Sustainable Software Engineering: Process and Quality Models, Life Cycle, and Social Aspects

Stefan Naumann, Eva Kern, Markus Dick and Timo Johann

Abstract Sustainability intersects Information and Communication Technology in two domains: Green IT (how can we make ICT itself more sustainable?) and Green by IT (how can we achieve sustainability through ICT?). On a closer look, it is software that links these two fields: In "classic" Green IT, there are many ways to build and use hardware in a more energy-efficient way. On the software side, Green by IT has often been software-based until now, involving tools that help to optimize logistics and automate processes to save energy, for example. However, the debate over software-induced energy consumption is just beginning. To date, few studies have been conducted about the energy saving potential of software itself. Therefore, it is important to investigate the meaning of sustainable software and sustainable software engineering. This chapter provides definitions of these concepts. In addition, it presents a reference model of sustainable software as well as its engineering. However, it provides only a short introduction of the model itself. The sub-model "Sustainability Criteria for Software Products" and sustainable software process models are examined in greater detail.

Keywords Green software · Green IT · Sustainable software engineering · GREENSOFT model · Software process models

S. Naumann (✉) · E. Kern · T. Johann
Institute for Software Systems, Trier University of Applied Sciences, Environmental Campus, Birkenfeld, Germany
e-mail: s.naumann@umwelt-campus.de

E. Kern
e-mail: e.kern@umwelt-campus.de

T. Johann
e-mail: t.johann@umwelt-campus.de

M. Dick
Sustainable Software Blog, Kusel, Germany
e-mail: markusadick@gmail.com
URL: http://sustainablesoftware.blogspot.com

L.M. Hilty and B. Aebischer (eds.), *ICT Innovations for Sustainability*, Advances in Intelligent Systems and Computing 310,
DOI 10.1007/978-3-319-09228-7_11

1 Introduction and Motivation

Green IT and Green by IT are common terms in the debate over ICT and sustainability. *Green by IT* (also: Green through IT) covers the support of sustainable development by means of ICT. This includes for example software that reduces environmental problems through optimization. *Green IT* (also: Green in IT) denotes actions through which ICT itself could become more sustainable. This concerns hardware and software, where the hardware part is common knowledge. The software side, by contrast, is still the subject of current research projects. Here, two main questions are being investigated: What is sustainable software, and how can software be produced in a sustainable way? The first question deals with quality characteristics concerning the energy consumption of software, for example. Since software is a very complex product in its architecture, functionalities, and usage, this is not an easy task. The second question concerns the process of software engineering and how it can be improved in order to produce sustainable software in a sustainable way.

In this chapter, we present a model for sustainable software and its engineering, as well as an approach to defining sustainability characteristics for software products and a software life cycle.

2 What Is Sustainable Software Engineering?

In order to classify issues in sustainable software and its engineering, we provide some definitions.

First we define *Sustainable Software* as software whose development, deployment, and usage results in minimal direct and indirect negative impacts or even positive impacts on the economy, society, human beings, and the environment [1]. A prerequisite for sustainable software is a sustainable development process, which refers to the consideration of environmental and other impacts during the software life cycle and the pursuit of the goals of sustainable development.

Building on this we can define *Sustainable Software Engineering* as the art of developing sustainable software through a sustainable software engineering process. In this, software products must be defined and developed in such a way that the negative and positive impacts on sustainable development that result or are expected to result from the software product over its whole life cycle are continuously assessed, documented, and used for further optimization of the software product [1].

Both of these definitions are based on product life cycles in terms of Life Cycle Assessment (abbr. LCA) or a cradle-to-grave approach. Additionally, Lami et al. assert that a *(Green and) Sustainable Software Process* is a "software process that meets its (realistic) sustainability objectives, expressed in terms of direct and indirect impacts on economy, society, human beings, and environment that result from its definition and deployment" [2].

