Information and Software Technology 95 (2018) 108–129

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Interactions between environmental sustainability goals and software product quality: A mapping study

Gabriel Alberto García-Mireles[a,*], Mª Ángeles Moraga[b], Félix García[b], Coral Calero[b], Mario Piattini[b]

[a] Departamento de Matemáticas, Universidad de Sonora, Blvd. Encinas y Rosales s/n col., Centro, 83000 Hermosillo, Sonora, México
[b] Instituto de Tecnologías y Sistemas de Información, Universidad de Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

ARTICLE INFO

Keywords:
Environmental sustainability
Greenability
Interaction
Software product quality
ISO/IEC 25010

ABSTRACT

Context: Sustainability is considered as either a quality requirement or a quality characteristic that should be included in software when environmental protection concerns are being taken into account. However, addressing sustainability in software projects might have an impact on the quality of the software product delivered. Conflicting goals between sustainability and particular software product characteristics should be studied when developing application software, since achieving users' requirements can be a hindrance in the quest to meet sustainability goals.

Objective: This paper aims to provide an overview of the approaches found in the literature for dealing with interactions between software product quality and sustainability in the context of application software.

Method: A systematic mapping study is conducted to identify practices for managing interactions between software quality characteristics and sustainability. The selected papers are classified according to the quality characteristic considered and their influence on sustainability.

Results: Most of the 66 selected papers focused on validating current technologies concerning their support for sustainability (46%%). The interaction between performance efficiency and energy efficiency is what is reported most and there is a fairly positive interaction. In addition, reliability and usability point to a positive interaction with energy efficiency, while security shows a conflicting interaction with energy efficiency. Functional suitability and maintainability can present both positive and negative interaction, with different goals derived from environmental sustainability.

Conclusions: Interactions between software quality and sustainability have been addressed within an explorative approach. There is a need for additional research work to characterize the impact of interaction on both software quality and sustainability. Furthermore, proposals should be validated in industrial settings.

1. Introduction

In recent years, software engineering has focused on developing sustainable software, which is defined as "software, whose impacts on economy, society, human beings, and environment that result from development, deployment and usage of the software are minimal and/or which have a positive effect on sustainable development" [1]. The impacts can be studied in three distinct scope levels [2]: direct (e.g. energy consumption), indirect (e.g. reducing energy consumption when supporting a business process), or rebound effect (e.g. optimizing energy efficiency of a product could have the effect of increasing its demand and therefore the overall energy consumption due to such product).

From the three dimensions of sustainable development identified in the Brundtland report [3] the environmental dimension, or "green" dimension, constitutes the context of this paper. In this dimension, sustainable software promotes energy efficiency, minimizes the environmental impact of the processes it supports, and has a positive impact on social and/or economic sustainability [4].

This paper focuses on the area of application software development, while also looking at the way environmental sustainability goals have an impact on software product quality. As a matter of fact, practices used to develop application software might well influence energy consumption. For instance, Capra et al. [5] reported that the application layer can increase the energy consumption of a server by up to 72% %; management information systems applications which satisfy the

* Corresponding author.
E-mail addresses: mireles@mat.uson.mx (G.A. García-Mireles), MariaAngeles.Moraga@uclm.es (M.Á. Moraga), Felix.Garcia@uclm.es (F. García), Coral.Calero@uclm.es (C. Calero), Mario.Piattini@uclm.es (M. Piattini).

http://dx.doi.org/10.1016/j.infsof.2017.10.002
Received 8 April 2017; Received in revised form 3 October 2017; Accepted 6 October 2017
Available online 09 October 2017
0950-5849/

same functional requirements can mean up to 145%% increment in energy consumption when they are executed in the same computer system. Furthermore, Ardito et al. [6] found that software applications have a significant impact on the power consumption of desktop computers. Given an increased trend towards energy consumption due to personal computers and mobile devices [7], it is relevant to study the effect of software engineering practices on application software sustainability.

The main goal of this work is to investigate the approaches proposed in literature for dealing with interactions between software quality and sustainability in the context of application software. To achieve this goal, a Systematic Mapping Study (SMS) [8] was conducted, whose aim was to obtain a set of relevant papers that describes the practices, methods and techniques used to identify and resolve interactions between software quality and sustainability in the context of application software. To fulfill this purpose, two main issues are researched.

The first matter dealt with is sustainability, which is considered to be both a nonfunctional property of software and a software quality characteristic. In the former case, it includes resource consumption, greenhouse gas emissions, social sustainability and recycling [9], which can be treated as nonfunctional requirements in software projects. From the Green IT perspective, Ardito and Morisio [7] provide a summary of recommendations found in literature about the way to improve energy efficiency when the software application layer is considered. In this latter case, sustainability can be considered a central quality attribute, as safety and security, among others, are [10]. Using the Software Quality Assessment based on the Lifecycle Expectation model, Ardito et al. [11] proposed that energy efficiency was a subcharacteristic of the efficiency characteristic, and should be considered during software construction. Moreover, Calero et al. [12] extended the ISO/IEC 25010 [13] quality model by adding greenability into both the product quality model and the quality in use model, thus providing practitioners with a way to include environmental sustainability goals in software projects. A new challenge, therefore, is how to handle the interactions of sustainability with the rest of the quality characteristics.

The second issue is about interactions, which in the requirements area are described as situations in which the satisfaction of one requirement may affect the satisfaction of another [14]. With respect to software quality and sustainability, several researchers have stated the need to understand this interaction. Lago et al. [15] pointed out “It is already extremely difficult to identify, address and balance quality concerns in non-energy-aware software. In addition, when green concerns enter the picture, they are even more pervasive and potentially impact all other system qualities”. Furthermore, delivering high-quality software may increase the level of energy consumption, making it difficult to meet software green goals [16]. Addressing both sustainability and software quality may require an additional conceptual framework in order to provide support to trade-offs analysis and decision making [15].

As a result, from a practitioner's point of view, the SMS aims to summarize the influence of sustainability concerns in specific quality characteristics. Hence, they may use this information to make decisions about the way environmental sustainability goals can be included in a software project. From a researcher's perspective, the aim of this work is to provide a classification of the topics addressed and the empirical methods applied, as well as the way software quality characteristics have been studied. Thus, the SMS can provide an initial identification of papers so that further research can be started on topics barely addressed as yet.

The remainder of this paper is structured as follows. Section 2 presents the fundamentals about sustainability in software engineering, as well as the outcomes of related literature reviews. Section 3 describes the method used to conduct the mapping study. Section 4 sets out the results of the mapping, and Section 5 presents the discussion (summary and implications). Section 6 gives a description of the study's limitations. Finally, Section 7 provides the conclusions and proposes future work.

2. Background

This section provides an overview of the related work with regard to environmental sustainability and software quality. The first part describes some notions of the concept sustainability in software engineering. The second part presents a summary of related literature reviews about software engineering and software quality.

2.1. Sustainability in software engineering

Sustainability is a term with multiple meanings, all of them referring generally to the “capacity of something to last for a long time” [17] or the “capacity to endure” [18]. However, there is not a consensus in the definition of the term. Hilty and Aebischer [19] noted that “many controversies related to sustainability stem from the fact that people think of different systems and functions to be sustained, as well as different time horizons, …” Thus, sustainability can be discussed with reference to a concrete system or a global system [4]. A concrete system can be a specific software system [4,20] or any other specific natural or man-made system. In contrast, sustainability of the global system “implies the capacity for endurance given the functioning of all […] systems in concert” [4].

Similar in scope to global sustainability, the concept of absolute sustainability [19] refers to the definition of sustainable development included in the Bruntland report: “sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs” [3]. Addressing sustainability in the context of software engineering is based on the Bruntland definition. Software engineering research is focusing on concerns about the impact of software systems on global sustainability [4]. Incorporating the concept of sustainability into the software engineering discipline implies a consideration of the impact of the software in the surrounding environment throughout its life cycle stages, e.g. development, operation and maintenance [1,17]. The impact can be characterized in three orders of impact [2,21]: the most common way to address sustainability in software is through energy efficiency [20,22], but other effects can be considered [10], such as energy usage, e-waste production, emissions caused by required infrastructure (first order effect); in addition, there are changes in user behavior caused by software (second order effects); and changes in social behaviors induced by software systems that erode the benefits of optimizing energy efficiency (third order effect).

In addition to energy consumption, this research also considers any other type of resources studied, including the analysis of level of impact; this is related to environmental sustainability, whose central goal is “to improve human welfare by protecting natural resources [such as] water, land, air, and ecosystem services” [23].

Sustainability during software development requires a “tangible decomposition of the concept of sustainability” [23] expressed as dimensions. Several research proposals address the economic dimension, social dimension, environment dimension, and technical dimension [20,23,24], as the very minimum that should be taken into account, due to the fact that “sustainability is achievable only when accounting for all dimensions” [20]. From a practical point of view, sustainability dimensions can be studied separately before exploring an integrated view [24].

This work is focused on the environmental dimension of sustainability, which is also called the green dimension [17]. In this context, green software has the same goals as sustainable software within the environmental dimension of sustainability [25–27]. Taking into account its purpose, green software is broken down into green in software and green by software [17]. Green in software is related to how to make software in a more sustainable way in order to develop a more sustainable software product, i.e., to develop a more environment-friendly software. On the other hand, green by software refers to software developed for domains focusing on the preservation of the environment,

as well as to software that helps to manage energy-intensive applications. Similarly, Penzendstadler et al. [4] interpret sustainable software in two ways: as software code that is sustainable, agnostic of purpose; and also as a software purpose directed at achieving sustainability goals. This research therefore focuses on green in software, since its context is the development stage of application software.

2.2. Related literature reviews

Few systematic literature reviews have been conducted with regard to sustainability and the software engineering discipline. Since the goals of the reviews identified do not address interactions between sustainability and software quality characteristics, this section presents their main findings about papers that discuss sustainability in, and topics related to, software quality.

The first review about sustainability in software engineering is presented by Penzenstadler et al. [28] who reported that there is a tendency to investigate sustainability in specific domains (e.g. transport, avionics, among others) and also to develop approaches such as conceptual models, software-based frameworks to be used in other domains (e.g. civil engineering), and measures (e.g. sustainability in e-business). In addition, they concluded that the field is highly complex, highly-domain specific, and that there is a lack of proposals for developing sustainable software.

In a follow up review, Penzenstadler et al. [4] performed a more in-depth overview on the trends of research in the topic area “Software Engineering for Sustainability” (SE4S). They found that the most common software engineering areas addressed in the selected articles are: software design, software engineering models and methods, software quality, and software requirements. These authors concluded that the SE4S research topic has received wide-spread attention over the past few years, though there is little reported evidence about the implementation of methods in industrial settings. As a threat to validity, they reported that they missed expected results because they did not consider terms like ‘software quality’ in the search string [4].

As far as including sustainability in software quality models is concerned, the state-of-the-art in software sustainability measures is tackled in [22]. They found measures that focused on sustainability also addressed other quality characteristics such as performance efficiency, maintainability, portability, usability and reliability [22]. Concerning green software metrics, Bozzelli et al. [29] found that application software context is the most common category for measures. They concluded that “research is continuously interested in developing and proposing measurement techniques to measure or estimate application energy consumption”.

In a nutshell, the systematic literature reviews previously alluded to [4,28] cover the development of software in disciplines such as environmental management, transport, agriculture and business economics, among others. Hence, they correspond to the green by software category [17]. The other two reviews [22,29] focus on the definition of measures to determine the extent to which any software is greener than others. Although both papers point out that green measures are related to other product quality characteristics, mainly related to performance efficiency, they do not analyze the relationship between sustainability and product quality.

3. Planning of the SMS to provide an overview of software quality and sustainability

The research goal of this SMS is to identify interactions reported between software product quality characteristics and the green aspects, along with practices reported for managing them. The classification of selected papers must answer the following two general questions:

- GQ1. What is the profile of papers addressing interactions between environmental sustainability goals and product quality characteristics when application software is developed?
- GQ2. What specific interactions between environmental sustainability goals and product quality characteristics are described in selected papers?

To answer GQ1, the profile of selected papers is determined by the following sub-questions:

- GQ1-RQ1. What are the publication trends considering interactions between green aspects and software product quality?
  This question looks for trends about frequency of publication by year, as well as the publications venues of selected papers.
- GQ1-RQ2. What software engineering topics are addressed?

To classify topics addressed by primary papers, the Guide to the Software Engineering Body of Knowledge (SWEBOK) [30] is used. Software product quality can be addressed in all stages of the software development life cycle, and the interactions between quality characteristics have been reported to be present in different life cycle processes [31,32].

- GQ1-RQ3. What research approaches are reported in primary papers?
  The research approaches are classified according to the Wieringa classification [33] and the rules provided by Petersen et al. [34].
- GQ1-RQ4. What approaches are used to describe software quality in the context of its interaction with sustainability aspects?

