Enhancing Sustainable Software Engineering Methods through SECoMo

Natalie Buchner

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

Abstract. Sustainability is a globally important topic in today's world. Its most prominent aspect is ecological sustainability, 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, for examply by the growing energy consumption of ICT hardware. But Software can also have negative impacts on ecological sustainability, thus it is important to consider how to increase the software sustainability. 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. Existing research proposes a number of sustainability metrics, measurement tools or process and life cycle models, but despite this variety of approaches, it seems that sustainable approaches of Software Engineering (SE) are not yet well established in practice. The Software Eco-Costs Model (SECoMo) approach is a new estimation approach in this field which allows to estimate and represent the ecological costs of software already from an early stage on in a software project. It enables stakeholders to have an early understanding of the sustainability impact of a software and to make design decisions accordingly.
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 the integration of such approaches in practice.

1 Introduction

In 2015, the United Nations (UN) introduced their "Sustainable Development Goals" [37] in order to raise awareness for the concept of Sustainability: these goals include the fight against poverty, protecting the planet and granting prosperitiy for everybody [22].The goals emphasize how important the topic of Sustainability has become for the world, and that taking action affects everybody: "governments, the private sector, civil society"[22, p. 1] and every private person.

Identifying existing negative influences on Sustainability is the first step towards reducing them and revealing the great potential for positive impacts everybody can achieve. Accordingly, many industries in the private sector have taken on their responsibility and have started to reduce their negative ecological impacts on Sustainability. The Information and Communication Technology (ICT) sector is no exception: with its products and services it contributes to high power consumption and carbon dioxide emission [19] and thus has a negative effect on global warming [36] - 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 increasing the Sustainability of data centers. The focus of these efforts is mainly on reducing the negative impacts of hardware products, but software can have an impact on Sustainability aspects, too - not only by enabling people or processes using it to be more sustainable, but also as a product itself, for example during its development process. Efforts directed at improving software Sustainability during its development and usage phases are an important aspect of IT Sustainability.
In the growing research field of Sustainable SE there is a great number of scientific contributions covering ideas for example on Sustainability guidelines for software, models for sustainable development life cycles or specific energy measurement tools for certain types of applications. However there is a lack of an evaluation of these approaches in practice and more conrete models and metrics that can be applied in a great variety of software projects are missing. This paper aims at comparing a new approach in the field of Sustainable SEoftware Engineering, SECoMo [32], 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, and how it can contribute to increase the practical adoption of Sustainable SE overall.
The relevant background topics for this seminar thesis, Sustainability and its relation to ICT, software in particular, are introduced and defined in the first section. In the following, the related work from the field of Sustainable SE is presented. In addition, the SECoMo approach is introduced and described with a focus on its general purpose and motivation. In the following section, SECoMo is considered in the context of Sustainable SE and analyzed regarding its ability to enhance existing approaches, in order to identify its benefits for the field of sustainable SE and its overall impact on the practical adoption. Lastly, limitations of SECoMo in the context of sustainable SE are presented and a conclusion of the paper is given.

2 Background

2.1 The Concept of Sustainability

Although it is clear that Sustainability is a concept of increasing importance in the world and impacts many areas of our daily lives, the definition of the term Sustainability itself is not as clear and not always consistent. 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 [11]) and thus it can be hard to find a common understanding [14]. In order to successfully work towards a sustainable future, finding this common understanding is absolutely necessary, though [14].
Most attempts at defining the concept of Sustainability start with the definition of the concept of *Sustainable Development* in the so-called *Brundtland Report* in 1987: "Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs"[24, p. 4]. In addition, the requirements for sustainable development according to [24] are related to three different aspects: society, economy and environment. These three aspects are now widely accepted as "the three dimensions of sustainable development" [37, p. 1].
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*. 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. It is chosen as the definition of Sustainability that is further referred to in this paper

