed software product and the development process, decisions for future projects, lessons learned, or best practices regarding sustainability issues of software products and development processes, are also conserved in the Journal for future use and reference.

5.7 Supporting Tools, Checklists, Guidelines and Educational Material

The process enhancements should be supported by tools (e.g. specialized software, spread sheets) that support the collection of data during Process Assessment and assist actors with LCA issues.
In order to mitigate impacts resulting from the usage phase of a software product, it is necessary, to take action already during development phase. This means that if impacts on sustainability are identified, actors, like e.g. software architects and developers, need tools, guidelines, or hints that provide them with ideas and suggestions, so that they can refine e.g. software architecture or coding. Hence, we propose a knowledgebase that assists actors in finding appropriate information. Exemplary guidelines are given in (Dick et al. 2010).
Educational material is necessary, because sustainability issues, like e.g. first order effects, second and third order effects or more involved concepts like rebound effects (Berkhout, Hertin 2001), are usually not well known to computer scientists or stakeholders/actors in software development projects. Hence, it is necessary to provide educational material that explains basic sustainability concepts, enables actors to assess impacts of software products and development processes, and to take action in order to mitigate negative or to amplify positive impacts. The objective of this educational material is not to educate actors, so that they become mature LCA specialists instantly, but to give thought-provoking impulses that have the potential to facilitate actors to reflect and assess their actions, which may hopefully lead to software products and software development processes that consider sustainability issues. Basically, the aim is that stakeholders/actors develop a pervasive awareness of the entire lifecycle of software products and its impacts on sustainability.

6. Instantiating the Model: Beyond the Waterfall Approach

The objective of this generic enhancement is that it can be applied to different software process methodologies. How that may be accomplished is shown with two methodologies that follow the agile paradigm. The first methodology is OpenUP (Open Unified Process) and the second is Scrum.

6.1 Tailoring to OpenUP

OpenUP (Eclipse Foundation 2009-10-08) is a lean Unified Process (Kruchten 2003). It claims to be an agile, lightweight process. It considers software development best practices, like e.g. iterative development, team collaboration, continuous integration and tests, and frequent deliveries of working software. The structure of the lifecycle has two dimensions. The first dimension represents the four lifecycle phases: “Inception Phase”, “Elaboration Phase”, “Construction Phase”, and “Transition Phase”. Each lifecycle phase can have several iterations. The second dimension represents activities that apply to the lifecycle phases. The main software development related activities are: “Identify and Refine Requirements”, “Outline the Architecture”, “Develop the Architecture”, “Develop Solution Increment”, and “Test Solution”. In a lifecycle phase, activities are executed in parallel. Which activities are executed and the workload generated by an activity depends on the focus of the lifecycle phase and the needs of the running process. E.g. during “Inception Phase” the activities “Identify and Refine Requirements” and “Outline the Architecture” are executed. Some activities occur only in one lifecycle phase, e.g. the “Develop the Architecture” activity, which occurs only in the “Elaboration Phase”. Other activities occur in several lifecycle phases with different strength, e.g. the “Develop Solution Increment” and “Test Solution” activities, which occur in the “Elaboration Phase”, “Construction Phase”, and in the “Transition Phase”. Each iteration closes with a review meeting, the so called “iteration assessment”, where the results of the iteration are reviewed together with the stakeholders (including the customer). If an iteration is the last iteration of the project, the review meeting is called “final assessment”, and its objective is the final acceptance of the software product by the customer.