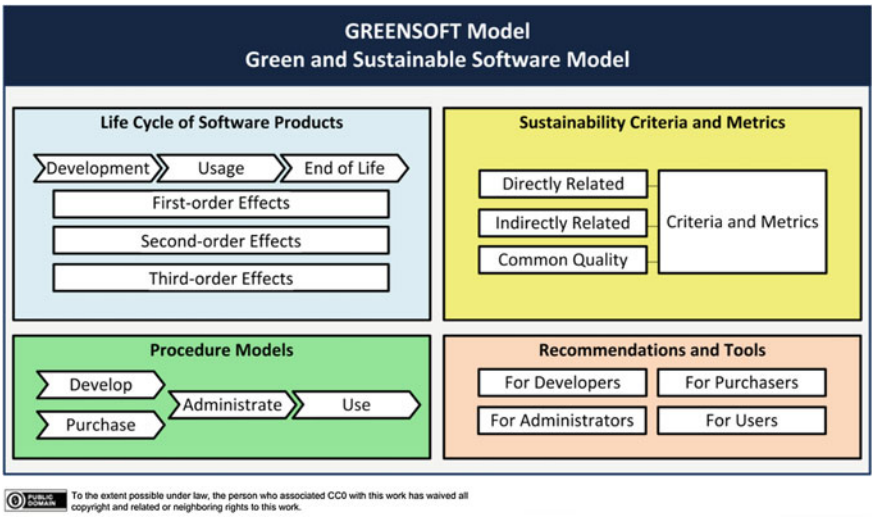

Fig. 1 The GREENSOFT reference model [3]

Summarizing these definitions, a sustainable software product ideally meets three conditions:

- The software is produced in a way that meets sustainability objectives.
- The software has minimal negative social and environmental impacts during its usage (first-order effects).
- The software functionality reinforces sustainable development or at least has no negative impacts on the society or environment (second-order and systemic effects).

The GREENSOFT Model (Fig. 1) classifies and sorts the described characteristics of sustainable software and its engineering [1]. This reference model contains four parts: the life cycle of software products; criteria and metrics that represent and measure sustainability aspects directly and indirectly related to the software product; procedure models for the different phases; and recommendations for action as well as tools. The model is described in detail in [1].

3 Related Work

Based on the definitions in Sect. 2 and the GREENSOFT Model, we summarize related work. A good overview of sustainable software engineering can be found in [4]: here, the author gathered 96 relevant publications on sustainable software engineering. They also discussed the question of what sustainability means in (and for) software engineering [5]. Amsel et al. [6] assert that sustainable software

engineering should develop software that meets the needs of users while reducing environmental impacts.

A methodology to measure and incrementally improve the sustainability of software projects has been presented by Albertao et al. [7]. They advise implementing sustainable aspects continuously throughout the assessment phase, reflection phase, and goal improvement phase. In order to make the different sustainability issues manageable, they describe properties of a quality model, which they develop further in a later work that considers the overall software process [8]. Calero et al. [9] examine such software quality standards in greater detail, examining how they could reinforce sustainability.

Agarwal et al. [10] analyze and discuss the possibilities and benefits of green software. One of their findings is that more efficient algorithms will take less time to execute and thus support sustainability overall. In addition, they present methods to develop software in a sustainable way, compare them to conventional methods, and list some further environmentally friendly best practices for the development and implementation of software systems.

Based on the life cycle of software, Taina proposes metrics [11] and a method to calculate software's carbon footprint [12]. To do so, he analyzes the impacts of each software development phase for a generic project. The resulting carbon footprint is mainly influenced by the development phase, but also by the way it is delivered and how it will be used by the customers. Lami et al. [2] define software sustainability from a process-centric perspective. They define processes in a way that they can be measured in terms of process capability according to the ISO/IEC 15504 standard. They also distinguish between the sustainability factors power, paper, and fuel consumption. Dick et al. [13] discuss how particularly agile methods could improve sustainability in software engineering processes.

Another approach to Green Software Development is presented by Shenoy and Eeratta [14]. They also take the whole life cycle of software into account and offer suggestions for the typical development phases of software. Käfer [15] also presents conceptual and architectural issues concerning software energy consumption as well as ideas for incorporating energy efficiency issues into software projects. Mahaux [16], Penzenstadler [17] and Kocak [18] consider requirements for engineering, while Sahin [19] and Shojaee [20] look at the design of energy-efficient software.

4 A Quality Model for Sustainable Software

In order to decide whether or not software is sustainable, appropriate criteria are required. A first approach to developing these is offered by the Quality Model for Sustainable Software, presented in the following section. Overall, the different aspects can be grouped into common criteria, directly related criteria, and indirectly related criteria and metrics, as pictured in our Quality Model for Sustainable Software (Fig. 2).