Software quality can be studied from diverse perspectives and using manifold levels of abstraction [13,35]. This work is focused on the product quality, although the process view might be applied when software is developed [35,36]. The *process view* of quality can be operationalized by counting defects during software life cycle activities [37,38]. These are treated as surrogates of software quality. In contrast, product quality models support the identification of specific properties in software that can be defined, measured and used to evaluate product quality from the user perspective in the context of tasks performed. ISO/IEC 25010 provides a *product quality model* and a *quality in use model*. The mission of the former is to define, measure, and evaluate software quality during the software development stage, while the latter is focused on the operational stage of a software product. This research focuses on the characteristics of the product quality model.

Considering the specific interactions between environmental sustainability goals and product quality, GQ2, should answer two sub-questions:

- GQ2-RQ1. What practices, methods, techniques, models, or measures are used to manage interactions between software quality characteristics and environmental sustainability goals?
  It addresses the methods, process, models and measures applied to identify and resolve conflicting interactions between software quality and sustainability.
- GQ2-RQ2. What specific interactions between software quality characteristics and environmental sustainability goals are presented in selected papers?

The identification of characteristics involved in an interaction needs to take into account the context in which an interaction is described or found, as well as the particular subcharacteristics or attributes mentioned.

3.1. Search strategy

The development of a search strategy was based on Kitchenham and Charters guidelines [8]. The identification of keywords was based on the main terms: software application, interaction, software product quality and sustainability (Table 1). Concerning interaction, the focus is on technologies that manage interactions or trade-offs between

Table 1
Keywords used in this SMS.

Keywords	Synonyms
Sustainability	Sustainab* OR green OR "energy efficient" OR ecolog*
Application software	Software
Software product quality	Quality OR goal OR nfr OR nonfunctional OR non-functional OR "quality requirement" OR "quality requirements" OR *9126 OR *25010 OR standard OR usability OR security OR compatibility OR functionality OR efficiency OR portability OR maintainability OR reliability

software product quality characteristics and sustainability. This includes approaches to identify interactions as well as to resolve interactions. Interactions can be referred to by different terms such as inconsistency, conflict, negative interaction, dependency, interdependency, balance, tradeoffs, among others [14,32]. So, if a paper explicitly mentions a *sustainability* term (Table 1) and any *software product quality* term, it can be a candidate paper. The set of keywords related to sustainability was formulated by taking into consideration the review of examined literature in Section 2. The list of synonyms for *software product quality* was extracted from a previous SMS which addressed the topic of software product quality [35].

With regard to the time span, the automatic search was conducted for publications from 2000 until December 28, 2016. The lower limit year was determined by checking publication trends reported in related literature reviews. For instance, Bozzelli et al. [29] reported on papers about measures in the green software engineering field since 2001. Looking for sustainability in both software engineering and requirements for software systems, [28] reported that research papers have been published since 2006.

The search string was tested on the Scopus database (http://www.scopus.com/). After assessing the outcome of applying selection criteria on papers retrieved from this database, automated searches were performed on the databases of the ACM digital library (http://dl.acm.org/dl.cfm), IEEE Xplore digital library (http://ieeexplore.ieee.org/Xplore/home.jsp) and Web of Science (http://apps.webofknowledge.com/). Appendix C includes some sections of the SMS review protocol.

3.2. Study selection

The process for identifying primary articles was divided into two parts. The first part consisted of reading the title and/or abstract and of applying the inclusion and exclusion criteria to select candidate papers. The second part consisted of reading the full text in order to identify primary papers. In contrast with the definition of primary paper [8], this SMS considers not only empirical papers, but also other categories of research papers [33]. The first author made the selection of the primary papers by applying the exclusion and inclusion criteria. The second author replicated the selection process, and the results were checked by all authors; discrepancies were discussed and the final list was obtained.

Inclusion criteria for selecting primary papers were as follows.

1. The article is written in English.
2. The paper is a peer-reviewed document, presented in a journal or conference, or in a book of scientific content.
3. The paper tackles the field of *Green in Software*.
4. The paper discusses aspects of development, use, evaluation, or disposal of application software. In general terms, the article should include or discuss a method belonging to any SWEBOK knowledge area [30] or activities and practices used to develop/maintain application software that can be addressed in the ISO/IEC 12207 [39].
5. The approach discussed in the article addresses aspects both of software product quality and of sustainability. In the former, it is expected that the paper should include at least one term related to software product quality, such as for instance, any of the quality characteristics or subcharacteristics of the product quality model described in the ISO/IEC 25010. With regard to sustainability, it is expected that the article address how to improve energy efficiency in software applications, as well as other approaches related to improving sustainable development.
6. The article must mention and/or discuss the nature of the relationship between software quality and sustainability. This includes among other considerations, the identification of interactions, approaches for balancing quality and sustainability goals, conflicting goals, tradeoffs between stakeholders' goals about quality and sustainability.
7. The paper discusses a practice, method, technique, model, process, or tool that is developed or applied in the field of software engineering. This work focuses on the green in software category [17]. Methods and practices can thus address the way environmental sustainability impact can be measured, monitored, evaluated, and optimized [21]. Since our focus is on software engineering practices, we need to take into account requirements engineering activities, whatever the domain in which they are applied may be.
8. The full article is accessible through the selected databases.

Exclusion criteria applied to retrieved articles were the following:

1. The full paper is not written in English.
2. The paper presents summaries, forewords of conference proceedings, editorials or systematic literature reviews.
3. The paper is not accessible in full-text.
4. The paper presents a duplicate study.
5. The paper tackles the field of *Green by Software*.
6. The paper addresses software that is not considered application software such as, embedded software, operating system software, middleware, services and web services, servers and data servers, software for optimizing communications networks, among others.
7. The paper discusses an approach to identify, model, analyze or evaluate interactions and tradeoffs that is not applied in the software engineering domain, or does it focus on describing methods of the field of search-based software engineering.
8. The paper includes either software quality or sustainability, but without providing support to identify interactions between software quality and sustainability.

In total, 3973 records were retrieved from four databases. As a result of applying selection criteria to title and/or abstract, 298 papers were selected as candidates. Selection criteria were applied again after the full-reading of candidate papers, and 66 primary papers were gathered. In Appendix A the main data used to categorize them are shown.

In order to provide a quality assessment of the studies chosen, a nine-question questionnaire was designed, derived from [40]. Each question is rated with the values: not included (0), partially included (0.5) and included (1). Authors evaluated each empirical study (34 of out 66 papers) in order to reach a final value for each one obtained by consensus. The evaluation of empirical papers showed a median of 8 and a mode of 9. Appendix B. presents quality assessment results.

3.3. Study classification

For the classification of the papers we considered: the year and source of publication, software engineering topics addressed according to the SWEBOK [30] and the research approach according to the Wieringa classification [33]. Statements about interactions between sustainability aspects and software product quality were extracted from primary papers, and the verbatim text (whenever possible) was used to

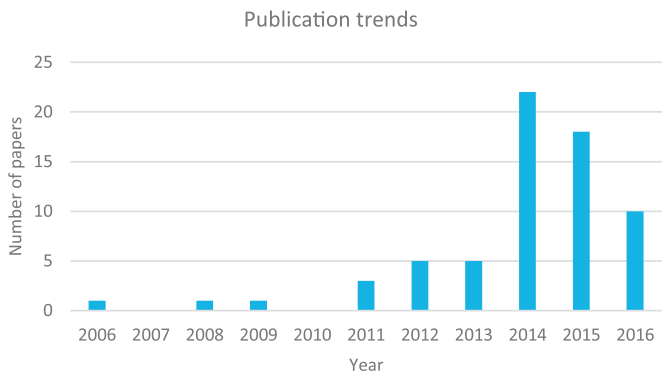

Fig. 1. Publication trends by year.

fill in a data extraction form. These propositions, named *text segments* in this paper, are the key evidence that all primary papers mention, at least, an interaction. Details about classifying selected papers by considering text segments are presented in Appendix D.

4. Results of the SMS: interactions between software quality and environmental sustainability

This section presents the results of the SMS, following the same order as the general research questions and sub-questions (Section 3).

4.1. Profile of selected papers

4.1.1. Trends in publication (GQ1-RQ1)

The year 2016 provides a partial set of primary papers, since the automatic search was carried out until Dec 28, 2016. As can be observed in Fig. 1, there is little work about interactions before 2011. Indeed, the first two papers focus on some ways of improving energy efficiency and its relation with usability [41] or resource utilization [42]. The third paper addresses the identification of requirements, considering sustainability goals in a project for the planning of a conference meeting [43]. The majority of papers, it is clear, were published after 2010 and especially in the last three years (50 papers). This trend shows that this topic is currently highly relevant for researchers from the software engineering field.

With regard to publication venues, this SMS focused on identifying peer-reviewed papers published in journals, conferences, workshops and research book chapters [34]. More than half of the papers (37 out of 66) were published in conferences, while 12 were published in workshops. The papers were published in both general software engineering conferences (e.g. ICSE, ESEM, SAC, among others) and specialized meetings for sustainable software development (e.g. GREENS, RE4SuSy). 21%% of papers (14 out of 66) were published in journals, two of them in a specific journal for sustainable development: Sustainable Computing: Informatics and Systems. Furthermore, three papers were published in the book Green in Software Engineering [44].

In relation to specific venues, there is a trend in publishing in a specific sustainable development workshop, such as GREENS and RE4SuSy, since they cover around 14%% of published papers. Other general conferences, such as ICSE, SAC, ESEM, to name just some, are also making an effort to meet the need to address sustainable development in the context of software engineering. Concerning journals, the special issue about sustainable software published by IEEE Software, and the specific journal *Sustainable Computing: Informatics and Systems* can be highlighted.

4.1.2. Software engineering topics (GQ1-RQ2)

The classification of the primary papers was based on the SWEBOK, in which software quality is a theme addressed in several knowledge areas [30]. Fig. 2 shows that only six areas (out of the 15 knowledge areas of the SWEBOK) have been addressed. The two most studied topics are software quality (22 papers) and software requirements (16 papers). Other software engineering topics tackled are software design, software construction, software maintenance and software engineering process.

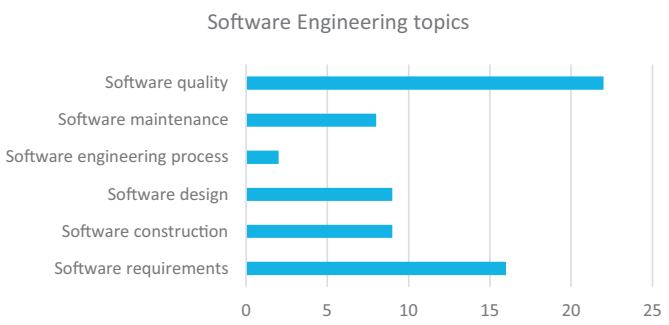

Fig. 2. Software engineering topics addressed.

With regard to software quality knowledge area, papers were classified by some subtopics derived from their content. The two main topics addressed are developing energy profilers to support measurement (68%%, 15 papers) and developing quality models (23%%, 5 papers). Other topics are prioritization of environmental aspects in the context of product quality characteristics and defect management.

4.1.3. Research type (GQ1-RQ3)

Fig. 3 shows the classification of the primary papers according to their research type, by following the guidelines in [33] and [34].

As can be observed in Fig. 3, the majority of primary papers belong to the validation research category (30 papers). This can be explained by the fact that researchers are exploring the suitability of using traditional software engineering methods and techniques in the context of developing sustainable software. The most frequent empirical methods applied in validation research category were: experiments (23 papers) and case studies (6 papers).

In the evaluation research category four papers were included. Koçak et al. [45] interviewed software practitioners to establish a priority between a subset of ISO/IEC 25010 quality characteristics and environmental sustainability. Moura et al. [46] explored the way energy efficiency is currently addressed by software developers using commits in the GitHub repository. Manotas et al. [47] carried out a survey to understand the way software practitioners address energy issues during the software development life cycle. Relying on mining data, Chowdhury and Hindle [48] analyzed the extent to which software projects are energy-aware. In the case of reporting experiences, the six papers focus on software requirements. The main topic studied was the identification of sustainability goals and the way they interact between them, as well as with other system goals.

In the category of philosophical papers, ten papers address both the characterizing of sustainability as a quality characteristic which can be included in ISO/IEC 25010, and the related measures needed to

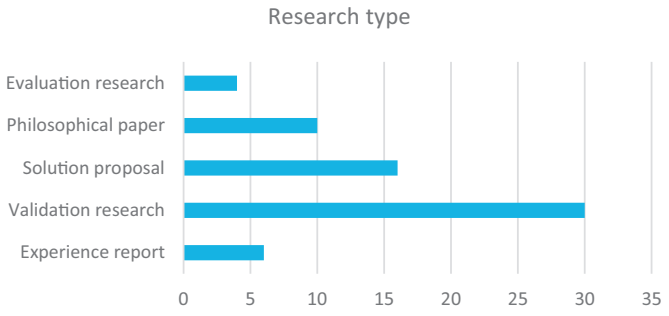

Fig. 3. Research type addressed by primary papers.