Our approach can be integrated into OpenUP either as separate tasks or directly in already existing tasks. The iteration assessment meeting is prepared by the development team members in a team collaboration approach towards the end of an iteration. In that preparation, the team assesses whether the objectives for the current iteration are met or not. Thematically, it seems suitable to integrate the Sustainability Reviews & Previews into these preparation meetings. In this way, there is no need for additional Sustainability Review & Preview meetings, and the sustainability relevant results and measures can be directly reported to the stakeholders in the following iteration assessment meeting. The disadvantage of this solution is that there is hardly any time to implement the optimization measures in the same iteration, because the preparation meetings proceed only a few days before the iteration assessment. In order to avoid this disadvantage, it is advisable to perform Sustainability Reviews & Previews still as standalone meetings. The continuous Process Assessment that collects sustainability relevant data about the running software development process can be integrated as a separate task into the ongoing iteration management activity, which is performed by project managers. The Sustainability Retrospective should take place just before the end of the last iteration of the software project, because the assessed impacts on sustainability are highly relevant information for the customer. Hence, it is necessary to combine Process Assessment and Sustainability Review & Preview results along with the preparations for the “final assessment” meeting.

6.2 Tailoring to Scrum

Scrum (Eclipse Foundation 2008-08-15) is an agile iterative project management method that tries to deliver a potentially shippable product in each iteration. The iterations are called “sprints”. Each sprint starts with a Sprint Planning Meeting and ends with the Sprint Review and the Sprint Retrospective. In Sprint Planning Meetings the development team and the product owner determine, which features should be im-

plemented in the following sprint. In Sprint Review meetings, the team demonstrates the completed features as a potentially shippable product increment. The implemented features are then accepted or rejected by the product owner. The Sprint Retrospective is a team learning approach that has the objective to improve the development process and the teamwork. Hence, the team discusses and reflects the last sprint and agrees on changes for the next sprint.
Our approach can be integrated in Scrum as additional aspects. The Sustainability Reviews & Previews can take place after two-thirds of a sprint as proposed in the relating section above. The outcomes of the Sustainability Reviews & Previews should be reported during the Sprint Review meetings, because the product owner accepts or rejects the implemented features and thus also the measures, which were taken to improve the sustainability of the software product. It is the responsibility of the development team to take care of the ongoing Process Assessment. The Sustainability Retrospective should take place just before the end of the last Sprint Review, so that the team is enabled to report the combined assessment results to the product owner. After the project as finished, the team should discuss the other aspects of the Sustainability Retrospective, like decisions for future projects, lessons learned, and best practices regarding sustainability issues. This ensures that the team can discuss and reflect these aspects without pressure, which hopefully leads to better results.

7. Conclusion and Outlook

In this paper, we presented a generic model that enhances common software development processes in the direction of sustainable software product design. It introduces several activities and artefacts in order to achieve "Sustainable Software Engineering", e.g. Sustainability Reviews & Previews, ongoing Process Assessments, a Sustainability Retrospective, and a Sustainability Journal. The model is accompanied by supporting tools, guidelines, and educational material. Although, we initially used a simple waterfall-like approach to show how the activities and artefacts of the model work together, the model itself does not rely on a specific software development methodology. This is emphasized by our proposal of how the model may be customized to fit agile software development processes. However, there is a chance that our model can be applied more easily to agile processes than to complex, high ceremony processes.
Our next steps are: developing and evaluating the necessary sustainability criteria for software, evaluating the model initially in student projects and subsequently in real-life projects, and developing the accompanying tools, guidelines, and educational material.
For the future, we plan to broaden our model. Broadening covers aspects like considering other phases of the software lifecycle and other aspects of sustainability like the social and economic pillars.
Standardized seals of quality for sustainable software development processes, commonly accepted criteria for the question what sustainable software should do/not do, or a standardized product declaration for sustainable software would constitute a great step forward. Following the development e.g. in organic food, such seals of quality will boost the necessity for models for sustainable software engineering in practice, research, and teaching.

8. Acknowledgement

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

9. Literature

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

Balzert, H. (1998): Lehrbuch der Software-Technik. Software-Management, Software- Qualitätssicherung, Unternehmensmodellierung, Spektrum Akademischer Verl., Heidelberg, Berlin.

Berkhout, F., Hertin, J. (2001): Impacts of Information and Communication Technologies on Environmental Sustainability: speculations and evidence. Report to the OECD. OECD (ed.). Internet http://www.oecd.org/dataoecd/4/6/1897156.pdf, last access 2010-06-24.