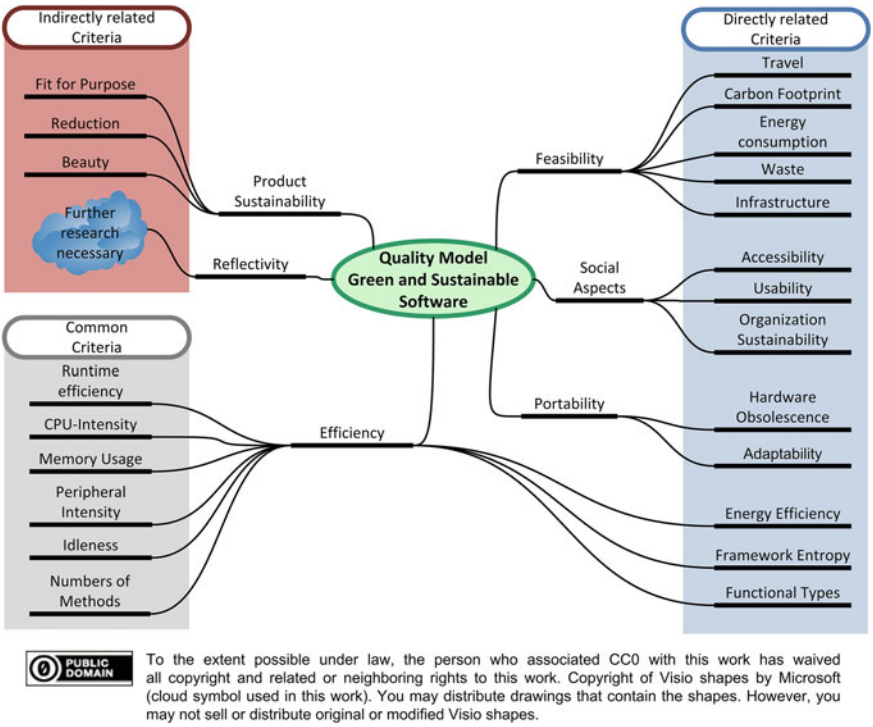

Fig. 2 Quality model for sustainable software [3]

Common Quality Criteria. The common criteria arise from the well known and standardized quality aspects for software, issued by the International Organization for Standardization [21]. The proposed quality model takes such aspects into account as *Efficiency*, *Reusability*, *Modifiability*, and *Usability*. These quality aspects extend to the field of sustainable development [8].

The quality aspects belonging to *Efficiency* are Runtime Efficiency, CPU-Intensity, Memory Usage, Peripheral Intensity, Idleness, and Number of Methods. *Runtime Efficiency* considers the time needed to finish executing depending on its implementation. In this context, Capra et al. [22] present the relation between faster application and higher energy consumption. The aspects *CPU-Intensity*, *Memory Usage* and *Peripheral Intensity* cover the resource utilization caused by software execution. In fact, the effects of the intensity of the resource consumption on the resulting energy consumption or even on system durability need to be analyzed. *Idleness* describes how often the system is idle. This aspect is relevant to certain types of software systems, such as virtual servers [11]. The total *Number of Methods* reflects the size of applications [22].

Directly Related Criteria. In contrast to common efficiency criteria, which can be considered as indicators of energy efficiency [23, 24], the energy efficiency of a system itself can be ranked in a reference system [25, 26]. This means its goal is to optimize the energy consumption in relation to delivered service items. Thus, the quality criterion *Energy Efficiency*, belonging to Efficiency in general, is assigned to the directly related criteria [1]. The *Framework Entropy* represents the grade of usage of external libraries and high level constructs. The different types of applications with their different levels of energy efficiency are indicated by the criterion *Functional Types* [22].

Hardware Obsolescence [1, 8, 27] considers the amount of hardware that needs to be replaced before the sustainably worthwhile lifetime is reached, due to software with higher system requirements. This criterion goes together with the quality aspect *Adaptability*. Both aspects belong to the criterion *Portability*. Here, Adaptability describes the possibilities to adapt the software to the specific consumer needs.

The quality criterion *Feasibility* evaluates the effects of the product development, or rather the software engineering process, on sustainable development. In this context, *Travel* includes business trips that belong to a software project and, where appropriate, the daily trips to work [8]. The resulting *Carbon Footprint* specifies the amount of CO_2 emissions caused by software development in the course of the software's life cycle [12]. The aspect *Energy Consumption* indicates the consumed energy caused by software development. *Waste* counts the resources consumed during software development for activities that do not have any additional value for customers or end users [11]. The aspect *Infrastructure* accounts for the conditions facilitating the usage of the software product.