2.2 Sustainability and ICT

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. It is obvious that ICT plays an important role in the context of Sustainability, too: on one hand, as negative contributor, for example to global warming to its increasing carbon dioxide emissions; on the other hand it has a great potential for positive impacts, 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 [36].
The energy consumption of hardware is the main direct impact ICT has on Sustainability. While direct impacts are often quite obvious, 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. A more fine-grained differentiation of the effects of ICT, which is often used in the literature ([12]; [20]), distinguishes between three types: First-order

effects, second-order effects and third-order effects [5]. While first-order effects refer to direct impacts "of the physical existence of ICT" [12], 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 [20]. 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" [12] which themselves have influence 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"[13, p. 5]? According to Hilty et al. [13], there are a couple of relevant approaches in the field of ICT that contribute to Sustainability: Environmental Informatics, Sustainable Human-Computer Interaction (HCI) and Green IT (or Green ICT).
Environmental Informatics deals with applications that process environmental information in order to tackle ecological problems [13], thus it is related to second-order ICT effects.
Sustainable HCI covers sustainability *in* and *through* software design [13] and is related to first and second-order effects.
Green IT is a broader ICT subfield that has been continuously adapted by companies and organizations since the publication of a Gartner Report [19] in 2007. It covers two categories: Green *In* IT and Green *By* IT [13]. 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 [7], Green By IT aims at improving the environmental sustainability of aspects in different areas (other products or processes) by using ICT as the enabler [13]. Green In IT approaches cover the whole life cycle of ICT products, from its production over the usage phase until its disposal or recycling phase. It is important to consider impacts of ICT in all these life cycle phases in order to achieve holistic environmental sustainability [13]. Green IT is related to first- and second-order effects of ICT, but as a strategic concept adapted by many organizations, it has the potential to create long-term, third-order effects, too.
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 do not take the impacts of software artifacts fully into account.

2.3 Sustainability and Software

Software artifacts actually play an important role in the relationship of ICT and Sustainability: Although it may seem that software as the immaterial part of ICT products does not have an impact on Sustainability and "is automatically green" [1, p. 3], this naive assumption too rashly. For example, when software is developed, a lot of physical resources are needed during this process; when software is used, it can support sustainability, for example by processing environmental data to better understand ecolocial challenges; when software is

deactivated, dealing with saving or converting its data might have economic impacts on a company [15]; and it can even have long-term third-order effects like changing user behavior towards more sustainable practices (like video conferencing instead of business travel [3]). During its usage phase, the software itself does not consume energy - but if software is regarded as "the ultimate cause of hardware requirements and energy consumption" [18, p. 1], it becomes clear that it is after all indirectly influencing energy efficiency aspects.
Considering all these direct and indirect impacts, software definitely needs to be taken into account as an equally important part of ICT Sustainability considerations. Accordingly, for the ICT subfield of SE there is a need to include Sustainability topics. But so far, such considerations are not a part of traditional SE methods [26]. This is critized by a growing number of authors ([26]; [1]; [3]), as the awareness for the importance of Sustainability in SE grows. The emerging field of *Green* or (more holistically) *Sustainable Software Engineering* aims at filling this gap by providing approaches that deal with sustainable software and its engineering. Sustainable SE thus constitutes another important apporach how to use ICT in the service of Sustainability.

In the literature, there does not yet exist one common definition of Sustainable Software and Sustainable SE [38]. Nevertheless, for the purpose of this paper we chose the definition by Dick et al. [10] 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" [10, p. 3]

Accordingly, Sustainable SE provides the systematic methodology to develop 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. 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" [20, p. 3]

This definition provides a holistic description of the idea of sustainable SE, especially due to its focus on the impacts the software product has over its whole life cycle.
In this context, Penzenstadler [26] 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 [26]. This categorization will be used in the following in order to describe and categorize existing contributions in the field of sustainable SE.

