Enhancing Sustainable Software Engineering Methods through SECoMo

Natalie Buchner

Universitt Mannheim, Chair of Software Engineering,
D – 68159 Mannheim,

Abstract. Sustainability is a central topic governments, businesses and communities globally are dealing with today. Its most prominent aspect is ecological sustainability and the need to fight global warming, but it also concerns social and economic issues. Many factors come into play that have a negative impact on sustainability, for example an increase in energy consumption or pollution. The Information and Communication Technology (ICT) sector contributes to these negative factors as well, a main reason being the growing energy consumption caused by ICT hardware. But software can also have negative impacts on (mainly) ecological sustainability directly and indirectly. Thus, it is equally important to consider how to increase the sustainability of software.
The growing field of sustainable software engineering deals with the questions of how to develop sustainable software and how to develop it in a sustainable way. It covers aspects in all life cycle phases of a software. Existing research proposes a number of sustainability metrics, measurement tools or process models, but despite this variety of approaches, it seems that sustainable software engineering is not yet well established in practice.
The Software Eco-Costs Model (SECoMo) approach by Thomas Schulze [33] is a new estimation approach in this field which allows to estimate the ecological costs of software already from an early stage on in a software project and to represent those costs and their causes in a comprehensible and clear way. With this, it enables stakeholders to have an early understanding of the sustainability impact of a software and to make design decisions accordingly. [33]
The purpose of this seminar thesis is to consider how SECoMo can be integrated with other existing sustainable software engineering approaches and how it can contribute to improving sustainable software engineering in practice.
As SECoMo can be integrated in all development phases, especially the early ones, it can help to enhance existing life cycle models with a specific method for understanding and improving ecological sustainability in the design and implementation phases of software engineering. In addition, with its new set of sustainability metrics, SECoMo offers new options for sustainability measurement in existing models and tools, as they which base on a general way of software specification that can reasonably be applied in practice.

1 Introduction

Under the slogan "Sustainable Development Goals - 17 goals to transform our world"[23], the United Nations are raising awareness for Sustainability and the need for everybody's action in order to "shift the world on to a sustainable and resilient path" [38]. The 17 goals mentioned are a central part of the sustainable development agenda published by the United Nation in 2015 [38]and are essential in order to achieve sustainability with the main topics of "end[ing] poverty, protect[ing] the planet and ensur[ing] prosperitiy for all" [23].
These ambitious goals emphasize how important the topic of Sustainability has become for the world, which is facing global challenges such as extreme poverty, climate change or war and injustices. Despite the global relevance of these issues, the scope of these goals reveals that taking action for sustainability affects everybody: "governments, the private sector, civil society"[23] and every private person.

This is especially true when considering the causes for these challenges, spanning for example corrupt governments that cause injustice in their countries, or the ever increasing pollution and carbon dioxide emissions caused by various industries in the private sector. Many individuals and groups have a negative impact on sustainability with their actions - but on the other hand, there is also a great potential for positive impacts, starting from there.
Embracing sustainability has become more important in many areas over the last decades . Accordingly, many industries in the private sector have started to consider how to reduce their impacts on sustainability, often with a focus on ecological sustainability.

The ICT sector is no exception: with the products and services it provides it contributes to high power consumption and carbon dioxide emission and thus has a negative effect on global warming [37] - but in the last few years, many efforts have been made to counteract these negative impacts, for example by improving energy efficiency of hardware products or by creating software solutions that support more eco-friendly business processes.
However, the focus of these efforts is still mainly on reducing the negative impacts of hardware artifacts, during their production and, more importantly, usage phase. But software artifacts can have an impact on sustainability issues, too - not only by enabling people or processes using it to be more sustainable, but also as a product itself, which for example impacts energy consumption during usage, or even during its development process. Thus, efforts directed at improving software sustainability during its development and usage phases are equally important in terms of IT Sustainability.
There is a growing number of scientific contributions covering ideas for example on sustainability guidelines for software, models for sustainable development life cycle models, abstract software sustainability metrics or specific measurement tools for certain types of applications.

What is lacking are concrete methods, models and metrics and their evaluation in practical projects.

This paper aims at comparing a new approach in the field of Sustainable Software Engineering, SECoMo by Thomas Schulze [33], with the existing variety of models and approaches in the field. The goal is to work out how this approach for estimating the ecological costs of a software system can enhance existing approaches for sustainable software and its development.

The relevant background topics for this seminar thesis, Sustainability and the relationship between Sustainability and ICT, are introduced and defined in the first chapter. In the following, the related work from the field of Sustainable Software Engineering is described, covering the current state of research and the most important existing approaches. In addition, the SECoMo approach is introduced and described with a focus on its general purpose and motivation as well as its most relevant elements and how to apply SECoMo within a software development process.
In the next section, SECoMo is then considered in the context of Sustainable Software Engineering and analyzed regarding its ability to enhance existing approaches in this field, in order to identify its benefits for the field of Sustainable Software Engineering.
Lastly, limitations of SECoMo in the context of Sustainable Software Engineering are presented and a conclusion of the paper is presented.

2 Background

2.1 The Concept of Sustainability

Although it is clear that Sustainability is a concept of increasing importance in the world and it has wide-ranging influences, on the way countries are lead, companies are strategically aligning themselves(?) or which causes societies are engaging for(?) for example, the definition of the term Sustainability itself is not as clear and not always unified / consistent / coherent?: There are various attempts at understanding and defining the concept of Sustainability, coming from different perspectives which influence the focus of the definition (e.g. the needs of humans or of nature [13]) and thus it can be hard to find a common understanding [16]. In order to successfully work towards a sustainable future, finding this common understanding is absolutely necessary, though [16].