From the perspective of the social aspects of sustainable development, there may be more qualities that require special measurements and assessment methods. As a first step, these aspects are called *Accessibility, Usability* and *Organization Sustainability*. Whereas Accessibility and Usability are well known aspects in software engineering, the sustainability of the organization is usually not considered in common software engineering processes and software's life cycle. *Accessibility* indicates whether the software product can be used by as many people as possible without any restrictions. *Usability* covers the understandability, learnability and operability of a software product, i.e. if the software usage is user-friendly and easily learned. The aspect *Organization Sustainability* covers the social situation in the software company, including working conditions, for example.

Indirectly Related Criteria. By using software, resources and energy can be reduced in other branches or application domains. Hence, criteria indirectly related to the software product itself also need to be considered.

Product Sustainability summarizes the effects of software on other products and services. It covers effects of usage, as well as systemic effects. The following aspects are presented by Taina [11]: *Fit for Purpose* takes the fitness of software for its specific operation purpose into account. The contribution of software within

its application domain to reducing emissions, waste, and the like is indicated by the aspect *Reduction*. *Beauty* aims to value the software for its contribution to sustainable development [11]. Indeed, since a reference system is required to have a comparative figure, the exact quantification of these aspects may be difficult.

According to Taina [11], the quality *Reflectivity* refers to the quality aspect of efficiency. It specifies the manner in which the usage of software influences the resources that are indirectly needed by other persons or systems. Since the aspect *Reflectivity* does not belong to the first-order effects or the efficiency effects of ICT in the proper sense of the term but counts as an effect of use, it is indicated as a standalone quality of the indirectly related criteria. However, more research is required in this field in order to take into account user behavior and usage scenarios.

5 Process Models of Sustainable Software Engineering

According to our definitions, the precondition for creating software products that meet the presented sustainability criteria is a software engineering process that meets sustainability objectives. Hence, corresponding process models are needed. In the following, different approaches for such models will be presented and compared. First, we provide a description of the process-centric software sustainability according to Lami et al. [2], which defines a set of reference processes to be applied during sustainable software development life cycles. Then, we describe our process model for sustainable software development [13] and the green model for sustainable software engineering by Mahmoud and Ahmad [28]. These two models are checked for their compliance with the reference processes by Lami et al. [2], in order to determine which reference processes are covered by the models and which ones are missing.

5.1 Process-Centric Software Sustainability

Lami et al. [2] define a set of high-level reference processes which add sustainability concepts to the software reference processes defined in ISO/IEC 12207. The standard reference processes describe all activities directly and indirectly related to development, maintenance, and operation of software.

Lami et al. point out that processes regarding sustainability are missing in ISO/IEC 12207. Therefore, they define a set of three complementary sustainability processes: Sustainability Management Process, Sustainability Engineering Process and Sustainability Qualification Process.

These additional reference processes aim at continuously improving the sustainability of both software products and software processes. According to their

definition of green and sustainable software processes, a process is sustainable if its impacts on sustainability are small (see definitions in Sect. 2).

Sustainability Management Process. The management process includes a preliminary phase, a planning phase, a monitoring phase, and a supplier sustainability control. In the preliminary phase, the principles and criteria for sustainability are established. In the planning phase, sustainability activities of the development process are indicated, and the corresponding requirements as well as necessary resources are planned. Afterwards, the sustainability of the deployed activities and their conformity with the requirements are monitored. The last part of the management process (supplier sustainability control) deals with sustainability policies and requirements for supplied products and services. Here, an agreement needs to be found and the supplier's sustainability needs to be monitored.

Sustainability Engineering Process. The engineering process is concerned with suitable tools and methods to enable and support a sustainable development process. In this context, sustainable issues and green principles for the development are defined, applied, and analyzed. The energy and resource consumption are factors that impact the sustainability and thus should be identified at the start of the engineering process. In the next step, the impacts of these effects should be analyzed in order to subsequently set sustainability objectives for the development process. In addition to the impacts of the process itself, the impacts of change requests on sustainability should be determined.

Sustainability Qualification Process. The qualification process applies to external resources such as engineering and management support tools. Aimed at sustainable products, these external resources need to be sustainable as well. In order to ensure their quality, a qualification strategy, an implementation plan for the strategy, documentation of the outcomes of the qualification, and a qualification report are required.