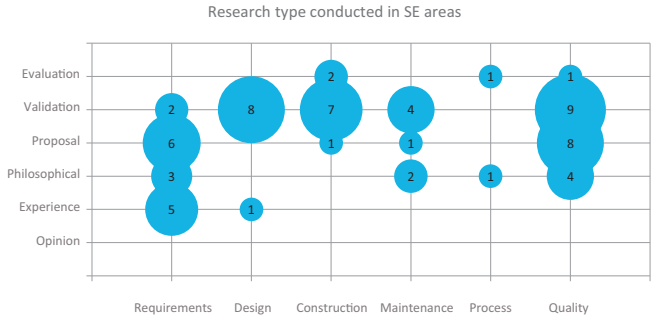

Fig. 4. Bubble chart relating software engineering knowledge areas to the type of research conducted.

evaluate it [12,49,50,105]. On the other hand, the main topic of the 16 solution-proposal papers is to measure energy consumption by introducing new tools and frameworks to monitor power consumption of application code [42,51–57]. Other proposals discuss methods to prioritize sustainability aspects and the way of adapting requirements modeling methods to include sustainability goals.

The bubble chart (Fig. 4) shows the software engineering areas addressed by the type of research conducted. As may be observed, validation papers have been published in the areas of design (8), construction (7) and software quality (9). In the first of these areas (design), we found several authors exploring design patterns from the perspective of optimizing energy efficiency. In the construction knowledge area, several authors reported that practices to improve performance can also improve energy efficiency, although there is a moderate correlation between them. In the last of the areas (software quality), papers focus on measurement approaches. Furthermore, the two most common software engineering areas are software requirements and software quality, which have also been addressed by conducting different types of research.

Meanwhile, as shown in Fig. 5, several research methods have been applied in the quest to understand the interactions between sustainability aspects and software quality. The majority of primary papers conducted experiments (24). In addition, 15 case studies were found, which include both exploratory case study and industrial case study. Furthermore, two surveys were also reported. However, nine papers did not present any empirical support.

4.1.4. What approaches are used to describe software quality in the context of interactions with sustainability aspects? (GQ1-RQ4)

The classification of papers was carried out by considering the main approach to address software product quality. 48 primary papers (73%%) mention, at least, a quality characteristic (or subcharacteristic) that is included in ISO/IEC 25010. The quality in use category (4%%) includes three papers whose main goal is to enhance energy efficiency while maintaining the user experience. Since authors do not address specific dimensions of the user experience, it was considered appropriate to use the quality in use model, since it can support the evaluation of interactions between users and systems [13]. The process view quality category (15 papers, 23%%) addresses the way in which defects, errors or bugs can have an impact on energy efficiency. In addition, it includes approaches for modeling interactions between software quality characteristics and sustainability goals.

Around 24%% of the papers are aware of a product quality model, such as [13] or ISO/IEC 9126, for example. Quality models serve two main purposes: to identify the quality characteristics which would support sustainability goals, or to adapt the quality model by adding a set of quality characteristics that support sustainability.

From each primary paper, text segments that mention the type of relationship between a software quality characteristic and a sustainability aspect were extracted. 106 distinct text segments were identified, since several papers considered interactions in multiple parts within the paper. From 44 papers (67%%), one text segment was extracted, while remaining papers contributed from two to five segments. Text segments that mentioned more than one quality characteristic were replicated to count the frequency of each quality characteristic. Fig. 6 sets out the quality characteristics addressed considering the ISO/IEC 25010 product quality model. Performance efficiency is the one studied most (43 records). Functional suitability, usability and maintainability are also studied in the context of achieving sustainability goals.

Energy efficiency is the main focus among selected papers, since 42 out of the 66 papers include terms such as “energy efficiency”, “energy consumption” or “power consumption”. The remaining papers present terms such as “sustainability”, “sustainability requirements”, “sustainability goals”, “green software” or “sustainable software”. Out of the whole set of primary papers, 33%% (22 papers) of them use some type of hardware to measure power consumption, while 27%% (18 papers) of papers use only software or an API to estimate the energy consumption of a software application.

4.2. Interactions between environmental sustainability goals and product quality characteristics

4.2.1. Methodological support for managing interactions between software quality and sustainability goals (GQ2-RQ1)

In order to characterize the interactions between sustainability aspects and software product quality characteristics, literature about interactions between requirements was reviewed [14,58,59,60] since the ISO/IEC 25010 product quality model is closely related to the specification of quality requirements [13]. *Interaction* is a type of dependency between two or more requirements that should be satisfied simultaneously in a software product [14]. In dependency models, *satisfies* is a specific kind of constraint relationship that is labeled as a conflicting relationship [60] and it defines the following situations: Two requirements cannot be met simultaneously in a software product, or increasing the satisfaction of one requirement has a negative influence in

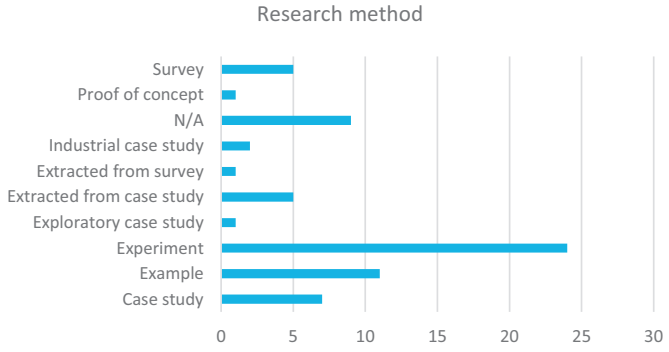

Fig. 5. Research methods used in primary papers.

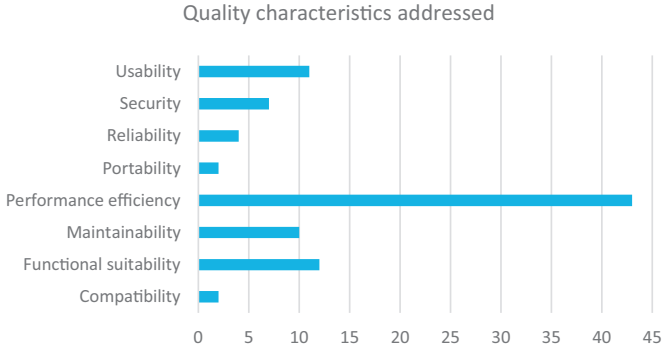

Fig. 6. Product quality characteristics addressed in the set of primary papers.

satisfying another requirement [58]. Although interactions can be described at requirements level, they can arise in all stages of software life cycle [31,32,61]. The way interactions are uncovered thus depends on the stage of the software life cycle and on other project settings, such as the goals and expectations of different stakeholders.

The way interactions are described in primary papers of this SMS were classified as general interaction, directed interaction and technology-mediated interaction. *General interaction* addresses statements of relationship or influence between a quality characteristic and sustainability aspects. This category includes the effort of incorporating sustainability as a quality characteristic into a quality model, measurement of sustainability aspects based on software attributes considered into a quality model, prioritization of quality characteristics and sustainability aspects, as well as a modeling of the influence quality characteristics and sustainability aspects have on each other. From a general interaction, a directed or mediated interaction can be derived. *Directed interaction* describes either a positive or negative interaction between a quality characteristic and a sustainability aspect. The direction of the interaction can also be established in a lower abstraction level, since quality characteristics, as well as sustainability aspects, can be described by more specific terms. Finally, *mediated interaction* describes the influence of a technology (e.g. programming language, software library, among others) both on quality characteristics and on sustainability aspects. When the paper addresses only a specific item of technology, such as an energy-aware library, the influence can either 'contribute' (positive) or 'hinder' (negative) the achievement of both statements simultaneously. In addition, if a paper describes a set of items belonging to the same technology category (e.g. several programming languages) the label 'affect' is used, since results sections provided in papers allow the items to be sorted considering energy consumption.

Software engineering process to Software quality describe approaches for managing interactions categorized by the SWEBOK knowledge areas, while Software product quality view (Section 4.2.2) presents interactions classified by ISO/IEC 25010 product quality characteristics.

Software engineering process: There is a philosophical paper whose main purpose is to introduce three sustainability processes into a software organization [62]. In addition, these authors consider sustainability as a new product quality characteristic that has a relationship with usability, functionality and reusability. On the other hand, Manotas et al. [47] reported that it is frequent to make tradeoffs between energy usage and performance of a software feature. Achieving a balance between energy usage and other software goals is considered in design, construction and testing stages [47].

Software maintenance: The software maintenance knowledge area is addressed by eight papers. Since refactoring techniques have been used to increase software maintainability, these papers explore their potential support to improve (or maintain the same level) software sustainability. The evidence about using refactoring patterns has not been clear until now, since researchers have reported both a positive [63] and a negative [64] influence on energy efficiency. Other researchers have found similar results when evaluating energy consumption of refactoring techniques [65] [66]. Refactoring techniques have also been reported in templates addressing energy code smells [67]. Jagroep et al. [68], for their part, reported the effect of software change when an encryption component is added to a commercial software product. Finally, some researchers provided arguments to describe the effect that sustainability and maintainability subcharacteristics have on each other [69] while others considered some reengineering techniques to improve energy efficiency for mobile applications [70].

Software construction: The software construction knowledge area is addressed by nine primary papers. Two evaluation research papers studied software developers' energy aware messages committed in GitHub archives [46] [48]. Moura et al. found that developers are trading software energy consumption for other software attributes [46]. In the process quality view, authors found that ill-chosen energy saving techniques and libraries can produce incorrect results. The other paper found that energy-related code changes tend to be larger than bug fixes [48]. In addition, energy bugs, errors in systems that unexpectedly increase energy consumption, also show negative interactions between energy efficiency and software process quality. Finally, enhancing the user experience requires more power consumption [46]; there is thus a negative interaction between quality in use and energy efficiency.

A solution research paper proposes an aspect-oriented programming approach and a tool for profiling dynamically-consumed energy at software component level [56]. Remaining papers are validation research papers that reported on experiments to show the impact on energy efficiency of technologies used to optimize computing resources and time performance. In particular, they addressed the effect on energy consumption of different languages [71], validating practices for energy efficiency in the MySQL server [72], several sorting algorithms [73], coding practices [74–76], compilation options and implemented data structures [71]. Some authors argued that the duration of the task explains more than 89%% of energy consumption when sorting algorithms were tested in mobile devices [73]. Moreover, practices use to enhance software performance have also been studied in the context of energy efficiency [75]. Indeed, a memoization framework can reduce both energy consumption (average saving of 86%%) and time performance (average saving of 87%%) [76].

Software design: The experience report provided by Ardito et al. [6] included a conceptual framework that considers both power consumption profiles of computer resources and design strategies as a basis for improving energy efficiency while developing software. The eight validation research papers addressed topics such as: design patterns [77,78]; the impact of web servers on web application energy consumption [79]; about energy consumption, the impact of frameworks on access to data in web-based applications [80], how to develop GUIs in the quest to reduce energy consumption [41,81]; software architecture and program features that influence energy consumption [82]; and use of some general techniques for computing efficiency, data efficiency and context-awareness as to improve energy efficiency [83].

Software requirements: Nine papers address software requirements in a process perspective by using terms such as goals, nonfunctional requirements, quality requirements, software qualities and indicators [10,20,43,84–89]. Modeling languages can thus support the identification of sustainability goals and means that contribute positively or negatively within a specific system goal [84,86,88,89]. Considering the quality in use viewpoint, how the research on users can identify a set of principles for designing software systems that enhance the user experience to motivate people to make behavioral changes is described in [90] and [91]. Furthermore, other research [92] showed the user's role in energy consumption when he/she selected an application for executing a task.

In order to achieve sustainability goals, the software product should take also into account other nonfunctional requirements such as usability by means of requirements elicitation techniques [93] or usability testing [94]. Indeed, if the requirements of a user-friendly GUI are not met, software may not achieve sustainability goals [85]. Finally, Saputri and Lee [95] proposed a framework for addressing sustainability requirements by considering both conflicting goals and a technique for prioritizing requirements.

Software quality: In this category, there is an evaluation paper whose aim is to understand the relationship between software quality and energy efficiency [45]. This paper reported that energy efficiency has a negative relationship with functional suitability and performance efficiency. On the other hand, reliability, security and usability are perceived to have a potential positive interaction with energy efficiency.