However, most attempts at defining the concept of Sustainability start with the definition of the concept of *Sustainable Development* by the United Nations World Commission on Environment and Development in 1987, also known as the *Brundtland Report* : "Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs"[25] . In addition, the requirements for sustainable development according to [25] are related to three different aspects: society, economy and environment. These three aspects are now widely accepted as "the three dimensions of sustainable development" [38].

Based on this understanding of what Sustainable Development is and which dimensions it covers, the concept of Sustainability can be defined as "a holistic concept that embraces environmental, social and economic factors which lead to a decent life for the current generation while maintaining natural, social and economic resources so that future generations are not limited in living the same decent life" [8].This definition recognizes Sustainability as a concept that enables Sustainable Development in all its dimensions which a focus on human needs, but not limited to it.

2.2 Sustainability and IT

The ICT sector has a major influence on the way economies and societies work, companies deliver their business and the way people live their daily life. As mentioned before, it is obvious that ICT also plays an important role in the context of Sustainability: on one hand, as negative contributor to global warming for example, due to its increasing carbon dioxide emissions, or due to the growing problem of e-waste. On the other hand it also has a great potential to have positive impacts on Sustainability, by supporting efforts to reduce the energy consumption of its own products for example, but also as an enabler to achieve energy efficiencies in other sectors [37].

The energy consumption of hardware is one of the main direct impacts ICT has on sustainability, and while these direct impacts are often quite clear, indirectly caused effects of ICT (like the ability to enable improvements in other industries) are often overlooked, even though they can have even greater impacts in the long run. In order to better illustrate the relationship between ICT and Sustainability, a more fine-grained differentiation of the effects of ICT is often used in the literature (for example by [14] or [21]) which distinguishes between three types: First-order effects, second-order effects and third-order effects [6]. While first-order effects refer to direct impacts "of the physical existence of ICT" [14], like resource consumption during the production or usage phase of hardware, second-order effects describe rather indirect effects that are caused by using ICT, like the optimization of processes which might lead to resource conservation [21]. The third-order effects refer to impacts observable on a long-term scale, like the "adaptation of behaviour (e.g. consumption patterns) or economic structures" [14] which themselves lead to certain impacts on different aspects of Sustainability.

If ICT can have negative and positive impacts on Sustainability on so many different levels, the question that comes to mind is: What are concrete approaches so that ICT can actually be used "in the service of Sustainability"[15]?
According to Hilty et al. [15], relevant approaches in the field of ICT that contribute to Sustainability in one way or another are Environmental Informatics, Sustainable Human-Computer Interaction (HCI) and Green IT[1]. Of those, Environmental Informatics and Sustainable HCI are rather small, specific sub fields, while Green IT is a wide ranging area with many scientific contributions that has also been continuously adapted by companies and organizations since the

[1] Also: Green ICT. In the following the term *Green IT* will be used in order to align with the literature.

publication of the Gartner Report *Green IT: A New Industry Shock Wave* [20], which calls attention to the need for more long-term sustainability in companies through Green IT practices.
The sub field of Environmental Informatics deals with applications that process environmental information in order to tackle environmental problems [15]. Thus, it contributes to Sustainability by providing tools that enable users to better analyze and understand environmental issues, share environmental data, and so on, and is therefore related to second-order ICT effects. The sub field of Sustainable HCI deals with the connection of software design, human interaction with it, and Sustainability [15]. This covers two categories [15]: Sustainability *In* software design (how sustainable is the use of the designed software itself) and Sustainability *Through* software design (how can the design of the software "promote sustainable behaviour" [15]). This approach therefore relates to first- and second-order impacts of ICT.
The approach of Green IT is slightly more complex, as it refers to various aspects of the lifecylce of ICT products but also related areas that are impacted by ICT (for example the creation of sustainable business processes or practices) . Essentially, Green IT also covers two categories: Green *In* IT and Green *By* IT [15]. While Green In IT refers to ICT as resource consumer and carbon emission producer itself and deals with the question how to improve the environmental sustainability of ICT hardware [9], Green By IT aims at improving the environmental sustainability of aspects in different areas (other products or processes) by using ICT as the enabler [15]. Especially the category of Green In IT is a very broad field that covers the whole life cycle of ICT products, from its production over the usage phase until its disposal or recycling phase, and it is important to consider impacts of ICT in all these life cycle phases in order to achieve holistic environmental sustainability [15]. Green IT is related to first- and second-order effects of ICT, but as a strategic concept adapted by many companies and organizations, it has the potential to create long-term, third-order effects.

Even though ICT products consist not only of hardware artifacts but also software artifacts, the main focus of Green In IT is usually on improving the sustainability of ICT hardware, and often Green By IT approaches also do not take the impacts of software artifacts fully into account.

2.3 Sustainability and Software

But software artifacts actually play an equally important role in the relationship of ICT and Sustainability: Although, at the first glance, it may seem that software as the immaterial part of the ICT products does not have an impact on Sustainability, especially environmental sustainability, and "is automatically green" [1], this naive assumption is wrong: when software is developed, a lot of physical resources are needed during this process; when software is used, it has impacts on other aspects of its surroundings, for example by processing environmental data to better understand Sustainability challenges; when software is deactivated, dealing with saving or converting its data might have economic impacts on a company for example [17]; and it can even have long-term third-order effects like changing user behavior towards more sustainable practices like video conferencing instead of traveling to business meetings [4]. During its usage phase, it is true that the software does not directly consume energy - but if software is regarded as "the ultimate cause of hardware requirements and energy consumption" [19], it becomes clear that it is after all indirectly influencing also energy efficiency aspects. Considering all these impacts, software definitely needs to be taken into account as an equally important part of ICT Sustainability considerations.