5.2 A Process Model for Agile and Sustainable Software Engineering

The Software Process Enhancement for Agile and Sustainable Software Engineering (GSSE) [13] was presented as an exemplary process that fits into the procedure model part of the GREENSOFT model (Fig. 1). It is not a complete software development process, but it claims to be adaptable to any process regardless of whether it works with iterations, phases or both. The only requirement is the possibility to inject its activities repetitively into the target process.

The enhancement is claimed to be lightweight. There are only two main activities that constitute the process: Process Assessment and Sustainability Reviews and Previews. The focus of Process Assessment is the sustainability of the software development process itself. The sustainability of the software product is addressed with Reviews and Previews. Both activities rely on a schematic life cycle model, inspired by life cycle assessment that denotes different impacts and effects of software on sustainable development that should be addressed.

The process distinguishes between three roles: the Customer Representative, the Development Team, and the Sustainability Executive. The Customer Representative represents the organization or the users who finance and use the software. The Development Team is responsible for developing the software as well as for improving the software with regard to sustainability-related non-functional requirements. The Sustainability Executive is responsible for continuous Process Assessment, organizing the Review and Preview meetings, and compiling a final report with the key figures and findings of Reviews, Previews and Process Assessment.

Process Assessment is a continuous activity alongside the software development process. It is meant to collect and edit data from the process that can be used for a carbon footprint calculation or even for life cycle assessment. The following data may be monitored and collected: Data regarding energy for running the developer's offices and IT infrastructure and for monitoring the means of travel for business trips and commuting, as well as the quantity of used stationery, e.g. paper for reports [29]. During the software development process, Process Assessment should briefly report to Reviews and Previews to keep the developers informed.

Sustainability Reviews and Previews focus on the software under development to optimize it in terms of its impacts and effects on sustainable development. During development, several Review and Preview meetings are conducted, e.g. after two-thirds of an iteration, which establishes a continuous improvement cycle. In reviews, the recent work is examined for any sustainability issues and solutions are then drawn up together with the expected rate of improvement, which is the preview part. In the subsequent Review and Preview session, these solutions are assessed to determine whether or not the issues have been addressed. Therefore, developers need methods and figures to explore these issues. In [13] a method to measure the energy efficiency of software is proposed that is based on unit testing. Other methods may be appropriate as well, because from a life cycle perspective, energy consumption is only one issue. Other aspects are resource and hardware requirements, accessibility, or even usage effects such as substitution, dematerialization, and rebound effects (cf. Fig. 2). Therefore, Reviews and Previews should already start in early process stages when the software product is defined [29]. The input from Process Assessment should be considered if there are issues that can be solved by the attendees themselves (e.g. chosen means of travel or local energy consumption).

At the end of the software development project, a report including the outcomes and key findings of Process Assessment (maybe a carbon footprint calculation) as

well as Reviews and Previews should be handed over to the customer representatives. Additionally, outstanding results, lessons learned, and best practices should be preserved in a knowledge base to be available to other projects.

5.3 A Green Model for Sustainable Software Engineering

The Green Model for Sustainable Software Engineering (GSEP) [28] is inspired by the GREENSOFT Model. It is a two-level model that comes with a complete Green Software Engineering Process in the first level and a collection of exemplary software tools, promoting green computing in the second level. In the following, we provide a short overview of the engineering process. The process claims to combine the advantages of sequential, iterative, and agile software processes to produce a more sustainable one. The single phases of the process are either green processes or there are at least green guidelines.

The process consists of nine successive phases: Requirements, Design, Unit Testing, Implementation, System Testing, Green Analysis, Usage, Maintenance, and Disposal. As can be seen, the traditional phases of sequential software processes are extended with Green Analysis, Usage, and Disposal. An increment milestone can be set between Implementation and System Testing. The system release milestone occurs between Green Analysis and Usage. The process flow explicitly models possibilities to return to Requirements from Unit Testing, System Testing, Green Analysis, and Maintenance.

The Requirements phase is the most critical and important phase for the resulting greenness of a software product, because the requirements define the basic features and thus affect all the following phases. Hence, it is important that the requirements consider green issues and objectives. The requirements phase includes a risk assessment activity in terms of energy efficiency as well as a requirements test activity. If any issues are discovered during these activities, the Requirement phase may be started again.