Other proposals discussed the way in which sustainability can be incorporated into the ISO/IEC 25010 quality model [12,49,105] or measures for a greenability quality model [50]. In the category of validation research papers, Beghoura et al. [96] proposed an extension to

ISO/IEC 25010 to include green efficiency quality characteristics. In addition, Ouhbi et al. [97] provide a survey of blood donation mobile applications that considers both sustainability dimensions and product quality attributes.

A framework entropy indicator to evaluate the impact of libraries and development frameworks on energy consumption is proposed in [5]. Authors found that using frameworks and external libraries has a positive effect on developing small and medium software applications, while they have negative impact on the development of large applications [5]. On the other hand, other researchers attempt to identify software metrics that can be used as indicators to measure power consumption in mobile applications [57]. With respect to indicators, other research [98] proposed a GreenUp indicator to evaluate energy efficiency.

An energy profiler can help in the analysis of legacy systems to identify highly energy-consuming components [51], can assist in the identification of abnormal energy consumption in mobile applications [52], and could support developers in profiling their code for energy consumption [53]. In addition, it is possible to profile energy usage of several computing resources in the attempt to guide design choices [42]. Profiling energy tools can also take into account usability [55] and security [99]. Moreover, Chang et al. [100] proposed an analysis methodology to establish indicators for resource utilization.

Another topic studied is the effect of software change on energy consumption. With reference to system calls, a study determines the energy consumption profile of open source software (OSS) relying on changes among different versions [54]. Ahmed et al. [101] found that there is a negative correlation between the number of bugs fixed and the amount of power the OSS application consumes. Moreover, system call profiles for Android applications are used to estimate energy consumption [102]. Finally, there are proposals to prioritize quality and sustainability characteristics [103] and to develop carbon footprint calculators [104].

4.2.2. Interactions between software quality and sustainability aspects (GQ2-RQ2)

The following paragraphs describe interactions found in the selected papers for the process quality view, the product quality view and the quality in use view.

Software process quality view: Defects counting is a common way to measure software quality from the process perspective [37] and it is also studied with regard to energy consumption. As a matter of fact, Zhang and Hindle [54] reported that bugs in software versions can make their power use seem extremely high or extremely low. Chowdhury and Hindle [48] found that energy-related code changes tend to be larger than fixing a bug, and that energy concerns can affect several modules in a software product. In addition, Aggarwal et al. [102] found that some versions of a software product have different energy consumption patterns derived from errors caused by two incomplete refactoring efforts; these errors can cause an increase in software energy consumption. Furthermore, Ahmed et al. [101] found a negative correlation between the number of bugs fixed and amount of power OSS applications consumes. Moreover, Moura et al. [46] reported energy bugs, such as keeping the system in the wrong c-state, or continuously updating a UI component when the screen is turned off, can increase energy consumption.

Within the process quality view, there also exist analysis activities related to identifying interactions between software requirements or user goals [23]. Several pieces of research have extended the goal-modeling approach to identify potential conflicts between sustainability and software goals [84]. Interactions can be discovered in the goals hierarchy in terms of conflicts, constraints or support between goals. In addition, an example of eliciting sustainability requirements for a medication adherence system is included in [85]. Furthermore, a matrix to document conflicting goals among sustainability requirements explicitly, along with a method to prioritize goals is proposed in [95]. We could also point to the relevance of the identification of sustainability indicators in a software system, as proposed by Rodriguez and Penzenstadler [87].

Goal modeling is also applied in sustainable requirements engineering by considering modeling approaches such as KAOS and i*. Bomfim et al. [86] reported an experience of developing a procurement system using the KAOS framework. The goal models can be analyzed by means of positive or negative contributions between goals, including sustainability goals. Cabot et al. [43] used the i* framework to model sustainability factors and goals. The interdependencies among goals can have either a positive or a negative effect on goals.

Based on an architecture evaluation model, Lago et al. [20] proposed a sustainability framework to capture sustainability aspects based on sustainability dimensions. The model allows the identification of influences, support or conflict, between different parameters of sustainability dimensions. Since addressing environmental sustainability requires a balance between sustainability dimensions, the resolution of a conflict can be handled by prioritization techniques and negotiation among stakeholders [10].

Software product quality view: The interaction between product quality characteristics and sustainability aspects is based on the quality characteristics described in the product quality model of the ISO/IEC 25010. In order to understand the extent of a directed interaction, particular subcharacteristics (which belong to the product quality model mentioned) or particular concerns of a sustainability aspect alluded to, are examined. Table 2 presents a summary of directed interactions, while Table 3 shows the main technologies explored in mediated interactions. The bold type in the tables indicates that the paper is either a validation or evaluation research paper. As such, they are empirical [34]. For each quality characteristic, this section describes interactions with sustainability aspects in the following order: general interaction, directed interactions and mediated interactions.

Functional suitability: General interaction. Calero et al. [12] propose a Bayesian Network model (BNM) which considers that functional appropriateness is related to the user's environmental perception. This latter concept refers to the user's perception about the effect of software in the environment. It is a subcharacteristic of greenability in use. Other proposal investigates the interdependencies between product quality characteristics and environmental sustainability criteria, seeking to conduct a trade-off analysis [103]. The researchers found that functionality is the third most important quality criterion where its accuracy subcharacteristic represents 44%% [103]. Finally, Cordero et al. [51] reported the relationship between executed functionality and energy consumption.

Directed interaction. As far as the direction of an interaction is concerned, papers show both positive and negative categories. On the one hand, Zhang et al. [92] compared energy consumption of applications with equivalent functionality for completing the same task. They found that software versions with limited functionality consume less energy. Indeed, authors noted that “more functionality often leads to increased energy consumption”. Similarly, Lami and Buglione [62] defined that a sustainable product should “meet current needs of required functionalities without compromising the ability to meet future needs” (p. 56) and should implement the required functionalities in order to reduce maintenance tasks [62]. For their part, Koçak et al. [45] found that functional suitability has a negative correlation with energy efficiency: “an increase in the functional suitability […] leads to a decrease in the energy efficiency”. However, resource efficiency has a positive correlation with functional suitability [45]. This result therefore points out that functional suitability has both positive and negative interaction with sustainability criteria.

Mediated interaction. Two approaches were found when assessing technology's contribution to sustainability aspects considering functional suitability. One approach studied the influence that different application categories have on energy efficiency. Categorization of applications relies on the functions provided by the software under

Table 2
Directed interactions between software quality and sustainability aspects in the product quality perspective.

Quality concepts	Sustainability aspects	Directed interaction	Papers (non-empirical / **empirical**)
Functional suitability (functional appropriateness, functional completeness)	Energy efficiency, resource efficiency[a]	Positive	[92,62] **[45]** [a]
Functional suitability	Energy efficiency	Negative	**[45]**
Maintainability (modularity)	Power consumption, greenability	Negative, positive	[69]
Maintainability (reusability)	Greenability	Positive	[69,62]
Maintainability (modificability)	Greenability	Positive	[69]
Reliability	Energy efficiency, environmental sustainability	Positive	[20,**45**]
Security (confidentiality)	Sustainability	Negative	[93]
Security	Energy efficiency	Positive	**[45]**
Usability	Perdurability, sustainability, environmental sustainability (resource efficiency)	Positive	[62,105,85] **[45]**
Performance efficiency	Energy efficiency[b] (power consumption)	Positive	[6,56] **[45,98,95,72]**
Performance efficiency	Energy efficiency	Negative	**[45]**
Performance efficiency (time behavior)	Energy efficiency[b] (energy usage)	Positive	[53,52,70] **[65,83]**

[a] *Resource efficiency* is a quality subcharacteristic within the environmental factor in Koçak et al. [45].
[b] indicates that text segments refer to using or consuming either energy or power. These were changed to *energy efficiency* in order to be consistent with remaining papers.

study. The second approach focused on functional correctness, where the purpose is to show that changes in the software code for improving energy consumption do not produce incorrect outcomes.

Considering the functions implemented in software, Capra et al. [5] found that applications that present higher degrees of functional complexity (e.g. ERPs, games, editors) consume more energy than simple software such as calendars. They concluded that the "functional complexity of an application, and in particular the extent to which its execution is processor intensive, negatively impacts its energy efficiency". Similarly, Ouhbi et al. [97] found that the type of functions deployed in blood donation applications influence the energy consumption. In addition, Chang et al. [100] found that the way with which functions are implemented affects energy consumption, as does the expected behavior.

When assessing correctness of software outputs, Noureddine and Rajan [77] compared software outputs before and after applying design patterns in order to verify that their modifications do not change the functionality of code. Based on mining data in Github server, Moura et al. [46] found that ill-chosen energy-saving techniques could affect correctness of software.

Compatibility: General interaction. In relation to the subcharacteristics which belong both to compatibility and greenability in use characteristics, in the BNM [12] the interoperability is related to the user's environmental perception. In addition, Koçak et al. [103] found that interoperability has a very low priority among quality characteristics when environmental sustainability is considered.

Portability: General interaction. Based on the subcharacteristics of portability and greenability in use, adaptability and replaceability are related to the user's environmental perception [12]. Furthermore, the generic sustainable software model proposes that portability promotes software perdurability [105].

Maintainability: General interaction. Three papers discuss the potential relationship between greenability and maintainability. In a BNM [12], modularity, reusability and analyzability are related to the user's environmental perception. Moraga et al. [50] identified a set of measures that can be related to the perdurability subcharacteristic, which is defined in terms of modifiability, adaptability and reusability. In addition, Amri and Bellamine Ben Saoud [105] proposed that maintainability should contribute towards software perdurability.

Directed interaction. A paper argued that maintainability can have both a positive and a negative influence on greenability [69]. Modularity is considered to increase power consumption due to the fact that fine-grained software architecture increases the number of messages sent between objects. On the other hand, a modular software architecture increases software modifiability. Furthermore, reusability is also considered to contribute towards green software [69] or a sustainable product [62].

Mediated interaction. On the one hand, a solution proposal paper reviewed several software reengineering techniques used to improve software maintainability, and proposed them to enhance efficiency of software energy [70]. On the other hand, empirical data provided by the authors in [64,66] showed that refactoring techniques applied to increase software maintainability can consume more energy in the updated software.

Reliability: General interaction. In a BNM [12], reliability is related to the user's environmental perception subcharacteristic. Furthermore, Koçak et al. [103] found that reliability is the most important quality characteristic when considering the environmental sustainability criteria.

Directed interaction. On exploring the correlation between software quality and energy efficiency, it is reported that reliability has the highest positive effect on energy efficiency [45]. In modeling sustainable systems, Lago et al. [20] showed that the system requires a reliability parameter.

Security: General interaction. Security is not considered to influence greenability in use [12]. In addition, security might have very low relevance when environmental sustainability criteria are taken into account [103].

Directed interaction. In the exploratory survey conducted by Koçak et al. [45], the authors found that security has a significantly positive influence on energy efficiency. The authors related security to reliability, in the sense that enhancing both may help to reduce faults and defects. However, another researcher argued that security and sustainably may have a negative interaction. Penzenstadler [93] noted that "the absolute availability of information would make sustainability easier in many domains, and that is in conflict with privacy". Privacy is related to security through the confidentiality subcharacteristic.

Mediated interaction. Three papers describe the software effect on the relationship between security and energy efficiency. Sabharwal et al. [83] compared two windows systems running in laptops based on the duration of battery life. They found that running security applications on the IT build degrades battery life roughly 14%% more than in a system without security application. Jagroep et al. [68] found that incorporating an encryption component into a commercial software product increases energy consumption. In the third paper, Ambrosio et al. [99] reported that the filtering app under study contributes positively to both energy efficiency and security.

Usability: General interaction. Calero et al. [12] identified two subcharacteristics of usability, appropriateness recognizability and learnability, which are related to the user's environmental protection

Table 3
Mediated interactions between software quality and sustainability aspects in the product quality perspective.

Quality concepts	Sustainability aspects	Technology	Mediated interaction	Papers (non- empirical/**empirical**)
Functional suitability	Resource consumption, energy consumption	The way functions are implemented, functional complexity, type of functionality offered by a category of software product	Affects	[100,51], [97]
Functional suitability	Energy-saving	Ill-chosen energy-saving libraries	Hinders	[46]
Functional suitability	Energy-saving	Transformation of design patterns	Contributes	[77]
Maintainability	Energy efficiency	Program analysis techniques	Contributes	[70]
Maintainability	Energy efficiency	Refactoring techniques	Hinders	[66,64]
Security	Battery life, energy efficiency	Running security applications, Incorporating an encryption element in software architecture	Hinders	[83,68]
Security	Battery consumption	Filtering adware in mobile systems	Contributes	[99]
Usability	Sustainability requirements, carbon footprint, energy efficiency	Improve usability of: recommender system, carbon footprint calculator. Using UI-based design techniques	Contributes	[94,55] [104,40]
Performance efficiency	Energy efficiency	Practices for enhancing performance of mobile devices, compressing file streams, memoization techniques, filtering ads in mobile web-based systems, energy-aware libraries	Contributes	[42], [76,99,63]
Performance efficiency	Energy efficiency	Decorator design pattern, tradeoffs in all stages of software development, energy issues are difficult to discover	Hinders	[46,78]
Performance efficiency	Energy efficiency	Programming languages, runtime scope of android-based systems, video quality level, sorting algorithms, video compression strategies, using libraries and frameworks, compiler optimization flags, data structures, web server, web application features, PHP frameworks, program size, using cache	Affects	[67], [75,74,96,100,5,73,71,79,80,82]

subcharacteristic. Moreover, Amri and

Bellamine Ben Saoud [105] proposed that usability is related to perdurability.