Overall, software thus has direct and indirect impacts on Sustainability just like ICT in general, either as the producer / consumer itself or as enabler for improvements in other areas. Accordingly, it is clear that also for the ICT sub field of software engineering there is a need for dealing with topics concerning Sustainability. But so far, Sustainability considerations are not a part of "traditional software engineering" [27] methods. This is critized by a growing number of authors ([27]; [1]) , as the awareness for the importance of Sustainability in Software Engineering grows, and the emerging field of *Sustainable* or *Green Software Engineering*[2] aims at filling this gap by working on approaches and methods that deal with sustainable software and its engineering.

Returning to the question of how ICT can be in the service of Sustainability, Sustainable Software Engineering constitutes another important approach that should be added to the list.

In the literature, there does not yet exist one common definition of Sustainable Software and Sustainable Software Engineering [39]. This is a result of the early stage of the research field, the complex nature of the concept of sustainability and software and the fact that different perspectives exists from which these definitions are created [39]. Nevertheless, for the purpose of this paper the mentioned concepts need to be defined, so we chose the definition by Dick et al. [12] for Sustainable Software, as it best relates to the three dimensions of Sustainability:

> "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" [12]

Accordingly, Sustainable Software Engineering can be understood as the systematic methodology to develop such Sustainable Software. It is defined by Naumann et al. as

> " [...] Sustainable Software Engineering is the art of developing [...] sustainable software with a [...] sustainable software engineering process.

[2] In the following, the term *Sustainable Software Engineering* will be used, as it comprises the aspect of environmental (green) aspects as well as social and economic aspects, which are equally important parts of Sustainability considerations, even though environmental aspects are the majority.

Therefore, it is the art of defining and developing software products in a way, so that the negative and positive impacts on sustainable development that result [...] from the software product over its whole life cycle are continuously assessed, documented, and used for a further optimization of the software product" [21]

This definition is providing a holistic description of the idea of sustainable software engineering, especially due to its focus on the impacts the software product produces over its whole life cycle. In this context, Penzenstadler [27] provides a categorization of Software Sustainability aspects that reflects the variety of impacts software can have. It differentiates between four aspects: the development process aspect, the maintenance process aspect, the system production aspect and the system usage aspect [27]. The first two aspects represent the development process viewpoint and describe the sustainability of the actual design and development phase (development process aspect) and the sustainability of the subsequent maintenance phase with aspects like monitoring or bug fixing (maintenance process aspect) [26]. The other two aspects constitute the product viewpoint and describe the sustainability of the actual software product during its production phase with regards to the resource consumption (System production aspect) and the sustainability of using the software product, in terms of indirect impacts triggered by its usage [26].
This categorization will be used in the following in order to describe and categorize existing contributions in the field of sustainable software engineering.

3 Related work in Sustainable Software Engineering

3.1 State of Research

The first considerations of Sustainability in combination with Software come from authors like Seacord et al. [34] or Tate [36] in the middle of the 2000s - but these authors mainly consider the Sustainability aspect of software in the context of software maintenance and how the longevity of software can be supported, instead of taking environmental and social impacts of Software into account [3]. When the concept of Green IT became popular in the middle of the 2000s and more attention was drawn to the impacts that ICT has on environmental sustainability ([6], [14]), also the awareness for the role of software specifically grew and was accepted to be "worth a greater attention" [11].

But as indicated before, neither environmental Sustainability nor Sustainability in general is actually an aspect that is integrated in traditional software engineering approaches and concrete guidance on how to support it in this field was missing, according to [27]. Consequently, there is/was a need for the definition of Sustainable Software, guidelines and approaches how to develop and use it and metrics and measurement methods in order to measure its impact.

According to a Systematic Literature Review (SLR) conducted by Penzenstadler et al. [26] in 2012, the research activity in the field of sustainable software engineering has mainly started in the beginning of the 2010s and has significantly

increased since then. This is also underlined by the follow-up study from 2014 [30], which identified around 40 relevant publications that were addedin the time between the two studies.
The main insights provided by these two studies can be summarized as follows: while there are many contributions over the last years that have to do with software and sustainability in some way, most research still goes into "domain-specific, constructive approaches" [26] and general reference approaches that can be applied to the whole domain of sustainable software engineering are missing [26]. The study from 2014 identified already more contributions that focused on general aspects in the areas of Software Engineering Processes and Software Design [30], but identified that most application domains covered by the contributions are still concerning fields not directly connected to Software Engineering (for example Business and Economics, Nature and Agriculture, Mechanics and Manufacturing [30]). Moreover, both studies identified a lack of case studies and experience reports about applications of proposed approaches and methods in practice [26], which shows that the research field is still mostly shaped by academia and the practical view is missing [30].

Overall, there are certain aspects in the current research that stand out because they are targeted by many authors:
The first aspect is the need for a unified definition of the concepts of sustainable software and the role of sustainability in software Engineering. Many authors have taken on the task to identify the one common understanding of Sustainability in Software Engineering and to extract a unified definition ([39];[5]; [9] ; [28]), but they all come to the conclusion that despite the variety of attempts, there does not exist one commonly accepted definition yet. According to [39] for example, many attempts are often too vague or not broad enough, and due to the complex nature of the concept of software sustainability, all attempts into account slightly divergent perspectives. The criticality of this "lack of [a] common understanding of the fundamental concepts of sustainability" [5] in the context of software and software engineering is underlined by Venters et al. [39], who state that until a commonly accepted definition is established, most efforts in the field will be likely to "remain insular and isolated" [39].