3 Related work in Sustainable SE

3.1 State of Research

According to a Systematic Literature Review (SLR) conducted in 2012 [25], most of the research activity in the field of sustainable SE has started in the beginning of the 2010s and has significantly increased since then. This is underlined by the follow-up study from 2014 [29], which identified around 40 relevant publications that were added just in between the two studies. In summary, the main insights provided by the studies are: While there are many contributions over the last years that relate to software and Sustainability in some way, most research still goes into "domain-specific, constructive approaches" [25, p. 5], and general reference approaches that can be applied to the whole domain of sustainable SE are missing [25]. The study from 2014 identified already more contributions with a focus on general aspects of SE processes and Software Design [29], but also found that most covered application domains are still only slightly connected to SE [29]. Moreover, a lack of case studies and experience reports about applications of proposed approaches and methods in practice was identified [25], which shows that the research field is still mostly shaped by academia and the practical view is missing [29].
The following aspects stand out in current research because they are targeted by many authors and are thus the most relevant issues of the field currently:
There is a *need for a unified definition* of the concepts of sustainable software and the role of Sustainability in SE. This need is apparent and covered by many authors ([38]; [4]; [7]; [27]). The general opinion is that despite the variety of attempts, it is not possible to identify a commonly accepted definition yet, due to the recency of the research field, but also due to the complex nature of the concept of software sustainability [38]. This lack of a defined common ground is critical, because until a wiedly accepted definition is established, most efforts in the field will be likely to "remain insular and isolated" [38, p. 5] and thus hinder the practical adoption of Sustainable SE.
Integrating Sustainability as a software quality attribute by *making it a Non-Functional Requirement (NFR)* in official SE standards is emphasized by many authors as well ([30]; [3]; [1]). This step would account for the central role sustainability plays today [30] and help to integrate Sustainability "early in the lifecycle of software development and influence design decisions" [31, p. 2], but there are still appropriate analysis and assessment methods missing as well as policies and standards ([30]; [38]).
In general, as also found by the SLR, the field is *missing concrete reference models* and methods that can provide holistic frameworks to actually perform sustainable SE [25] and thus provide guidance to Sustainable SE researchers and practicioners. 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 aim at providing concrete models and metrics to be used in the context of sustainable SE. The most relevant ones will be presented in the next section.

3.2 Overview of existing contributions to Sustainable SE Practices

Sustainability metrics and measurement methods Many contributions to sustainable SE focuses on methods and tools to measure energy consumption of software. There are several tools or measurement frameworks for tracking or measuring energy consumption of software, like *GreenTracker* [3], *PowerIndicator* [21] or *PowerAPI* [23], that all have different focus aspects. PowerAPI for example enables the monitoring of energy consumption of an application during runtime, and especially allows to consider the impact different programming languages have on the power consumption [23]. Other approaches introduce mathematical metrics like the energy efficiency measures proposed by [8] and [16] - they consider the relative energy consumption of applications and in addition they do so on a low level, which makes it possible to "find resource intensive parts of programs and improve them" [16]. The required effort to test and use these metrics is quite high, and the metrics are very specific, which indicates that it is not likely that these metrics can be applied in practice soon [16].
Kern et al. [18] propose another interesting metric and the respective calculation method for the carbon footprint of a software project. This sustainable SE contribution from 2015 aims at finally providing a practical approach for measuring the environmental impact of software which can be integrated in a software development process [18]. The basic idea is to calculate the respective current footprint of the development process of a software project or of the software product itself [18]. The metric itself is expressed as kg CO2 per person month and is calculated based on rather high-level inputs like number of working days or number of employees [18].
Nearly all of the existing metrics and measurement methods focus exclusively on power and energy consumption and thus mainly target environmental sustainability. Nevertheless, there are efforts to consider software impacts in relation with other dimensions of Sustainability, like the Sustainability properties and related metrics provided by Albertao et al. [2]. Based on existing software quality measures, eleven software Sustainability properties were derived, which include aspects like Usability, Efficiency or the Project's Footprint [2]. In addition, this contribution proposes a sustainability measurement method which includes assessing the sustainability metrics after the release of a software in order to derive improvement goals [2].