The Design phase develops the system architecture based on the requirements. Three design decision guidelines are provided to support architects: (1) outline a compact design of data structures and efficient algorithms (however, a better performance does not necessarily result in a higher energy efficiency), (2) design smart and efficient functionality (results in an efficient algorithm and fewer lines of code during implementation), and (3) decide carefully which frameworks and external libraries are necessary (additional layers may cause additional processing time but may also simplify software engineers' work). Components should be reused if possible (requires less energy-consuming development efforts). If reused components do not meet the energy efficiency requirements, they should be reorganized to conform to these requirements. The Implementation phase is described together with the Design phase and does not come with any further process descriptions, since it depends on the Design phase.

Following Mahmoud et al., the Unit Testing phase comes before Implementation because if non-conformance with the requirements is encountered here, less energy is consumed if the process steps back to the Requirements phase than if Unit Testing comes after the Implementation. However, this does not mean that Unit Tests during the Implementation phase are not allowed. After Implementation and System Testing, the Green Analysis phase is performed. This phase determines the greenness of each increment by applying specific metrics to the developed code.

As an example of green performance indicators that characterize the greenness of each stage in the software engineering process, Mahmoud et al. suggest performance or CPU usage. The identified energy usages should be mapped to the source code to optimize the affected algorithms. Afterwards it is decided whether even (small) changes in requirements are necessary in order to optimize energy usage. In fact, this means that the process may step back to Requirements in order to add missing requirements, correct misunderstandings during the Requirement phase and in this way optimize the product. They argue that this step back is acceptable if it results in less severe mistakes. If only changes to the source code are necessary, these are directly implemented within this phase without stepping back.

Whereas Requirements, Design, Implementation, the Testing phases, and the Green Analysis phase come with some process and workflow descriptions, Usage, Maintenance, and Disposal come only with a set of general guidelines.

5.4 Comparison of the Process Models

In the following, we compare how Mahmoud and Ahmad's [28] Green Software Engineering Process (GSEP, see Sect. 5.3) and the process enhancement for agile and Sustainable Software Engineering (GSSE, see Sect. 5.2) by Dick et al. [13] implement the reference processes for green and sustainable software defined by Lami et al.

Both GSSE and GSEP processes implement only the Planning phase of the Sustainability Management Process. GSEP explicitly defines different sequential phases, including a specialized Green Analysis phase, as well as the possibility of having several iterations. It is necessary to plan the number of iterations and to schedule different activities connected to the Green Analysis phase. The same is true for GSSE, where the Reviews and Previews have to be scheduled after two-thirds of an iteration or phase. More specifically, the Sustainability Executive is responsible for organizing the sustainability-centric activities, which obviously include time planning as well as resource allocation. For the other phases of the reference process, the Preliminary phase, Monitoring phase, and Supplier Sustainability Control, no equivalents could be spotted in either GSEP or in GSSE.

The Sustainability Engineering Process is implemented by GSEP and GSSE and both processes supply the defined outcomes. GSEP identifies factors that affect

sustainability explicitly within requirements engineering, whereas GSSE references a schematic Life Cycle model of software products that has the sole purpose of collating exemplary effects and impacts on sustainable development with life cycle phases known from Life Cycle Analysis (manufacturing, distribution, usage, end of life). Both processes perform sustainability analysis to determine the impact of the sustainability factors. GSEP has the Requirements and the Green Analysis phase and GSSE has the Reviews and Previews as well as Process Assessment. GSEP explicitly defines sustainability objectives in Requirements, whereas GSSE mentions sustainability objectives implicitly together with the role of the Development Team. The team is responsible for the non-functional requirements [13] and is therefore responsible to drive the Reviews and Previews. Exactly where the sustainability objectives as non-functional requirements are defined is left by GSSE to the specific implementation of the target process. GSEP and GSSE refer to principles and guidelines that are helpful to find adequate techniques and methods appropriate to accomplishing the sustainability objectives. Regarding the techniques and methods for sustainability, GSEP defines a large group of metrics and tools in its second level. GSSE mentions a carbon footprint calculation and especially methods and metrics to measure and assess the energy efficiency of software. The reference process requires an analysis of the sustainability of change requests, which is not specifically implemented by GSEP or GSSE. However, both processes claim to be agile and thus should welcome changes. GSEP may possibly analyze change requests in Requirements or the Green Analysis; GSSE may handle change requests in Reviews and Previews.