A second aspect that is often mentioned is a more concrete idea to tackle Sustainability considerations in Software Engineering: Many authors emphasize the need to integrate Sustainability as a software quality attribute by making it a Non-Functional Requirement (NFR) in official software engineering standards ([31]; [4]; [1]). This step would account for the central role sustainability plays today [31] and help to integrate sustainability "early in the lifecycle of software development and influence design decisions" [32].
According to [31] and [39], in order to make Sustainability a NFR, there is still a lot to do: there is for example a need for concrete methods to perform sustainability requirements analysis, existing policies and standards need to include sustainability aspects and assessment techniques and sustainability metrics need to be defined in order to enable sustainability assessment.

Existing attempts at providing such methods and metrics will be presented in the next section.

Furthermore, the aspect of missing guidance for the efforts of including sustainability in software engineering is often addressed in the existing literature, as mentioned before. This is observable in the concerns brought forward that especially generic reference models and measurement methods are missing that can be applied to the whole field, as identified for example by [26]: "An encompassing reference framework for SE is still missing". In addition, the need for concrete, unified metrics that enable to measure the actual impact of software on sustainability is also evident .
To meet this demand, more and more contributions from the past five to six years aimed at providing concrete models and metrics to be used in the context of sustainable software engineering. The most relevant ones will be presented in the next section.

A quite recent contribution that tries to tackle the issue of missing guidance in the field is the *Karlskrona Manifesto for Sustainable Design* [2] from 2015. This Mainfesto presents "the fundamental principles underpinning design choices that affect sustainability" [5] and with its creation, the authors aimed at finally providing "a common ground and a point of reference"[5] for the research community of sustainable software. With this, a first starting point is given to align efforts in the area of sustainable software engineering towards a common understanding.

3.2 Overview of existing contributions to Sustainable Software Engineering Practices

In the following, some existing approaches of different aspects of Sustainable Software Engineering are presented to give an overview about suggested practices the field. They are divided into approaches regarding sustainability metrics and measurement methods, approaches covering software life cycle and process models and approaches that specifically target requirements engineering (RE) aspects.

Sustainability metrics and measurement methods A variety of contributions to sustainable software engineering focuses on methods and tools to measure energy consumption of software. One of the first tools introduced was *GreenTracker*, which aims at making users aware of the impacts software has on environmental sustainability by tracking its energy consumption (based on CPU usage) [4]. Another tool that also aims at raising awareness for power consumption by software, more specifically websites, is the *Power Indicator* tool by Naumann et al. [22] - in detail, it depicts if the server of a website is run with renewable energy. Other approaches like *PowerAPI* by Noureddine et al. [24] provide more comprehensive frameworks for measuring energy consumption. PowerAPI in particular enables the monitoring of energy consumption of an application during runtime, and has the specialty to also consider the impact different programming languages have on the power consumption [24].

Apart from these measurement tools, there also exist approaches that specifically introduce mathematical metrics dealing with energy consumption. Some examples that have a specific focus on energy *efficiency* are provided by [10] and [18] - their metrics consider the relative energy consumption of applications (for example in relation to the "useful work done" [18]), and in addition they do so on a very low level, which makes it possible to "find resource intensive parts of programs and improve them" [18]. The described set-ups for testing each of the metrics are quite complex though, which indicates that there is still some work to do before these metrics can be applied in practice [18].
Another interesting contribution in this area comes from Kern et al. [19], who propose a metric and the respective calculation method for the carbon footprint of a software project. It is a relatively new contribution to the field of sustainable software engineering and thus aims at finally providing a practical approach for measuring the environmental impact of software which can be integrated in a software development process [19]. The basic idea is to calculate the respective current footprint of the development process of a software project or of the software product itself [19]. The metric itself is expressed as kg CO2 per person month and is calculated based on rather fuzzy inputs like number of working days, number of employees, IT infrastructure and so on [19].
Nearly all of the existing metrics and measurement methods focus exclusively on power and energy consumption and thus mainly target environmental sustainability. Nevertheless, there have also been efforts to consider software impacts in relation with other dimensions of sustainability. The most prominent contribution in this area is the list of Sustainability properties and related metrics provided by Albertao et al. [3]. Based on existing measurements of software quality, which were evaluated with regards to their relevance for the different sustainability dimensions, eleven properties were derived that include aspects like Modifiability, Usability, Efficiency or the Project's Footprint [3]. Furthermore, this contribution proposes a sustainability measurement method which includes assessing the sustainability metrics after the release of a software in order to derive goals for improvement for the next development cycles, thus promoting continuous improvement [3].

Software life cycle, process and other software sustainability models With regards to approaches that propose actual systematicmodels in the field of sustainable software, there is a great variety, spanning models for very specific purposes and application domains to rather general reference models for sustainability aspects. Such a general model is for example proposed by Hilty et al. [14] with the purpose of assessing the high-level application fields where ICT can have an impact on environmental sustainability. Another model that addresses a quite different aspect is the Sustainability model by Penzenstadler and Femmer [29] - it is a reference model that allows to specify Sustainability goals and related values and activities from either a company or product specific point of view [29]. Thus, it facilitates the general analysis of sustainability aspects in different contexts, but also enables the definition of sustainability aspects in relation to a software product, for example.