Directed interaction. Lami and Buglione [62] noted that sustainable software should consider its usability, since it can contribute to extending the product life cycle beyond what was initially expected. In addition, an easy-to-use graphical interface contributes to achieving sustainability goals [85]. As a result of an exploratory survey, Koçak et al. [45] found that usability has the highest positive impact on resource optimization, a criterion belonging to environmental sustainability.

Mediated interaction. Several researchers indicated that software usability contributes to sustainability goals. Usability testing is recommended and applied to identify sustainability requirements and to support the usage of sustainable software (e.g. carbon footprint calculators) [94,104]. In addition, Jolinar, a profiling energy tool, is presented as an easy-to-use application for both software developers and users [55]. Furthermore, other researchers have proposed techniques to enhance a user interface design that contributes positively to energy efficiency without affecting software usability [41].

Performance efficiency: General interaction. In the BNM [12], performance efficiency is related to the user's environmental perception. In addition, a main finding in [103] was that energy impact and resource usage, both belonging to environment criteria, are related to time behavior. Similarly, the generic sustainability software model in [105] considers that the software energy consumption can be controlled by runtime efficiency and usage of system resources. Looking for making greenability characteristics operative, Moraga and Bertoa [50] found that an important number of measures are related to resource optimization. These measures evaluate the consumption and optimization of different resources that a software product uses. In other quality model [49], resource saving, as a green software subcharacteristic, is related to resource utilization. Moreover, green efficiency is considered as a quality characteristic by Beghoura et al. [96] and it is composed of subcharacteristics related to performance efficiency. Furthermore, Koçak et al. [45] found that performance efficiency has a significant energy-related impact. A similar idea was considered by Cordero et al. [51], to correlate an application's execution time and its energy consumption.

Directed interaction. Ardito et al. [6] indicated that resource usage metrics are key predictors in building resource-based power consumption models. Despite the fact that the most important resource to be monitored is the CPU, others resources, such as memory usage or input/output operations, have a significant correlation with power consumption. In contrast, Koçak et al. [45] found that an increase in performance efficiency leads to a decrease in energy efficiency. Manotas et al. [47] reported that it is difficult to address the tradeoffs between performance and energy when software is in the development stage.

Considering software time behavior, researchers reported correlations between execution time and energy consumption [95]. Studying several optimization techniques, Sabharwal et al. [83] noted that “the faster we can complete the workload and get the computer back to idle, the more energy we can save.” Jelschen et al. [70] argued that execution time is strongly related to energy consumption in the context of the use of dynamic analysis and refactoring techniques for optimizing the energy profile of mobile applications. However, others researchers look at a weak correlation between execution time and energy consumption. Sahin et al. [65] concluded that factors beyond execution time influence energy usage [65] while other authors concluded that either the correlation is not direct [53] or there are other factors to be taken into account [52].

The type of resource used and the way they are managed affect energy consumption. Jelshen et al. [70] reported that performance requires more energy when powerful devices are used (e.g. energy demands of a laptop vs. mobile phone). Using a tool to automatically profile energy consumption of applications can increase performance

Table 4
Findings summary.

Main findings	Relevant information
GQ1-RQ1. Publication trends	• In 2006, we found the first reference of interactions between energy efficiency and usability in the mobile app domain
• More than 75%% of primary papers published in the last three years	• Specialized conferences: GREENS, RE4SuSy. SE conferences: ICSE, ESEM, SAC
• In the last years specialized conferences about sustainability and sustainability tracks in SE conferences have arisen	
GQ1-RQ2. Software engineering areas	• The majority of papers were classified in the knowledge areas of software quality and software requirements. The other areas addressed are: software design, software construction, software maintenance, and the software engineering process
• Software quality: 22 papers	• Only 6 areas (out of 15) of SWEBOK were addressed
• Software requirements: 16 papers	
GQ1-RQ3. Research type	• More than 50%% of primary papers describe an empirical study
• Evaluation research papers: 4	• Trend: exploring current technologies about the extent to which they can address sustainability aspects
• Validation research papers: 30	• There is a need for empirical studies in industrial settings
GQ1-RQ4. Software quality approaches	• All ISO/IEC 25010 product quality characteristics addressed
• 73%% of papers in product quality view	• Performance efficiency studied by experimental design
• Performance efficiency is the quality characteristic studied most	• Improvements to quality models uses ISO/IEC 25010
• 24%% of papers are aware of ISO/IEC 25010	• 33%% of papers reported a hardware-based instrumentation to measure power consumption
GQ2-RQ1. Approaches for dealing with interactions	• Goal modeling techniques allow both positive or negative contribution of subgoals to be analyzed
• Modeling goals methods can be used to elicit and analyze sustainability goals	• Approaches to profile energy in software applications are based on hardware or software instrumentation
	• Incorporating sustainability as a quality characteristic into ISO/IEC 25010
• Energy profiles for software applications can monitor energy efficiency	• Software design patterns and refactoring techniques need to be studied in depth, to discover the way they affect energy consumption
• Quality models incorporate a sustainability characteristic	• Recommended practices to enhance performance can also optimize energy efficiency
GQ2-RQ2. Direction of an interaction	• Enhancing software process quality has a potentially positive interaction with energy efficiency
• Fixing defects contributes to optimizing of energy efficiency	• Functional suitability, performance efficiency and maintainability can participate in both positive and negative interactions with sustainability
• Usability and reliability have a potential positive interaction with sustainability	• Security has a potential negative interaction with sustainability.
• Performance efficiency can have both positive and negative interactions	• Reliability has a potentially positive interaction with sustainability
	• Usability can have a positive interaction with sustainability when addressing indirect sustainability impacts. Direct sustainability impact can have a negative interaction with usability

by reducing resources used and can maintain similar energy consumption of manual approaches [56]. Other researchers reported that by improving the number of operations executed by code and optimizing the usage of main memory and cache, energy can be saved [98]. Moreover, there is a correlation between resource usage and energy consumption that can lead to trade-offs between energy efficiency and performance [72].

Mediated interaction. In the experiments to characterize power consumption profiles when design patterns are used, Sahin et al. [78] found that decorator pattern flexibility requires more work at runtime, which can explain the amounts of energy use. They concluded that design patterns should be studied in the context of design and implementation decisions to evaluate their impact on energy efficiency.

Kansal and Zhao [42] reported that using a compressed file stream saves energy, and enhances performance [42]. Agosta et al. [76] found that by applying a memoization-based framework they can improve both average energy saving (86%%) and average performance (87%%). In [99], filtering techniques of a privacy-enhancing component for Android systems were explored. As a result, time savings were obtained, which ranged from 5%% to 28%% on the web sites assessed, while the energy consumption savings ranged from 3%% to 29%%. Furthermore, it is reported an improvement in the energy usage by around 3%% by using a framework for an energy-aware library for Java Collections API [63].

Other topic addressed is the effect on energy consumption of a set of technologies of the same kind (e.g. programming languages [74]). Rashid et al. [73], found that both the algorithms and the language significantly affect the total energy consumption. Li and Halfond [75] reported that practices recommended to enhance the performance of mobile applications also improve energy efficiency from 10%% to 30%%. Furthermore, Beghoura et al. [96] reported that the greenest efficient sorting algorithms are the merge sort and the quick sort. Abdulsalam et al. [71] contributed by studying the effect of languages such as C, C++++, Java and Python and compiler flag options in the efficiency of three programs. They reported that programming languages, compiler options and data structures affect energy use [71]. Rajan et al. [82] found that several practices, used to improve performance, affect energy consumption. However, software practitioners consider that compromises between energy efficiency and performance, along with other requirements, should be done in several stages of the software development life cycle, and that to make informed decisions they need more information [47]. In addition, some refactoring techniques optimize energy efficiency as well as performance [67].

Other specific technologies have been studied. Beghoura et al. [96] reported that high-quality videos consume more energy than a standard-definition quality of a video. Moreover, Chang et al. [100] described that differences in energy consumption in systems such as Vimeo and YouTube can be explained by the larger workload in data receptions, as well as by higher variances in CPU interruptions derived from the adoption of different video compression strategies. Furthermore, Manotas et al. [79] found that energy consumed by a web application depends on the web server and the web application features. Moreover, high-performance production web servers may be detrimental as far as energy use is concerned [79]. Finally, Procaccianti et al. [80] found that using object-relational mapping frameworks have an impact on energy consumption.

Quality in use view: Developing sustainability-aware applications that engage users in sustainable behavior relies on understanding users' perceptions and values about sustainability. Based in a market research [90], Marcus et al. proposed a set of principles for designing sustainable applications that are more engaging. The experience of making a rewards program based on sustainability actions incorporate user

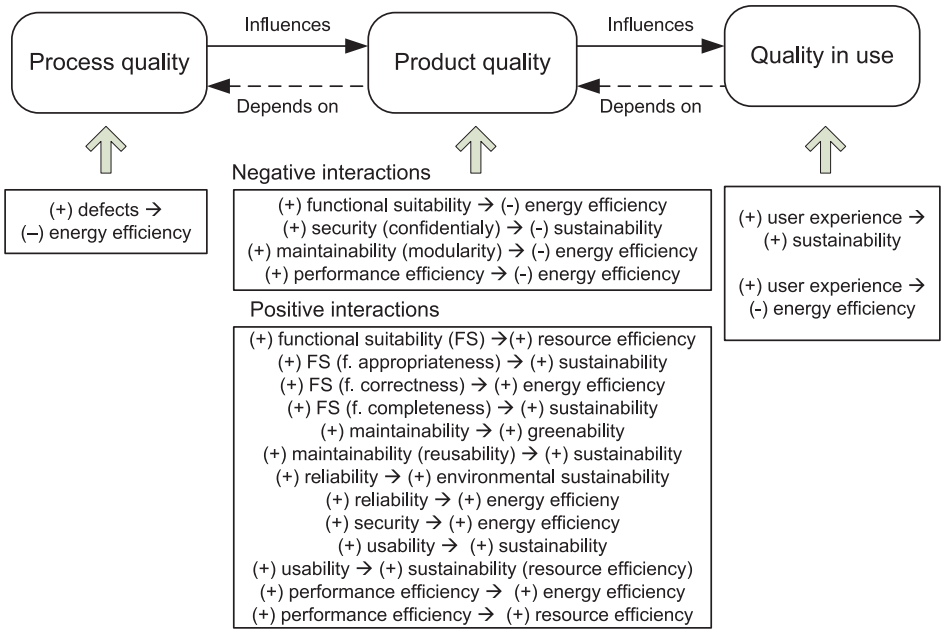

Fig. 7. Potential interactions between software quality and sustainability aspects (adapted from ISO 25010).

experience principles and several software components such as a success tracker and a game [91].

Moura et al. [46] found that better responsiveness on mobile devices needs more power consumption, due to an on-demand sampling rate. In addition, Yu et al. [81] proposed an energy-efficient engine, E3, to adjust the frame rate when scrolling is used. They reported up to 35% % energy saving using the framework, and found positive user experiences when applications such as Facebook, Google Maps, or browsers are used. Another proposal focused on defining the greenability in use characteristic [12], which is composed of three subcharacteristics: efficiency optimization, user's environmental perception, and minimization of environmental effects.

5. Summary and implications of the SMS

Table 4 presents a brief response to research questions described in Section 3. Based on the quality in life cycle model (ISO/IEC 25010), Fig. 7 is an overview of the answer for the question GQ2-RQ2 interactions between software quality and sustainability aspects considering the three quality views: process quality, product quality and quality in use. Concerning process quality, defects in software can increase the energy consumption of software. On the other hand, interactions in the quality in use view shows that improving user satisfaction with software can promote user participation in sustainable behavior. However, high responsiveness in touch events may increase energy consumption.

The main topic of this study is the product quality view using ISO/IEC 25010 (Tables 2 and 3). Only a few papers investigated the functionality of application software and its energy consumption. The more functions a software product provides, the greater its energy consumption [5,92,100]. This statement is related to functional suitability through the functional appropriateness subcharacteristic [12]. Both positive and negative correlations are reported between functional suitability and an environmental criterion [45]. Since the correlations were obtained by an exploratory survey, these findings need to be verified. Further work is thus needed to define the way functional suitability should be measured in experiments, to determine whether functional suitability promotes energy efficiency or not.