Both GSEP and GSSE focus on an implementation of the Sustainability Engineering Process. The Sustainability Qualification Process does not exist in either process model. As for the Sustainability Management Process, only the Planning phase can be seen to be rudimentarily implemented. Where GSSE was, according to the references, not designed with Lami et al. in mind, GSEP references Lami et al. and thus could have been designed to recognize more aspects of the reference processes. Both process descriptions should be extended to reflect more of the missing aspects. In particular, GSSE should describe how sustainability objectives are obtained or whose role it is to determine them. Taken as a whole, GSEP and GSSE cover most aspects of the Sustainability Engineering Process, while the aspects of the other processes are almost completely missing.

6 Conclusions and Outlook

In conclusion, we found that software plays an important role for ICT and sustainability. We looked especially at how software engineering can be made more sustainable. In our contribution, we presented a model for classifying sustainable software and it is engineering. If ICT and especially software are to make a contribution to sustainability, it is also necessary for software engineers to take

the energy consumption into account. Our model especially showed quality aspects of software engineering. The main goal is to make the software itself as well as software's development process more sustainable.

Measured against the requirement to take all aspects of sustainability into account, one must concede that the social aspect has been neglected thus far [30]. Recent publications in the field of Sustainability Informatics mention social aspects only in passing. Therefore, one should also take a close look at the socialization of the software engineering process. The latest research shows the high likelihood of the inclusion of users and their communities in the engineering process of supporting and developing more sustainable systems [31].

Overall, Green IT, sustainable software or green software engineering and other concepts of ICT for Sustainability are not standalone problems. Moreover, every process of hardware and software development should involve aspects of Sustainable Development. In this context, we answered the questions "What is a green, or rather a sustainable software product?" "What are its criteria?" and "How can we develop metrics for sustainable software and its engineering?" To address these questions, it is necessary to provide additional tools and methods for software developers and for everyone else who has a stake in software development and usage. Aspects of sustainable development should be routinely addressed during the whole life cycle of software products. Ideally, even customers should understand Green IT as a "must-have" requirement. Hence, it is necessary to reach a degree of standardization in the field of energy-efficient software and its sustainable production.

References

1. Naumann, S., Dick, M., Kern, E. et al.: The GREENSOFT model: a reference model for green and sustainable software and its engineering. SUSCOM **1**(4), 294–304 (2011). doi:10.1016/j.suscom.2011.06.004
2. Lami, G., Fabbrini, F., Fusani, M.: Software sustainability from a process-centric perspective. In: Winkler, D., O'Connor, R.V., Messnarz R. (eds.) EuroSPI 2012, CCIS 301, pp. 97–108. Springer, Berlin, (2012)
3. Research Project GREENSOFT: Website: Research Project Green Software Engineering—Downloads (2014) http://www.green-software-engineering.de/en/downloads.html
4. Penzenstadler, B., Bauer, V., Calero, C., Franch, X.: Sustainability in software engineering: a systematic literature review. Accessed 10 Jul 2012
5. Penzenstadler, B.: Towards a Definition of Sustainability in and for Software Engineering. In: SAC '13, Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 1183–1185
6. Amsel, N., Ibrahim, Z., Malik, A., Tomlinson, B.: Toward sustainable software engineering: NIER track. In: 33rd International Conference on Software Engineering (ICSE), pp. 976–979. 2011
7. Albertao, F.: sustainable software engineering (2004). http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering#about. Accessed 30 Nov 2010
8. Albertao, F., Xiao, J., Tian, C. et al.: Measuring the sustainability performance of software projects. In: IEEE Computer Society (ed) 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China, pp. 369–373. 2010