Brundtland, G. H. (1991): Our common future. 13. ed., Univ. Press, Oxford.

Coroama, V., Hilty, L. M. (2009): Energy Consumed vs. Energy Saved by ICT – A Closer Look. In: Wohlgemuth, V., Page, B., Voigt, K. (eds.): Environmental informatics and industrial environmental protection: concepts, methods and tools. EnviroInfo 2009 ; 23rd International Conference on Informatics for Environmental Protection ; proceedings of the 23rd International Conference Environmental Informatics - Informatics for environmental protection, sustainable development and risk management, September 09 - 11, 2009, HTW Berlin, University of Applied Sciences, Germany. Volume 1: Sessions für Band 1, Shaker, Aachen, vol. 2, pages 353–361.

Deutsches Institut für Normung (2003): Umweltmanagement-Integration von Umweltaspekten in Produktdesign und -entwicklung. Deutsche und englische Fassung ISO/TR 14062:2002. 1. ed., Beuth, Berlin.

Dick, M., Naumann, S., Held, A. (2010): Green Web Engineering. A Set of Principles to Support the Development and Operation of "Green" Websites and their Utilization during a Website's Life Cycle. In: Filipe, J., Cordeiro, J. (eds.): 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, pages 48–55.

Eclipse Foundation (ed.) (2008): Scrum. Mountain Goat Software; ATSC. Internet http://epf.eclipse.org/wikis/scrum/, last access 2010-06-28.

Eclipse Foundation (ed.) (2009): OpenUP. Internet http://epf.eclipse.org/wikis/openup/, last access 2010-06-28.

Erdmann, L., Hilty, L. M., Goodman, J., Arnfalk, P. (2004): The Future Impact of ICTs on Environmental Sustainability. C. R. Casal, C. V. Wunnik, L. D. Sancho, J. C. Burgelman, P. Desruelle (eds.). European Commission; IPTS - Institute for Prospective Technological Studies. Internet http://ftp.jrc.es/EURdoc/eur21384en.pdf, last access 2010-04-26.

Göhring, W. (2004): The Memorandum “Sustainable Information Society”. 18th International Conference “Informatics for Environmental Protection”, EnviroInfo 2004, 21—23 October, Geneva. Internet http://www.iai.fzk.de/Fachgruppe/GI/litArchive/EnviroInfo/2004Geneva/Volume2-Sh_ring_EnviroInfo_2004/The_Memorandum_Sustainable_Information_Society_--a1306.pdf, last access 2010-07-01.

Hilty, L. M. (2005): Information systems for sustainable development, Idea Group Publishing, Hershey, Pa.

Käfer, G. (2009): Green SE: Ideas for Including Energy Efficiency into your Software Projects. Technical Briefing (TB2). 31st International Conference on Software Engineering, Vancouver. Internet http://www.cs.uoregon.edu/events/icse2009/specialSessions/#TB2, last access 2010-03-04.

Kruchten, P. (2003): The rational unified process. An introduction. 2. ed., Addison-Wesley, Boston.

Mocigemba, D. (2006): Sustainable Computing. In: Poiesis & Praxis: International Journal of Technology Assessment and Ethics of Science, volume 4, number 3, pages 163–184. Internet http://springerlink.com/content/uuu134v142p0g521/?p=0ee983bcf4e34a9d886563d1a9a90172&pi=1, last access 2010-02-19.

Royce, W. W. (1970): Managing the development of large software systems: concepts and techniques. In: Proc. IEEE WESTCON, Los Angeles, pages 1-9. Internet http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf, last access 2010-05-23.

Sommerville, I. (1998): Software engineering. 5. ed., Addison-Wesley, Harlow.

Tischner, U., Dietz, B., Maßelter, S., Schmincke, E., Prösler, M., Rubik, F., Hirschl, B. (2000): How to do EcoDesign? A guide for environmentally and economically sound design, Verlag form, Frankfurt am Main.