Compatibility and portability are addressed as general interactions. Reliability is considered as having a relationship with sustainability [12,103] and this relationship might be positive [20,45]. However, there is a need for experimental studies to confirm the relationships.

Security is poorly addressed in the selected papers. With regard to directed interaction, data from a regression model showed a positive interaction between security and energy efficiency [45], but the paper lacks a discussion of this correlation. Apart from that, privacy might have a negative influence on sustainability [93]. Some papers supporting this idea showed that adding security components into a system could increase energy consumption [68,83,99]. Hence, the main concerns about security are to investigate both to what extent security components are energy efficient and whether an application software has implemented appropriate security mechanisms for it to meet its purpose.

Another positive relationship is that existing between usability and sustainability goals. In this case, it is necessary to consider how the set of usability subcharacteristics behave, taking into account different sustainability aspects (e.g. energy consumption). In relation to maintainability, there is a common belief that it might influence the sustainability of application software positively [62,70]. In contrast, a few papers with experimental data showed that some refactoring techniques have a negative impact on energy efficiency due to the fact that new methods and classes incorporated into the updated software might increase the number of messages between objects (increased modularity) [64,66]. A limitation of these experiments is that they need to address larger software systems, as well as to determine accurate and complete execution scenarios [64]. Different contextual settings should be investigated to understand the extent to which the technology used to increase maintainability has a negative impact on energy efficiency. Furthermore, the relationship between sustainability and maintainability should be studied in software, considering both a static view (design) and a dynamic view (running).

As Table 2 shows, a number of statements about interactions do not rely on sound empirical studies, and several quality concepts used in papers are not defined. This means that selected papers discuss *potential interactions* between software quality characteristics and sustainability aspects. Although a quality assessment was conducted on empirical papers, the majority of them are validation studies that need to be evaluated in an industrial context. The lack of empirical evidence in industrial settings does not hinder these results from being used in software projects, since perceptual attributes are important for establishing a basis for general theory [106]. In addition, the lack of appropriate methods to detect conflicting interactions leads to the fact that an engineer “relies (almost) entirely on the intuition of experienced modelers” [61]; expert judgment is one of the most common

approaches for reconciling conflicts that arise in product software quality [32].

5.1. Research implications

Although 66 papers addressing interactions between software product quality and sustainability were selected, the analysis of these shows that research is in an exploratory stage. 34 papers were classified as empirical papers, but only four address practices in industrial settings. Two research strategies can therefore be derived. First, validating that current software development technologies to enhance a particular software product quality attribute also contribute to optimizing energy efficiency. Second, evaluating approaches both for improving sustainability aspects and for enhancing product quality in industrial settings. The research community effort has focused on the relationship between performance efficiency and energy efficiency, with performance efficiency being the one most related to sustainability aspects. Tables 2 and 3 provide an initial map of practices studied and quality characteristics addressed. These findings can become a starting point for a deeper analysis of either other software development practices or evaluation of practices in distinct contextual settings.

Reducing energy consumption is a main goal in developing sustainable software [17,22]. In the context of green in software, this goal should be addressed during the whole software life cycle [21]. Moreover, the resources consumed during the software development stage need to be considered from the sustainable development perspective, as do the resources required for software operation. Optimizing software considering sustainability aspects offers opportunities to identify other sustainability indicators [71]. There is thus a need to develop methods that allow software developers to elicit, model, analyze, and negotiate sustainability goals and requirements.

There are two main approaches to developing energy profiles of software applications: hardware-based and software-based. A main goal in the development of energy profiles for software is to establish a relationship between the behavior of software components and energy consumption measures. These models are at an exploratory stage and need additional validation in industrial settings. Furthermore, there is a need to compare methods for developing energy profiles to assure the consistency of results among them. For instance, Capra et al. [5] and Zhang et al. [92] assessed energy consumption of different software applications, categorizing them by the extent to which different applications have similar functions. However, there are differences in hardware/software platforms, measurement devices, and procedures used for gauging energy consumption, data acquisition and data analysis. There is thus an obvious need for information with which to make decisions about the selection of the best energy profile method for monitoring specific software categories.

An important topic addressed in primary papers is improving quality models to include sustainability as a quality characteristic into a quality model [13]. The majority of proposals consider adding a new quality characteristic called *greenability* or *sustainability*. Some proposals have decomposed the sustainability characteristic into subcharacteristics. However, a comparison of these models should be carried out in order to identify the main aspects that make it possible to consider sustainability as a general quality characteristic. For it to be a quality characteristic, development of empirical models, including appropriate measures is required. Currently there are few conceptual quality models, so there is further research work to be done on this topic.

The relationship between performance efficiency and energy efficiency is what is reported most frequently in the set of primary papers. The definition of performance efficiency relies on the requirements stated for a software system about the response time required to perform its functions, the amount and type of resources used when software performs its functions, and the maximum limits of product parameters [13]. Without considering the requirements specified for the software under study since they are not included in primary papers, enhancing performance efficiency has a positive interaction with energy efficiency. Other quality characteristics such as functional suitability, reliability, maintainability, security and usability, have also been explored in relation to sustainability aspects. In order to synthesize evidence of empirical studies, appropriate definition of quality characteristics should be provided.

Finally, some quality characteristics showed conflicting interactions with sustainability aspects. A way to resolve conflicting interactions is to refine the definition of the concepts under study [14]. It is thus suggested that in their research reports researchers should include a definition both of the quality terms used and the sustainability aspects addressed. Furthermore, it is important to study contextual factors that can have an impact on the interactions between two conflicting goals or requirements. Describing contextual data can support the identification of factors that need to be studied to explain why the conclusions reached by the different research papers are not consistent with each other.

5.2. Practical implications

Software developers need practical guidance to support sustainability during software development [23,107]. Although there are scarce results in industrial settings, we can rely on the expert opinion of researchers as a basis for providing some general suggestions. They must be adapted considering the particular settings of the project or organization.

1. In a process view, improving software quality by means of fixing defects can contribute to the enhancement of energy efficiency. Studies with different versions of the same open source software have showed that defects have an impact on energy consumption. Fixing defects can reduce energy consumption, and defect management practices also contribute to reducing energy consumption.
2. Goal modeling methods, such as KAOS or i*, can support the elicitation of sustainability goals. These models also allow software developers to identify sustainability goals and to relate them with business or system goals. Subgoals and the means to achieve a given goal can be evaluated to determine if they have a positive or negative contribution.
3. Using practices to improve performance efficiency can have a positive impact on energy optimization. From the analysis of primary papers, we found that performance and energy efficiency have a positive interaction. This means that among other things, software developers need to evaluate the feasibility of using practices such as memoization, prevent the sending of small HTTP packets, or minimize the execution in the background of irrelevant applications. Performance-enhancing practices should be considered as a first approach to optimizing energy efficiency when software developers lack appropriate and validated practices to deal with sustainability aspects. Table 3 provides a list of the technologies assessed with regard to energy efficiency.
4. Usability can contribute positively to sustainability when software developers aim to promote the use of a sustainable product among users. In addition, enhanced software usability allows users carry out tasks more efficiently. That being the case, if a software project needs to address sustainability aspects in order to promote a sustainable behavior (indirect impact on sustainability), enhancing usability is a first step to achieving sustainability goals. However, some usability requirements can increase energy consumption when they are implemented. In that sense they can hinder the achievement of energy efficiency goals (direct impact on sustainability).
5. Addressing some of these quality characteristics implies writing more code. But more code might consume more resources and more energy. Software developers should therefore identify the purpose and goals of the software under development and the extent to which sustainability should be addressed. As Sierszecki et al. [108]

pointed out in the domain of embedded software: “safety, reliability and availability are more important to drive operation than power consumption is.”

6. Study limitations

In a systematic review, there are two main threats that have an impact on the reliability of the SMS: the selection bias and the data extraction bias [8,40]. Considering the mitigation of selection bias, a protocol was developed to establish the goal of the research and the main research questions. The protocol identified the search terms and included a previous literature review about sustainability and software engineering.

The main purpose of this SMS is to provide a snapshot of the way software quality interacts with sustainability aspects, and to see the main trending topics researchers are working on. There is no need for an exhaustive search process to identify all the papers about the topic under review. Indeed the search procedure should be auditable, but not necessarily complete [109]. An automatic search process was carried out in four databases recommended for the software engineering field [110]. It should be added that this SMS includes both empirical and non-empirical studies [34] published in conference papers, book chapters derived from research results, and journal papers [34].

Although the main focus of this SMS is on product quality based on the ISO/IEC 25010 product quality model, the search string included terms to explore quality in the perspectives of process quality and quality in use (e.g. quality, model, *25010, *9126), but a few papers were classified in the “quality in use” category. The result of this SMS can be considered consistent with identifying measures for quality in use [22].

Data extraction bias was mitigated by developing a data extraction template. The template fields were filled with verbatim data (whenever possible) extracted from selected papers. In the case of text segments that address interactions specifically, the data extracted included the page on which the text segment appears.

Classification of papers, on the other hand, was based on models accepted by the software engineering community such as research-type papers, SWEBOK knowledge areas, and product quality characteristics [33,13,30]. Specific topics about software quality views and interactions were appropriately documented in the respective sections (e.g. 4.2). Classification of specific interactions was difficult, given the diversity of ways of expressing them. The interaction model used in this paper was derived from the analysis of selected papers. However, relationships found in a text segment could belong to different interaction categories. The way this issue was mitigated was to consider practices (or techniques) assessed and the generality of the claim. The energy profile proposals that supported an interaction direction were considered in the directed interaction category. However, this interaction model needs to be improved. Other papers (e.g. [47,95]) included text segments of different quality views. The text segments were classified in the appropriate quality view. Given that 67%% of papers were classified with only one text segment, it is possible that other papers may have addressed other quality views.

This SMS focused on identifying interactions or trade-offs between software quality and sustainability aspects. Although energy efficiency is the main concept studied under environmental sustainability, goals belonging to other sustainability dimensions are also addressed in proposals to elicit and analyze sustainability requirements. In addition, sustainability in software applications is tackled throughout the stages of the software development life cycle. This is consistent with the work of Bozzelli et al. [29], which found that application software is the context reported most in sustainability measures and that the measures can be used in different stages of the software development life cycle. Finally, the approaches used to manage interactions between software quality and sustainability concerns are consistent with the classification of software quality trade-offs [32].

7. Conclusions

This paper presents the result of conducting an SMS about the interactions between software quality and sustainability aspects in the context of application software during its development stage. The set of primary papers allowed us to answer the research questions and to identify approaches for managing interactions between software quality and sustainability aspects, as well as to see potential interactions between these.

The results highlight that researchers are actively working in the study of the interactions between sustainability and software quality, as it is observable in the notably increasing trend of publications. An important research approach is to explore the extent to which the current software development technology can be applied to save power consumption during software development. However, these technologies are still not fully used in industrial settings. Performance efficiency is the product quality characteristic reported most in validation research papers, but there is an important set of quality characteristics and subcharacteristics that need to be investigated in the quest to understand whether interactions with sustainability aspects arise and to identify relevant contextual factors that can trigger a conflicting interaction. Under these conditions, it is concluded that the research topic is at an exploratory stage.

From a practitioner's point of view, the SMS contributes by summarizing the influence of sustainability aspects on specific quality characteristics. Hence, they may use this information to make decisions about the way environmental sustainability goals can be included in a software project. In addition, the SMS provides a set of practices to deal with software product quality and sustainability goals that practitioners may explore when they are seeking to improve their software process. From a researcher's perspective, the output of this work provides a classification of the areas addressed, the empirical methods used, as well as the way software quality characteristics have been studied. The SMS provides an initial identification of papers so that further research can be started on topics barely addressed as yet.

Further work is needed to characterize interactions found in primary papers. Some of these are based on expert judgment that must be corroborated through the conducting of empirical research. Interactions found by experiments need to be assessed in different contextual settings in order to understand the extent to which they might arise when variations in contextual settings exist. A model to manage interactions and factors influencing these is needed if appropriate recommendations are to be provided to software developers. There is a need for methods and practices to develop green software without hindering software quality that supports all software life cycle stages. Indeed, empirical studies should be conducted in industrial settings so that the recommended practices can be evaluated.

Acknowledgments

This work has been funded by the project: GINSENG (Ministerio de Economía y Competitividad y Fondo Europeo de Desarrollo Regional FEDER, TIN2015-70259-C2-1-R).

Appendix A

The set of primary papers.