Software life cycle, process and other software sustainability models A great variety of approaches exist that propose systematic models for sustainable SE, covering models for very specific purposes and application domains or rather general reference models. General models are for example the ICT impact assessment model by Hilty et al. [12] or the Sustainability reference model by Penzenstadler and Femmer [28], which provide general gudiance, but rather for side aspects of sustainable SE.
Very important types of systematic models for sustainable SE are models with a focus on the Software Development Life Cycle (SDLC) and general process models, as they give guidance on how to actually perform the development. Most of

these models concentrate on enhancing existing traditional models 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 [34], which include aspects like simply reducing resource usage like paper [34], 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]; [34]).
The systematic sustainability model that currently seems to be most important in research area, as it is highly referenced in many related contributions, is the *GREENSOFT* model [20]. This "reference model for green and sustainable software and its engineering" [20] is a comprehensive model that combines many aspects of sustainable SE and can be considered to be the best starting point for applying sustainable SE to practical projects at the moment. It consists of four parts: a *life cycle model* that is used to assess the impact of the software during its different phases; a set of *sustainability criteriy and metrics*; different procedure models (like for development or administration); and a collection of further recommendations for different stakeholders [20]. 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 (for example with sustainability reviews [20]), 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 [20].

3.3 Adaption of sustainable SE in practice

As the previous sections show, Sustainability in SE is a relevant topic in current research, with increasing acadmic interest and many contributions. Nevertheless, as various authors point out, there is a lack of practical evaluation of existing methods and apporaches and of experience reports ([25]; [29]).
This lack of practical evaluation is a sign for the very limited adoption of sustainable SE practices in non-academic areas. As Kern [17] points out, the field itself is not yet open for non-academic contexts - but this is absolutely relevant in order for sustainable SE to actually make a difference and to be useful. Reasons for this problem are on one hand the still missing common ground in the field (see section 3.1, [17]) but also the fact that practioners are not yet included in the discussions [17].
Kern proposes ideas how to open up the research field in order to increase the adoption of sustainable SE practices in the industry and the public sector, and names two possible ways forward: either starting with including software delevopers in the discussion and use them to evaluate existing ideas, or starting with

raising awareness for the issue with private users in order to come up with new ways how to characterize sustainable software [17]. The first idea seems to be a logical step forward given the issues of the field, but in the author's opinion, there is still one significant issue impeding this: As long as there is still a lack of models and metrics that can be easily adopted for a variety of different projects, the adoption and evaluation of existing sustainable SE approaches will continue to be limited to small, specific cases and will not reach a broader range of practitioners. The existence of reference models like GREENSOFT is already a step in the right direction by providing a framework for actual, sustainble software projects, but it is still missing concrete details for development processes, for example. Thus, what is needed to promote the field of sustainable SE are such concrete approaches that can still be applied for a variety of projects.

4 The SECoMo approach

The **Software Eco-Cost Model** (SECoMo) was introduced by Schulze [32] in 2016. 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 [32]. Thus, SECoMo represents a concrete estimation approach for the impact a software system has regarding ecological sustainability during its usage phase.

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 project in order to keep the costs for changes low [32]. 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 [32]. 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 [32].

In addition to the mathematical models, the SECoMo approach defines a set of eco-cost drivers in order to identify causes for certain ecological impacts and to better describe under which circumstances they occur [32]. 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 [32]. 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 [32].

Against this background, the SECoMo approach is intended to be used in the early stages of SE 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 [32]. 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 [32].
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 [32].
From the initial design phase on, SECoMo can be applied in a SE project, if information are available on 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, which are the basis for the further steps of SECoMos estimation technique. SECoMo does not require a specific modeling approach to create these views [32].
If this basis is given, the following steps of the SECoMo approach can be integrated in any step of the SE process - in the design phase already if early eco-cost estimations are required, otherwise 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 [32].
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) [**?**]
3. Calculation of the estimates for the different eco-cost metrics based on the available information and the specified parameter values, which specify aspects like user type or case [32].

5 SECoMo in the context of Sustainable SE

Classification of SECoMo Recalling the definition of sustainable SE (see section 2.3), it becomes evident that the SECoMo approach aligns with many of its 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 SE, 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 [26]: 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 [27], the system usage aspect is the most relevant one as it has the biggest impacts, especially the more the software is used.