9. Calero, C., Bertoa, M.F., Moraga, M.Á.: Sustainability and quality: Icing on the cake. In: Penzenstadler, B., Mahaux, M., Salinesi, C. (eds.) Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems, Rio, Brasil, July 15, 2013. CEUR-WS.org
10. Agarwal, S., Nath, A., Chowdhury, D.: Sustainable approaches and good practices in green software engineering. IJRRCS **3**(1), 1425–1428 (2012)
11. Taina, J.: Good, bad, and beautiful software—in search of green software quality factors. CEPIS UPGRADE **XII**(4), 22–27 (2011)
12. Taina, J.: How green is your software? In: Tyrväinen, P., Cusumano, M.A., Jansen, S. (eds.) Software Business. First International Conference, ICSOB 2010, Jyväskylä, Finland, 21–23 June 2010. Proceedings, pp. 151–162. Springer, Berlin, (2010)
13. Dick, M., Drangmeister, J., Kern, E. et al.: Green software engineering with agile methods. In: Green and Sustainable Software (GREENS), 2013 2nd International Workshop on, pp. 78–85. 2013
14. Shenoy, S.S., Eeratta, R.: Green software development model: an approach towards sustainable software development. In: India Conference (INDICON), 2011 Annual IEEE, pp. 1–6. 2011
15. Käfer, G.: Green SE: Ideas for including energy efficiency into your software projects. In: Technical Briefing (TB2). 31st International Conference on Software Engineering, Vancouver (2009)
16. Mahaux, M., Canon, C.: Integrating the complexity of sustainability in requirements engineering. In: Svensson, R.B., Berry, D., Daneva, M. et al. (eds.) 18th International Working Conference on Requirements Engineering: Foundation for Software Quality. Proceedings of the Workshops RE4SuSy, REEW, CreaRE, RePriCo, IWSPM and the Conference Related Empirical Study, Empirical Fair and Doctoral Symposium, pp. 28–32. 2012
17. Penzenstadler, B., Khurum, M., Petersen, K.: Towards incorporating sustainability while taking software product management decisions. In: 7th International Workshop of Software Product Management, Essen, Germany, (2013)
18. Kocak, S.A.: Green software development and design for environmental sustainability. In: 11th International Doctoral Symposium an Empirical Software Engineering (IDOESE 2013). Baltimore, Maryland, 9 Oct 2013
19. Sahin, C., Cayci, F., Clause, J. et al.: Towards power reduction through improved software design. In: IEEE Energytech 2012. Cleeland, Ohio. IEEE, [Piscataway, N.J.] 29–31 May 2012
20. Shojaee, H.: Rules for being a green software engineer. ship software ontime! The blog that helps you build great software, (2007). http://shipsoftwareontime.com/2007/12/24/rules-for-being-a-green-software-engineer/. Accessed 26 Jul 2011
21. International Organization for Standardization: Software engineering—Software product Quality Requirements and Evaluation (SQuaRE)—Guide to SQuaRE 35.080(ISO/IEC 25000:2005 (E)). (2005)
22. Capra, E., Francalanci, C., Slaughter, S.A.: Is software green? Application development environments and energy efficiency in open source applications. Inf. Softw. Technol. **54**, 60–71 (2011)
23. Wang, S., Chen, H., Shi, W.: SPAN: a software power analyzer for multicore computer systems. SUSCOM **1**(1), 23–34 (2011). doi:10.1016/j.suscom.2010.10.002
24. Kansal, A., Zhao, F., Liu, J., et al.: Virtual machine power metering and provisioning. Proceedings of the 1st ACM Symposium on Cloud computing, pp. 39–50. ACM, Indianapolis, Indiana, USA (2010)
25. Dick, M., Kern, E., Drangmeister, J., Naumann, S., Johann, T.: Measurement and rating of software-induced energy consumption of desktop PCs and servers. In: Pillmann, W., Schade, S., Smits, P. (eds.) Innovations in sharing environmental observations and information. Proceedings of the 25th International Conference EnviroInfo, Ispra, Italy. Shaker, Aachen, pp 290–299. 5–7 Oct 2011

26. TPC—Transaction Processing Performance Council: TPC-Energy specification(TPC-Energy 1.2.0) (2010). http://www.tpc.org/tpc_energy/spec/TPC-Energy_Specification_1.2.0.pdf. Accessed 02 Sep 2011
27. Hilty, L.M.: Information technology and sustainability. Essays on the relationship between ICT and sustainable development. Books on Demand, Norderstedt (2008)
28. Mahmoud, S.S., Ahmad, I.: A green model for sustainable software engineering 2013. Int. J. Soft. Eng. Appl. **7**(4), 55–74 (2013)
29. Dick, M., Naumann, S. Enhancing software engineering processes towards sustainable software product design. In: Greve, K., Cremers, A.B. (eds.) EnviroInfo 2010: Integration of Environmental Information in Europe. Proceedings of the 24th International Conference on Informatics for Environmental Protection, Cologne/Bonn, Germany. Shaker, Aachen, pp 706–715. 6–8 Oct 2010
30. Johann, T., Maalej, W.: Position paper: the social dimension of sustainability in requirements engineering. In: Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems (2013)
31. Mahaux, M.: Could participation support sustainability in requirements engineering? In: Penzenstadler, B., Mahaux, M., Salinesi, C. (eds.) Proceedings of the 2nd International Workshop on Requirements Engineering for Sustainable Systems, Rio, Brasil, 15 July 2013. CEUR-WS.org