There is a certain type of systematic sustainability models that is targeted by many contributions for sustainable software: models with a focus on the Software Development Life cycle (SDLC) and general process models - as these models are very relevant in the field of Software Engineering in general. Most of these models concentrate on enhancing existing models, such as the waterfall model for software development, with activities and best practices that improve the sustainability of the software development process itself in all its phases. Examples are the models by Agarwal et al. [1] and Shenoy and Eeratta [35], which include aspects like simply reducing resource usage like paper [35], but also consider activities like writing energy-efficiency code [1]. Furthermore, they emphasize the need to integrate Sustainability considerations in the requirements phase and to use respective measures to ensure sustainability and software quality ([1]; [35]). The conceptual model for sustainable software systems engineering by Betz and Caporale [7] for example has a slightly different focus - it concentrates on integrating the engineering life cycle of software systems with the life cycle of Sustainable process engineering to emphasize the importance of both processes in order to tackle Sustainability issues [7].

Besides all these mentioned approaches, there is one software sustainability model that seems to become more and more important in the research area, as it is highly referenced in many related contributions: *GREENSOFT*[21] by the research group GreenSoft at the Trier University of Applied Sciences. This "reference model for green and sustainable software and its engineering" [21] is a comprehensive model that combines many aspects of sustainable software engineering and can be considered to be currently the best starting point for applying sustainable software engineering to practical projects: It consists of

- a life cycle model of software products; this is used to assess the impact of the software during its different phases like development or usage; [21]
- a set of sustainability criteria and metrics, based on those proposed by Albertao et al. [3]; [21]
- different procedure models that specify in more details how sustainability can be integrated in the different phases of software engineering (like the development phase), but also related procedures like administration or purchase [21]
- and a collection of further recommendation for the different stakeholders; this includes suggestions of external sources for sustainable programming or the call for a general knowledge base for software sustainability best practices [21].

Overall, this complex model aims at providing information on sustainable software, its engineering, use, administration and so on for different stakeholders and is thus a very important source in the research field. But nevertheless, it is not the perfect solution yet that directly enables to integrate sustainability in practical projects - the development procedure model for example concentrates on describing how to manage sustainability during the development process, but it does not specifically say how to realize it with the actual development. For this, the integration of other tools or approaches is needed that for example specify how to calculate certain sustainability metrics or that allow estimation of processing time or energy consumption in early stages [21] .

Sustainability approaches suitable for requirements engineering Recently, also the focus on approaches that integrate Sustainability aspects in the subfield of Requirements Engineering has intensified. The Sustainability reference model by [29] is a good example for an approach that is specifically targeted for this area - as the authors state, "[i]t is intended to serve as a reference model [...] for a requirements engineer" [29] . Further approaches in this area for example go into the direction of creating a "Sustainability Non-Functional Requirements Framework" [32] that supports to formulate and classify sustainability requirements.
In addition to these directly related approaches, it is also possible to consider existing approaches from other areas for the field of RE. As mentioned before, for the promotion of Sustainability as NFR respective assessment models and metrics are needed: The proposed sustainability properties and related metrics by Albertao et al. [3] could for example be used here in order to assess sustainability in all its dimensions; if the focus is on environmental sustainability requirements only, also approaches like the carbon footprint calculation method by [19] could be used.

4 The SECoMo approach

A relatively new addition to the field of sustainable software engineering approaches is the **Software Eco-Cost Model** (SECoMo) Approach by Thomas Schulze [33]. This approach provides Software Engineers with generic models and metrics necessary to estimate and express the ecological costs a software system causes when it is used [33]. Thus, SECoMo represents a concrete estimation approach for the impact a software system has regarding ecological sustainability during its usage phase.

4.1 What is SECoMo about?

The main motivation behind SECoMo is to provide an approach that allows to not only measure the ecological costs that are actually caused by a software system, but also to be able to estimate those costs upfront, for example already during the design phase of a software engineering project [33]. In order to achieve this, SECoMo offers a set of mathematical models which allow to precisely calculate eco-cost metrics, based on information that is already available in the design phase: specification models that describe the functionality, behavior and structure of the software system [33]. Furthermore, SECoMo is intended to be highly adaptable in order to allow the estimates to be calculated for different levels of details available - from an early level where only very general information about the software system is available, over an intermediate level with partly more detailed information, to an advanced level with very specified details that allow for more accurate estimates [33].

In addition to the mathematical models, the SECoMo approach also defines a set of eco-cost drivers in order to identify causes for certain ecological impacts a software system has and to better describe under which circumstances they occur [33]. The auxiliary models used in SECoMo which extend the specification models provide information about these cost drivers, but can also be used to express the estimated eco-costs of the software system [33]. This way, SECoMo additionally offers a possibility to communicate estimated or measured eco-costs to stakeholders of a project which can use this information to make improved decisions [33].
Against this background, the SECoMo approach is intended to be used in the early stages of software engineering projects to create estimates about the ecological impact of a software system, so as to enable transparency about the sustainability aspect right from the start [33]. This again makes it possible for software engineers and other stakeholders to make decisions about changes to the software at the design stage which take the impact on ecological costs into account - be it to improve certain eco-cost critical aspects of the software because ecological sustainability is a major concern, or to at least be aware of the eco-cost trade-offs other decisions cause that might be motivated by other concerns, e.g. profitability. [33]
But SECoMo can also be used in the context of defining requirements for a software system, for example in terms of specifying upper bounds for the eco-cost metrics that must not be exceeded, or even to calculate exact eco-costs if enough details are given, e.g. for the specific usage scenario of a software system in a certain environment [33].