ID	Reference	Year	Paper type	SE area	Research type	Domain
abdulsalam2015	[98]	2015	Conference	Software quality (measurement)	Validation research	Four algorithms
abdusalam2014	[71]	2014	Conference	Software construction	Validation research	3 apps.: FFT, Linked list and Quicksort
agosta2012	[76]	2012	Journal	Software construction	Validation research	JavaGrande benchmark suite
ahmed2014	[101]	2014	Conference	Software quality (defect management)	Validation research	Open source software
ambrosio2014	[99]	2014	Conference	Software quality (measurement)	Validation research	Mobile apps
amri2014	[105]	2014	Conference	Software quality (quality model)	Philosophical paper	No described
ardito2015	[6]	2015	Journal	Software design	Experience report	Examples in tables: mobile
arggarwal2014	[102]	2014	Conference	Software quality (measurement)	Validation research	Mobile apps
beghoura2015	[96]	2015	Journal	Software quality (quality model)	Validation research	Mobile apps
bomfim2014	[86]	2014	Conference	Software requirements	Experience report	Energy sector in procurement systems
cabot2009	[43]	2009	Conference	Software requirements	Solution proposal	Conference planning
calero2014	[12]	2014	Workshop	Software quality (quality model)	Philosophical paper	No described
capra2012	[5]	2012	Journal	Software quality (measurement)	Validation research	63 open source applications
chang2014	[100]	2014	Conference	Software quality (measurement)	Validation research	Mobile apps: Youtube, Vimeo, Facebook, Twitter
chinenyeze2016	[56]	2016	Conference	Software construction	Solution proposal	Sorting apps
chowdhury2016	[48]	2016	Conference	Software construction	Evaluation research	Mobile apps, embedded apps, desktop
cordero2015	[51]	2015	Conference	Software quality (measurement)	Solution proposal	Java software
corral2014	[74]	2014	Conference	Software construction	Validation research	Rely on implementation of five algorithms
couto2014	[52]	2014	Conference	Software quality (measurement)	Solution proposal	Open source Android application
field2014	[53]	2014	Conference	Software quality (measurement)	Solution proposal	Sorting algorithms running in a laptop
garciaRod2015	[69]	2015	Book chapter	Software maintenance	Philosophical paper	Two open source software
gordieiev2015	[49]	2015	Workshop	Software quality (quality model)	Philosophical paper	Not described
gottschalk2016	[67]	2016	Conference	Software maintenance	Solution proposal	Mobile android-based apps
jagroep2016	[68]	2016	Conference	Software maintenance	Validation research	Commercial software product
jelschen2012	[70]	2012	Conference	Software maintenance	Philosophical paper	Not described
kansal2008	[42]	2008	Journal	Software quality (measurement)	Solution proposal	Windows-based software
keong2015	[57]	2015	Conference	Software quality (measurement)	Solution proposal	Mobile devices
kozac2014	[103]	2014	Workshop	Software quality (prioritization)	Solution proposal	Not described
kozac2015	[45]	2015	Workshop	Software quality (quality model)	Evaluation research	Not described

lago2015	[20]	2015	Journal	Software requirements	Solution proposal	Systems: paper-mill control and car-sharing
lami2012	[62]	2012	Conference	Software engineering process	Philosophical paper	Not described
li2014	[75]	2014	Workshop	Software construction	Validation research	Mobile apps
mahaux2011	[88]	2011	Conference	Software requirements	Experience report	Planning events system/business
manotas2013	[79]	2013	Workshop	Software design	Validation research	Web application: Tracks
manotas2014	[63]	2014	Conference	Software maintenance	Validation research	Java applications that use the Collections API.
manotas2016	[47]	2016	Conference	Software engineering process	Evaluation research	Mobile, traditional, embedded and datacenter apps
marcus2011	[90]	2011	Conference	Software requirements	Experience report	Not described
marcus2014	[91]	2014	Conference	Software requirements	Experience report	Web-based software tool and mobile apps
moraga2015	[50]	2015	Book chapter	Software quality (measurement)	Philosophical paper	Not described
moura2015	[46]	2015	Conference	Software construction	Evaluation research	Open source software
noureddine2015	[77]	2015	Conference	Software design	Validation research	Small programs gathered from Github repository (C++++, Java) Domain no specified)
noureddine2016	[55]	2016	Conference	Software quality (measurement)	Solution proposal	Application number Pi
ouhbi2015	[97]	2015	Conference	Software quality (measurement)	Validation research	Mobile devices apps
park2014	[66]	2014	Conference	Software maintenance	Validation research	63 C++++ programs
penzen2013	[23]	2013	Workshop	Software requirements	Solution proposal	Car-sharing system
penzen2014	[10]	2014	Journal	Software requirements	Philosophical paper	Not described
penzen2015	[85]	2015	Workshop	Software requirements	Validation research	Health domain
penzen2015A	[93]	2015	Conference	Software requirements	Philosophical paper	Web-based guide
penzend2015From	[84]	2015	Book chapter	Software requirements	Experience report	Car-sharing system
perez2014	[64]	2014	Journal	Software maintenance	Validation research	Open source systems
procaccianti2016	[80]	2016	Conference	Software design	Validation research	Web applications
procaccianti2016b	[72]	2016	Journal	Software construction	Validation research	Web applications
rahman2011	[104]	2011	Journal	Software quality (measurement)	Validation research	Application in .Net framework and mobile device
rajan2016	[82]	2016	Conference	Software design	Validation research	Desktop programs
rashid2015	[73]	2015	Conference	Software construction	Validation research	Mobile app
raturi2014	[89]	2014	Conference	Software requirements	Solution proposal	Joulery energy awareness tool, hypothetical Hotel Resource Tracking System
rodriguez2013	[87]	2013	Workshop	Software requirements	Solution proposal	Car-sharing system
roher2013	[94]	2013	Workshop	Software requirements	Philosophical paper	Not described
sabharwal2013	[83]	2013	Journal	Software design	Validation research	Mobile devices based on Windows
sahin2012	[78]	2012	Workshop	Software design	Validation research	Open source applications
sahin2014	[65]	2014	Conference	Software maintenance	Validation research	Java application (several domains)
saputri2016	[95]	2016	Conference	Software requirements	Validation research	Climate monitoring system

vallerio2006	[41]	2006	Journal	Software design	Validation research	Mobile applications
yu2015	[81]	2015	Journal	Software design	Validation research	Mobile apps
zhang2014	[92]	2014	Journal	Software requirements	Solution proposal	Text editing apps., email clients, and music players
zhangHindle2014	[54]	2014	Conference	Software quality (measurement)	Solution proposal	Open source software

Appendix B

Quality assessment of empirical papers.

ID	Aim	Context	Research design	Sampling	Control group	Data collection	Data analysis	Findings	Value of research	Total
abdulsalam2015	1	1	1	1	1	1	1	1	1	9
abdusalam2014	1	1	1	1	1	1	1	1	1	9
agosta2012	1	1	1	1	1	1	1	1	1	9
ahmed2014	1	1	1	1	0	0.5	0.5	1	1	7
ambrosio2014	1	1	1	1	1	1	1	1	1	9
arggarwal2014	1	1	1	0.5	0	0.5	1	1	1	7
beghoura2015	1	1	1	1	0	1	1	1	1	8
capra2012	1	1	1	1	0	1	1	1	1	8
chang2014	1	1	1	0.5	0	0.5	1	1	1	7
chowdhury2016	1	1	1	1	1	1	1	1	1	9
corral2014	1	1	1	0.5	0	0.5	0.5	1	1	6.5
jagroep2016	1	1	1	1	1	1	1	1	1	9
kozac2015	1	1	1	0.5	0	1	1	1	1	7.5
li2014	1	1	1	1	0	0.5	1	1	1	7.5
manotas2013	1	1	1	1	0	1	1	1	1	8
manotas2014	1	1	1	1	1	1	1	1	1	9
manotas2016	1	1	1	1	0	1	1	1	1	8
moura2015	1	1	1	1	0	1	1	1	1	8
noureddine2015	1	1	1	0.5	1	0.5	1	1	1	8
ouhbi2015	1	1	1	1	0	0.5	0.5	1	1	7
park2014	1	1	1	1	1	0.5	0.5	1	1	8
penzen2015	1	1	1	1	0	0.5	0.5	1	1	7
perez2014	1	1	1	1	1	1	1	1	1	9
procaccianti2016	1	1	1	1	0	1	1	1	1	8
procaccianti2016b	1	1	1	1	1	1	1	1	1	9
rahman2011	1	1	1	1	0	1	1	1	1	8
rajan2016	1	1	1	1	1	1	1	1	1	9
rashid2015	1	1	1	1	1	1	1	1	1	9
sabharwal2013	1	1	1	1	1	0.5	1	1	1	8.5
sahin2012	1	1	1	1	1	1	1	1	1	9
sahin2014	1	1	1	1	1	1	1	1	1	9
saputri2016	1	1	0	0	0	0	0.5	1	1	4.5
vallerio2006	1	1	1	1	1	1	1	1	1	9
yu2015	1	1	1	1	1	1	1	1	1	9
Value (1)	34	34	33	28	18	23	28	34	34	
Value (0.5)	0	0	0	5	0	10	6	0	0	
Value (0)	0	0	1	1	15	1	0	0	0	

Appendix C

Extracted parts of the SMS review protocol.

C.1. Search string

The selected databases in this SMS belongs to the set of databases suggested to conduct SMS on SE discipline with automated search [110]. In addition, [34] suggest selecting IEEE and ACM as well as two indexing databases (e.g. Scopus). The search string (Table 1) used in each database and its corresponding number of retrieved records is set out in Table 5.

Table 5
Operationalization of search string in four databases. Automatic search conducted at **December 28, 2016**.

Database	Search	Retrieved records
ACM (469 in the time span (2000-2016)	record Abstract: ((sustainab* OR green OR "energy efficient" OR ecolog*) AND (software) AND (quality OR goal OR nfr OR nonfunctional OR non-functional OR "quality requirement" OR "quality requirements" OR *9126 OR *25010 OR standard OR usability OR security OR compatibility OR functionality OR efficiency OR portability OR maintainability OR reliability))	469
IEEE		971
First search string: 435	(("Abstract":sustainab* OR "Abstract":green OR "Abstract":"energy efficient" OR "Abstract":ecolog*) AND "Abstract":software AND ("Abstract":quality OR "Abstract":goal OR "Abstract":nfr OR "Abstract":nonfunctional OR "Abstract":non-functional OR "Abstract":"quality requirement" OR "Abstract":"quality requirements" OR "Abstract":*9126 OR "Abstract":*25010 OR "Abstract":standard)) and refined by Year: 1999–2016	
Second search string: 536	(("Abstract":sustainab* OR "Abstract":green OR "Abstract":"energy efficient" OR "Abstract":ecology*) AND "Abstract":software AND ("Abstract":usability OR "Abstract":security OR "Abstract":compatibility OR "Abstract":functionality OR "Abstract":efficiency OR "Abstract":portability OR "Abstract":maintainability OR "Abstract":reliability)) and refined by Year: 2000–2016	
Web of Science	Tema: ((sustainab* OR green OR "energy efficient" OR ecolog*) AND (software) AND (quality OR goal OR nfr OR nonfunctional OR non-functional OR "quality requirement" OR "quality requirements" OR *9126 OR *25010 OR standard OR usability OR security OR compatibility OR functionality OR efficiency OR portability OR maintainability OR reliability)) Refinado por: Áreas de investigación: (COMPUTER SCIENCE) Período de tiempo: 2000–2016	932
Scopus	ABS (sustainab* OR green OR "energy efficient" OR ecolog*) AND ABS (software) AND ABS ((quality OR goal OR nfr OR nonfunctional OR non-functional OR "quality requirement" OR "quality requirements" OR *9126 OR *25010 OR standard) OR (usability OR security OR compatibility OR functionality OR efficiency OR portability OR maintainability OR reliability)) AND PUBYEAR > 1999 AND PUBYEAR < 2017 AND (LIMIT-TO (SUBJAREA, "COMP"))	1601
Total retrieved from 4 DB		**3973**

The search string was tested on Scopus database (http://www.scopus.com/) in order to verify at what extent we can retrieve the articles we are looking for (the procedure is detailed in Section 3.1). After examining the selected primary papers, we performed the automated searches on the databases of ACM digital library (http://dl.acm.org/dl.cfm), IEEE Xplore (http://ieeexplore.ieee.org/Xplore/home.jsp) and Web of Science (http://apps.webofknowledge.com/).

C.2. Selection criteria

Selection criteria are presented in Section 3.2 Study selection of the main paper.

C.3. Procedure for selecting primary papers

This SMS is carried out by the researchers included in Table 6.

The selection criteria were applied by the first author when reviewing the records retrieved from each database. First of all, the first author made the selection of the primary papers by applying the exclusion and inclusion criteria. This selection was made by reading only the title and the abstract. In parallel, the second author replicated the selection process and obtained a set of primary papers. The selected papers were retrieved in order to read them and verify that they met selection criteria. The two sets of primary papers were checked by all authors; discrepancies were discussed in order to determine whether it was appropriate to include the different papers or not, and the list was obtained.