SECoMo and Sustainability Metrics Like the majority of the introduced metrics (see section 3.2), SECoMo focuses mainly 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 to a high-level scope (e. g. from execution costs for a single operation to those for a concrete scenario [32]), 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 ([21]; [8]).
What the SECoMo metrics do not provide are information about energy efficiency (like [8] and [16]), 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. 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 on which parts of the software might be very resource intense (for example the execution of a certain operation, or updating a certain data type [32]).
The goal and motivation of SECoMo actually resembles those of the presented carbon footprint calculation method by Kern et al. [18]: 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 ([18]; [32]). Moreover, the authors mention in their motivation the need for creating awareness and transparency for the aspect of software energy consumption [18] (which SECoMo aims at, too, mainly with regards to the stakeholders [32]). Finally, 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 [18] - which is one of the main benefits of SECoMo, too [32].
The differences between the two approaches however are also clear: While the Carbon Footprint metric by Kern et al. has a fixed unit and is thus restricted to a certain context, the SECoMo metrics can vary in the units they represent (for example CO2 emissions, but also dollars [32]) 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 [32], and they are not restricted to the high-level inputs the Carbon Footprint metric requires [18], but can be way more fine-grained [32]. 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 [18], SECoMo only applies to the eco-costs caused by the software product itself [32].
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 section 4, SECoMo can be integrated into traditional SE processes by adding the estimation or calculation steps in the design phase or later phases. Thus, it should also be possible to integrate SECoMo into sustainable SE processes and SDLC models that are enhanced with sustainability activities (like the ones proposed by [1] and [34]), 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 [20], 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 [20] - 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 [20]: 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 [20] - 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 [32].
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.

Sustainable SE and SECoMo in practice

5.1 Limitations

Even though the SECoMo approach offers many new aspects to the field of sustainable SE and can enhance certain approaches in a positive way, there are a

few limitations to the benefit of SECoMo for sustainable SE:
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 SE 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 [32]. 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 SE 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 SE approaches to be adapted in practical projects - this is after all 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 [32], 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

In the context of Sustainability, ICT plays a major role as contributor to negative impacts, but also as enabler to improve Sustainability - in the ICT sector itself, but also in other sectors. Even though it might not be obvious at the first glance, software products are relevant in that context, too: They can have direct and indirect impacts on the different dimensions of Sustainability, for example by influencing a hardware product's resource consumption or by promoting sustainable life styles.
The research field of sustainable SE deals with concepts and approaches on how to ensure that software products can improve their positive impact on Sustainability during their whole life cycle. Over the past few years, the amount of contributions to the field has significantly increased, and a variety of concepts exist that tackle different aspects of software sustainability. But a common understanding and concrete guidance is still missing in the field, and despite the variety of approaches, not much is evaluated in practice, yet.

The SECoMo approach introduced in this paper is yet another addition to the contributions of this field, in particular in the area of approaches dealing with software usage sustainability. It has the ability to enhance existing approaches in the field in various ways and might thus be suitable to help leading Sustainable Software Engineering to more adaption in real life projects. In any way, SECoMo can play an important role in the field in different areas:
With regards to existing metrics, SECoMo provides a new set of eco-cost metrics

that can be applied to various types of software development projects as they are more flexible and can cover a broad range of contexts. In comparison to existing sustainability models in the field, an important role that SECoMo can play is to enhance the already well-rounded GREENSOFT model with a concrete estimation approach for environmental sustainability aspects in order to make its development procedure model more likely to be adapted in practice. Over all, SECoMo introduces a new type of approach to sustainable SE with its ability to estimate eco-costs even before development phase has started. SECoMo can even contribute to the important area of Requirements Engineering by providing a much needed vocabulary to define sustainability requirements and also the respective method how to assess them.

The future work on SECoMo will show whether it is indeed a suitable approach that can enable the adoption of sustainable SE in practice, and in any way there is still some work to do in the research field itself, starting from finding a common understanding of the underlying concepts over to evaluating the existing variety of approaches in real life projects. But the trend of Green IT and Sustainability adaption in companies for example shows, that the work in this direction is important and very much needed.

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] 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.

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

[4] 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.

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

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

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] 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.

[17] Eva Kern. How to cast the approaches on green software engineering upon the world: Collection of ideas in order to exit the ivory tower. In *4th International Conference on ICT for Sustainability (ICT4S)*, Amsterdam, Niederlande, August 2016.

[18] 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.

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

[20] 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.

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

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

[23] 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.

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

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

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

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

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

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

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

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

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

[38] 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.