4.2 How can SECoMo be integrated in a software engineering project?

On a very high level, the SECoMo approach, when being used within a software engineering project in order to estimate the eco-costs, can be structured as follows:
In order to start with the SECoMo approach, the initial design phase of the software needs to be in a stage where it is already clear on some level how the software will be structured, which behavior it should follow and what functionalities it should provide. Ideally, this is reached by using a modeling approach that creates a structural, behavioral and functional view on the modeled software. SECoMo does not require a specific modeling approach that needs to be used to create these views, but these model types are the basis for the further steps of using SECoMo as estimation technique [33].
If this basis is given, the following steps of the SECoMo approach can be integrated already in the earliest stages of a software development process (if early eco-cost estimations are required), otherwise also in later iterations of the process to refine the initial estimates or to calculate the actual eco-cost values (for example in the testing phase):

1. Calibration of the models, in order to receive accurate values concerning the hardware energy consumption factors (resource factors) that are later needed as inputs to the mathematical models [33];
2. Preparation of the auxiliary models so that they include all information available regarding the relevant eco-cost drivers (on the intermediate and advanced level certain information can be derived using Markov Models) [33];
3. And finally Calculation of the estimates for the different eco-cost metrics based on the available information and the specified parameter values, which identify aspects like user type or case [33].

Based on the estimated or calculated eco-costs it is also possible to derive specific reports on the environmental impact of the software for different types of stakeholders, or to simply communicate the eco-costs via the existing auxiliary models [33]. This way, it is possible for stakeholders to make informed decisions on the software design taking the eco-cost information into account, and especially doing so at an early level [33].

5 SeCoMo in the context of Sustainable Software Engineering

Classification In order to analyze how SECoMo can contribute to the field of Sustainable Software Engineering and how it can enhance existing approaches, the following question needs to be answered: How does SECoMo relate to the field of Sustainable Software Engineering and how can it be classified?
Recalling the definition of Sustainable Software Engineering used in this paper, it becomes evident that the SECoMo approach aligns with many of the basic ideas: SECoMo is an approach that can be used within a **software engineering process** to support design decisions; it deals with attributes of **software products**, namely its ecological costs during the usage phase (part of the software **life cycle**); as an adaptive estimation and calculation approach it deals with **continuously assessing and documenting** eco-cost, which are relevant in the context of **negative and positive impacts on sustainable development**, mainly for environmental sustainability; and it has the general goal of **optimizing the software product**.
Due to this strong similarities SECoMo can be described as an approach that clearly fits into the field of Sustainable Software Engineering, while not as an engineering process itself, definitely as a method that enhances and supports aspects of such a process with regards to sustainability.
More specifically, SECoMo can be classified with regards to the four aspects of Software Sustainability impacts proposed by [27]: Due to its focus on eco-costs of the software product and the fact that those occur whenever the software is used, SECoMo belongs to the approaches that exclusively focus on the **system usage aspect**. It does not consider eco-costs that appear during the development or maintenance process itself, nor those that are caused by producing the software

product. Nevertheless, as stated by [28], the system usage aspect is the most relevant one as it has the biggest impacts, especially the more the software is used - thus, SECoMo is an approach that is directed at a very important aspect in sustainable software engineering: the sustainability of the software product itself.
The next sections deal with the question what SECoMo can contribute to the different areas of approaches and if it can possibly even enhance some of them.

SECoMo and Sustainability Metrics Like the majority of the presented sustainability metrics in this paper, SECoMo also focuses entirely on environmental aspects of Sustainability, the ecological costs of software in particular. The eco-cost metrics that are part of SECoMo range from a very fine-grained level (for example the ecological costs of the execution of one single operation [33]) to a rather broad scope (for example the ecological costs of a concrete execution scenario of the software operations [33]), and are thus suitable for a variety of different purposes for which eco-costs are regarded. Most of the presented metrics and measurements are only usable in a specific context (like [22] or [10]).
What the SECoMo metrics do not provide are information about energy efficiency (like the metrics defined by [10] and [18]), and as they are estimated or calculated without information about the concrete implementation, they can not directly be used to identify resource intense coding parts like those two metrics can - but the SECoMo metrics can give even more useful insights, as they can be calculated before any implementation is even considered and still give information which parts of the software might be very resource intense (for example the execution of a certain operation, or updating a certain data type [33]).
The goal and motivation of SECoMo actually resembles those of the presented carbon footprint calculation method by Kern et al. [19]: they share the motivation of providing a concrete method to calculate the environmental impact of a software product, that can be integrated in software development processes in a reasonable way ([19]; [33]). Moreover, the authors mention in their motivation the need for creating awareness and transparency for the aspect of software energy consumption [19] (which SECoMo aims at, too, mainly with regards to the stakeholders [33]). Finally, the Kern et al. emphasize that the benefit of their method lies in its ability to finally integrate sustainability aspects early in the software development and design process [19] - which is one of the main benefits of SECoMo, too [33].
The differences between the two approaches however are also clear: While the Carbon Footprint metric by Kern et al. has a fixed unit (kg CO2 per person month [19]) and is thus restricted to a certain context, the SECoMo metrics can vary in the units they represent in the context of eco-costs (for example CO2 emissions, but also dollars [33]) and are thus far more flexible. Furthermore, they can adapt to many different contexts due to the variety of aspects from different dimensions they can cover [33], and they are not restricted to the high-level inputs the Carbon Footprint metric requires [19], but can be way more fine-grained [33]. The only aspect where SECoMo is more restricted that the Carbon Footprint calculation method is the scope: while the latter method can calculate

the footprint of the software product itself and in addition the footprint of the development process, too [19], SECoMo only applies to the eco-costs caused by the software product itself [33].
Overall, these differentiation aspects underline the general benefits of the SECoMo eco-cost metrics in relation to existing sustainability metrics: they are more flexible and can be applied for different contexts and are thus more adaptive and they allow an early understanding of causes for inefficient energy consumption of a software product, as early as in the design phase.

SECoMo in sustainability models As outlined in the previous section, SECoMo can be integrated into traditional software engineering processes by adding the estimation or calculation steps in the design phase, but also later phases. Thus, it is clear that it should also be possible to integrate SECoMo into sustainable software engineering processes and SDLC models that are enhanced with sustainability activities (like the ones proposed by [1] and [35]), if those are adapted to the steps of SECoMo.
A very interesting question is whether it makes sense to *add* SECoMo to the GREENSOFT model. As GREENSOFT is a reference model that comprises a variety of approaches [21], SECoMo can not simply be integrated *into* the model, but it needs to be considered at which level it makes most sense. The general concept of eco-costs in SECoMo relates to the first level of GREENSOFT, with the life cycle model that is used to assess the impacts of the software in different phases [21] - the eco-costs are impacts that occur during the usage phase. With regards to the second level, the Sustainability Criteria and Metrics included in GREENSOFT could be extended by adding SECoMo's eco-cost metrics as part of the environmental metrics. But where SECoMo actually fits best is the third level of GREENSOFT, the procedure models [21]: For the development procedure model, so far there only exists a process model which suggests ways to manage sustainability during the development process as mentioned. What is missing is a concrete method how to include the measurement of Sustainability aspects in this development procedure, and as the authors note themselves, additional aspects like for example early estimations of the energy consumptions may enhance their proposed model [21] - SECoMo thus is the perfect addition to the development procedure model of GREENSOFT, especially because it enables estimations at such an early stage like the design phase which makes resulting changes way cheaper [33].
So overall, SECoMo is a suitable method that could be added to the GREENSOFT model, as it enhances it by providing a concrete approach not only to measure sustainability metrics (here: eco-costs), but also to estimate these eco-costs early, before even the first deployable software is available. With this unique quality the SECoMo approach can also contribute to Sustainable Software Engineering practices in general. In addition, the SECoMo approach is likely to be applied in practical projects due to its clear structure and the high adaptability, and thus might help to finally trigger the general adoption of sustainable software engineering models in practice.

SECoMo and Requirements Engineering for Sustainability With regards to Requirements Engineering, SECoMo does not provide an RE approach per se - but it certainly enables the integration of environmental sustainability aspects in this field, if appropriate elements from the SECoMo approach are used: The existing eco-cost metrics provided by SECoMo are ideal in order to be used as basis to formulate environmental requirements a software must fulfill and to fix required thresholds [33]. Moreover, the auxiliary models created during the SECoMo process can be used in order to define eco-cost requirements and in order to communicate them to stakeholders [33].
The actual estimations or calculations derived from the mathematical models of SECoMo on the other hand are adequate assessment measures to check if the defined requirements are fulfilled. This way, SECoMo offers concrete solutions to some of the issues that are present for Sustainability as a (non-functional) requirement (lack of assessment methods for example, as mentioned by [31]).
In conclusion, SECoMo enhances RE approaches in the context of Sustainability with a vocabulary / models to define and communicate requirements and a also a suitable way to assess them.

5.1 Limitations

Even though the SECoMo approach offers many new aspects to the field of sustainable software engineering and can enhance certain approaches in a positive way, there are a few limitations to the benefit of SECoMo for sustainable software engineering:
SECoMo is an approach that focuses on the estimation and calculation of software eco-costs. With that, it has a clear focus on environmental sustainability and is thus also mainly enhancing the field of Sustainable Software Engineering in terms of this dimension. In a way, it also refers to economic sustainability, to the extend that it comprises for example considerations like the high costs of changes to a software product during late phases of the development process, or the possibility to express eco-costs in a monetary unit [33]. But to be considered an approach that supports sustainability in a comprehensive way, some kind of social sustainability component would be necessary. This limitation however is qualified by the fact that most approaches that are considered to belong to the field of sustainable software engineering actually focus on environmental sustainability, thus SECoMo is not an exception.
Another critical aspect is the assumption that SECoMo can have a major impact on the readiness for sustainable software engineering approaches to be adapted in practical projects - this is only an assumption based on the concrete nature of the approach itself and the fact that it allows to be used even with only a small amount of input information available [33], which reduces the obstacles for a practical adoption of the method. In order to actually see if this assumption holds true, it is necessary to first perform further evaluation of the SECoMo approach itself.

6 Conclusion

References

[1] Shalabh Agarwal, Asoke Nath, and Dipayan Chowdhury. Sustainable approaches and good practices in green software engineering. *International Journal of Research and Reviews in Computer Science (IJRRCS)*, 3(1), 2012.

[2] Sedef Akinli-Kocack, Christoph Becker, Stefanie Betz, Coral Calero, Ruzanna Chitchyan, Leticia Duboc, Steve M. Easterbrook, Martin Mahaux, Birgit Penzenstadler, Guillermo Rodriguez-Navas, Camille Salinesi, Norbert Seyff, and Colin C. Venters. The karlskrona manifesto for sustainability design, 2015.

[3] F. Albertao, J. Xiao, C. Tian, Y. Lu, K. Q. Zhang, and C. Liu. Measuring the Sustainability Performance of Software Projects. In *2010 IEEE 7th International Conference on E-Business Engineering*, pages 369–373, November 2010.

[4] Nadine Amsel, Zaid Ibrahim, Amir Malik, and Bill Tomlinson. Toward sustainable software engineering (NIER track). In *International Conference on Software Engineering*, May 2011.

[5] Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve M. Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C. Venters. Sustainability design and software: the karlskrona manifesto. In *International Conference on Software Engineering*, May 2015.

[6] Frans Berkhout and Julia Hertin. Impacts of information and communication technologies on environmental sustainability: Speculations and evidence. *Report to the OECD, Brighton*, 21, 2001.

[7] Stefanie Betz and Timm Caporale. Sustainable Software System Engineering. In *Cloud Computing*, December 2014.

[8] Natalie Buchner. Understanding sustainable software - of impacts and development models. Seminar paper, School of Economics and Management, Lund University, Sweden, 2016.

[9] Coral Calero and Mario Piattini. *Green in Software Engineering.* Springer Publishing Company, Incorporated, 2015.

[10] Eugenio Capra, Chiara Francalanci, and Sandra A. Slaughter. Is software green? Application development environments and energy efficiency in open source applications. *Information and Software Technology*, 54(1):60–71, 2012.

[11] Eugenio Capra and Francesco Merlo. Green IT: Everything starts from the software. *ECIS 2009 Proceedings*, January 2009.

[12] Markus Dick, Stefan Naumann, and Norbert Kuhn. A Model and Selected Instances of Green and Sustainable Software. In *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience*, pages 248–259. Springer, Berlin, Heidelberg, 2010. DOI: 10.1007/978-3-642-15479-9_24.

[13] Thomas N. Gladwin, James J. Kennelly, and Tara-Shelomith Krause. Shifting paradigms for sustainable development: Implications for management theory and research. *Academy of management Review*, 20(4):874–907, 1995.

[14] Lorenz M. Hilty, Peter Arnfalk, Lorenz Erdmann, James Goodman, Martin Lehmann, and Patrick A. Wger. The relevance of information and communication technologies for environmental sustainabilitya prospective simulation study. *Environmental Modelling & Software*, 21(11):1618–1629, 2006.

[15] Lorenz M. Hilty, Wolfgang Lohmann, and Elaine M. Huang. Sustainability and ICT An overview of the field, 2011.

[16] Dale Jamieson. Sustainability and beyond. *Ecological Economics*, 24(2):183–192, 1998.

[17] Timo Johann, Markus Dick, Eva Kern, and Stefan Naumann. Sustainable development, sustainable software, and sustainable software engineering: An integrated approach. In *Humanities, Science & Engineering Research (SHUSER), 2011 Symposium*. IEEE, June 2011.

[18] Timo Johann, Markus Dick, Stefan Naumann, and Eva Kern. How to Measure Energy-efficiency of Software: Metrics and Measurement Results. In *Proceedings of the First International Workshop on Green and Sustainable Software*, GREENS '12, pages 51–54, Piscataway, NJ, USA, 2012. IEEE Press.

[19] Eva Kern, Markus Dick, Stefan Naumann, and Tim Hiller. Impacts of software and its engineering on the carbon footprint of ICT. *Environmental Impact Assessment Review*, 52:53–61, April 2015.

[20] Simon Mingay. GREEN it: A new industry shock wave, 2007.

[21] Stefan Naumann, Markus Dick, Eva Kern, and Timo Johann. The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, 1(4):294–304, December 2011.

[22] Stefan Naumann, Sascha Gresk, and Kerstin Schfer. How green is the web? Visualizing the power quality of websites. *EnviroInfo*, pages 62–65, 2008.

[23] Florencia Soto Nino. Sustainable development goals - United Nations, 2017.

[24] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier. A preliminary study of the impact of software engineering on GreenIT. In *2012 First International Workshop on Green and Sustainable Software (GREENS)*, pages 21–27, June 2012.

[25] World Commission on Environment and Development. Our common future. also: Brundtland report, United Nations, 1987.

[26] B. Penzenstadler, V. Bauer, C. Calero, and X. Franch. Sustainability in software engineering: a systematic literature review. pages 32–41, January 2012.

[27] Birgit Penzenstadler. Supporting sustainability aspects in software engineering. In *3rd international conference on computational sustainability (CompSust)*, 2012.

[28] Birgit Penzenstadler. What does Sustainability mean in and for Software Engineering? (PDF Download Available), 2013.

[29] Birgit Penzenstadler and Henning Femmer. *A generic model for sustainability with process- and product-specific instances.* March 2013.

[30] Birgit Penzenstadler, Ankita Raturi, Debra Richardson, Coral Calero, Henning Femmer, and Xavier Franch. Systematic Mapping Study on Software Engineering for Sustainability (SE4s). In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 14:1–14:14, New York, NY, USA, 2014. ACM.

[31] Birgit Penzenstadler, Ankita Raturi, Debra J. Richardson, and Bill Tomlinson. Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century. *IEEE Software*, 31(3):40–47, May 2014.

[32] Ankita Raturi, Birgit Penzenstadler, Bill Tomlinson, and Debra J. Richardson. *Developing a sustainability non-functional requirements framework.* June 2014.

[33] Thomas Schulze. *A Cost model for Expressing and Estimating Ecological Costs of Software-Driven Systems.* PhD thesis, Universitt Mannheim, Mannheim, 2016.

[34] R. C. Seacord, J. Elm, W. Goethert, G. A. Lewis, D. Plakosh, J. Robert, L. Wrage, and M. Lindvall. Measuring software sustainability. In *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, pages 450–459, September 2003.

[35] S. S. Shenoy and R. Eeratta. Green software development model: An approach towards sustainable software development. In *2011 Annual IEEE India Conference*, pages 1–6, December 2011.

[36] Kevin Tate. *Sustainable software development an agile perspective.* Safari Books Online. Addison Wesley, Upper Saddle River, NJ, 2005.

[37] The Climate Group. SMART 2020: Enabling the low carbon economy in the information age, 2008.

[38] United Nations. Transforming our world: the 2030 agenda for sustainable development. New York, 2015.

[39] Colin C. Venters, Caroline Jay, Lydia Lau, Michael K. Griffiths, Violeta Holmes, Rupert Ward, Jim Austin, Charlie Dibsdale, and Jie Xu. *Software Sustainability: The Modern Tower of Babel.* November 2014.