C.4. Quality assessment

In an SMS, quality assessment of primary papers is not required [34] because an SMS can include papers without validation. However, in order to determine the extent to which rigor and relevance is reported in empirical papers selected in this SMS, we adapted a quality assessment instrument (Table 7).

The instrument is based on Dyba and Dingsoyr [40] and it is adapted to specific topics addressed in this SMS. The nine questions can be rated as included (1), partially included (0.5) and not included (0). Each question provides criteria to assign an included or partially included rating. Each primary paper was evaluated by the authors, and the final value assigned to each question was obtained by consensus.

Table 6
Researchers conducting this SMS.

Author-ID	Name	Area of expertise relevant for this SMS
1	G. A. García-Mireles	Software product quality, interactions between product quality characteristics
2	M. A. Moraga	Product quality models, green software measures
3	F. García	Software process models, product quality, measurement
4	C. Calero	Web-based quality models, product quality, green quality models, product quality measurement
5	M. Piattini	Systems and software quality

Table 7
Quality assessment instrument.

Number	Question	Included 1	Partially included 0.5	Not included 0
1	**Screening questions** Was there a clear statement of the aims of the research? • Article explicitly describes research aims or hypothesis (0.5) • Rationale for undertaking the study (0.5) • Article explicitly describes both (rationale and research aim) (1.0)			
2	**Screening questions** Was there an adequate description of the context in which the research was carried out? • The technology under development/study (mobile apps, web app, among others) was identified(0.5) • The hardware/software platforms where software is tested/running were described (0.5) • The technology under study, along with the platform and approach for measurement sustainability aspects or energy consumption were described (1.0)			
3	**Research design** Was the research design appropriate for addressing the aims of the research? • Paper justifies the main approach (qualitative methods, quantitative methods,) and a specific method for sampling, data collection and data analysis. (0.5) • Paper justified at least two methods used in research (e.g. one for data collection and one for data analysis). (1.0)			
4	**Sampling** Was the recruitment strategy appropriate to the aims of the research? • Researchers explained how the participants or cases were identified and selected (0.5) • Cases were identified and selected, and were defined or described precisely (1.0)			
5	**Control group** Was there a control group with which to compare treatments? • Researches described how controls were selected (0.5) • Researchers described how the controls were selected and used to contrast results (1.0)			
6	**Data collection** Was the data collected in a way that addressed the research issue? • All measures were clearly defined (e.g. units and counting rules) (0.5) • Description of data-collecting method (e.g. focus group, energy profiler, etc.) (0.5) • Both of these measures were clearly defined and description of data collection methods existed (1.0)			
7	**Data analysis** Was the data analysis sufficiently rigorous? • Only a superficial description of analysis method (i.e., only mention or refer to a method) was given (0.5) • In-depth description of the analysis process was given, and there were sufficient data to support the findings (1.0)			
8	**Findings** Was there a clear statement of findings? • Interactions were derived from related literature or discussion, or were not explicitly described but inferred (0.5) • Findings about interactions were explicitly described and derived by empirical result (e.g. magnitude of effect) (1.0)			
9	**Value of the research** Was the study of value for research or practice? • Value of research was poorly addressed (only one contribution was mentioned, without discussing how it was related to existing knowledge or understanding) (0.5) • Researchers discussed the contribution the study makes to existing knowledge or understanding (1.0)			

C.5. Data extraction and classification

A template was designed for extracting data from primary papers (Table 8). Each template field is related to the research questions.

To classify papers, we used several software engineering classifications such as knowledge areas of the SWEBOK [30], research type categories [33] and product quality characteristics (ISO/IEC 25010)[13]. However, specific interactions reported in papers need an appropriate classification.

Table 8
Template for extracting data.

Data item	Value	RQ
Study ID	Last name of first author + + publication year	
Article title	Name of the article	
Authors name	List of authors	
Year of publication	Calendar year assigned by the publisher. Some conference proceedings are published in the year following the conference date.	RQ1
Publication type	Peer reviewed articles (journal articles, conference and workshop proceedings, chapters in research books) **Excluded:** literature reviews and systematic reviews, since these were searched previously to determine the necessity of this work.	RQ1
Venue	Name of the publication venue	RQ1
Study goal	Extract either the study aim or goal	
Area in SE	Knowledge area in SWEBOK. We are considering the essential areas of SE such as: software requirements, software design, software construction, software testing, software maintenance, software configuration management, software engineering management, software engineering process, software engineering models and methods, software quality, software engineering professional practice, software engineering economics. **Excluded:** computing foundations, mathematical foundations, and engineering foundations	RQ2
Research type	Use [33] categories plus the rules recommended by [34]. Wieringa categories are: evaluation research, solution proposal, validation research, philosophical papers, opinion papers, experience papers.	RQ3

(*continued on next page*)

Table 8 (*continued*)

Data item	Value	RQ
Sustainability concerns	Describe the terms related to sustainability used in the paper. These include terms such as sustainability, greenability or energy efficiency. Describe the way energy/power was measured if it is included in paper.	RQ4
Quality terms used	Describe the quality terms the researchers include in the paper: **General terms:** these correspond to quality requirements, quality goals, quality concerns, software quality, technical quality, software defects. **Specific terms:** refers to both quality characteristics and quality subcharacteristics addressed in ISO 25010, including the model itself. Also includes other hierarchical quality models such as ISO/IEC 9126, among others.	RQ4
Approach to manage interactions	Describe the type of contribution presented in article (it can be a process, method, model, tool, and metric) or it may describe an empirical study (e.g. survey, interviews). When the article does not describe it, note this in this cell. Also include the name of the approach, as well as a brief description.	RQ5
Specific interactions	Describe the specific interactions addressed in the paper. Describe the influences between sustainability concerns and the software quality reported by the researchers, including both the potential and that validated by empirical research. Include the direction of the influence and the related terms. (This item corresponds to text segments.)	RQ5

Appendix D

Text segments used to classify interactions (partial list).

PaperID	Description	Type (ISO25010)	Direction	Interaction type
amri2014	Quality attributes such as **maintainability, portability and usability promote software perdurability. P. 233**	Maintainability	Relates	General
calero2014	From Table 1. All product quality characteristics, except security, have a relationship with sub-characteristics from greenability in use. (p. 7)	Maintainability	Relates	General
garciaRod2015	Modularity is likely to be related to greenability for two different reasons. p. 212	Maintainability	Negative	Directed
garciaRod2015	"Highly reusable assets are prone to be optimized, as is their greenability". P212	Maintainability	Positive	Directed
garciaRod2015	If an asset is easy to modify, it is likely to keep (and not worsen) its greenability. p. 212	Maintainability	Positive	Directed
jelshen2012	*Code smell detection* [36], the identification of patterns known or suspected to be detrimental to software quality (especially maintainability). In the same way, energy-wasting code patterns can be defined. This is tightly linked to *refactoring. P. 356.* Software reengineering aims to improve software quality. … maintainability… the same techniques are useful for driving software systems towards energy efficiency. P.356	Maintainability	Contributes	Mediated
lami2012	Sustainability would be more comprehensive than solely 'maintainability,' since it can be seen in the middle of the product quality model and quality in use model (p. 56.) Furthermore, a sustainable product should have a lower impact when introducing changes (reusability) or should improve usability that can make the product lifecycle longer (p. 57)	Maintainability	Positive	Directed
moraga2016	Majority of measures are related to energy efficiency (30 of the 81 measures for product), an important subset of measures are related to resource optimization (23%%) and perdurability (28%%) (p. 268)	Maintainability	Relates	General
park2014	Basically, code refactoring techniques have the intention of improving **maintainability**. … However, these techniques can give negative effects in **power consumption**. … But exceptions to them also exist. P. 720	Maintainability	Hinders	Mediated
perez2014	A better architecture in terms of maintainability can certainly be worse in terms of **power consumption**. P. 52	Maintainability	Hinders	Mediated
abdulsalam2015	"By improving the code's number of operations and the optimum usage of main memory and cache there is more room for energy savings." p. 2 "Defining the Greenup, Powerup and Speedup metrics to explain the correlations of energy, power and performance when optimizing software." p. 2	Performance efficiency	Positive	Directed
abdusalam2014	"For both C and C++++ code [of FFT], using optimization flags not only improves **performance,** but also reduces **energy consumption**." P. 3 Fast Fourier Transform, Linked List Insertion/Deletion and Quicksort.	Performance efficiency	Affects	Mediated

References

[1] M. Dick, S. Naumann, Enhancing software engineering processes towards sustainable software product design, Proceedings of EnviroInfo Conference, 2010, pp. 706–715.

[2] L.M. Hilty, P. Arnfalk, L. Erdmann, J. Goodman, M. Lehmann, P.A. Wäger, The relevance of information and communication technologies for environmental sustainability – a prospective simulation study, Environ. Model. Softw. 21 (2006) 1618–1629.

[3] United-Nation, W.C.o.E.D., Report of the world commission on environment and development: our common future, in United Nation conference on environmental

development. (1987).
[4] B. Penzenstadler, A. Raturi, D. Richardson, C. Calero, H. Femmer, X. Franch, Systematic mapping study on software engineering for sustainability (SE4S), Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (p. 14), ACM, 2014, pp. 1–10.
[5] E. Capra, C. Francalanci, S.A. Slaughter, Is software "green"? Application development environments and energy efficiency in open source applications, Inf. Softw. Technol. 54 (2012) 60–71.
[6] L. Ardito, G. Procaccianti, M. Torchiano, A. Vetrò, Understanding green software development: a conceptual framework, IT Prof. 17 (2015) 44–50.
[7] L. Ardito, M. Morisio, Green IT – available data and guidelines for reducing energy consumption in IT systems, Sustain. Comput.: Inform. Syst. 4 (2014) 24–32.
[8] B. Kitchenham, S. Charters, Ver. 2.3 EBSE Technical Report, Technical report EBSE, 2007.
[9] P. Lago, N. Meyer, M. Morisio, H.A. Muller, G. Scanniello, 2nd International workshop on green and sustainable software (GREENS 2013), Proceedings of International Conference on Software Engineering, 2013, pp. 1523–1524.
[10] B. Penzenstadler, A. Raturi, D. Richardson, B. Tomlinson, Safety, security, now sustainability: the nonfunctional requirement for the 21st century, IEEE Softw. 31 (2014) 40–47.
[11] L. Ardito, G. Procaccianti, A. Vetro, M. Morisio, Introducing energy efficiency into sqale, Proceedings of the Third International Conference on Smart Grids, Greeen Communictions and IT Energy-aware Technologies, Lisbon, Portugal, 2013, pp. 28–33.
[12] C. Calero, M.A. Moraga, M.F. Bertoa, L. Duboc, Quality in use and software greenability, Proceedings of CEUR Workshop, 2014, pp. 28–36.
[13] ISO/IEC-25010: Systems and Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Software Product Quality And System Quality in Use Model. (2010).
[14] W.N. Robinson, S.D. Pawlowski, V. Volkov, Requirements interaction management, ACM Comput. Surv. 35 (2003) 132–190.
[15] P. Lago, N. Meyer, M. Morisio, H.A. Müller, G. Scanniello, Leveraging "energy efficiency to software users": summary of the second GREENS workshop, at ICSE 2013, ACM SIGSOFT Softw. Eng. Notes 39 (2014) 36–38.
[16] A.B. Bener, M. Morisio, A. Miranskyy, Green software, IEEE Softw. 31 (2014) 36–39.
[17] C. Calero, M. Piattini, Introduction to green in software engineering, in: C. Calero, M. Piattini (Eds.), Green in Software Engineering, Springer, Cham, 2015, pp. 3–27.
[18] SustainAbility. Can Our Society Endure? Available from: http://sustainability.com/sustainability. (accessed 29 September 2017).
[19] L.M. Hilty, B. Aebischer, ICT for sustainability: an emerging research field, in: L. Hilty, B. Aebischer (Eds.), ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing, vol. 310, Springer, Cham, 2015, pp. 3–36.
[20] P. Lago, S.A. Koçak, I. Crnkovic, B. Penzenstadler, Framing sustainability as a property of software quality, Commun. ACM 58 (2015) 70–78.
[21] S. Naumann, M. Dick, E. Kern, T. Johann, The GREENSOFT model: a reference model for green and sustainable software and its engineering, Sustain. Comput.: Inform. Syst. 1 (2011) 294–304.
[22] C. Calero, M.F. Bertoa, M.A. Moraga, A systematic literature review for software sustainability measures, Proceedings of 2013 2nd International Workshop on Green and Sustainable Software, GREENS 2013, 2013, pp. 46–53.
[23] B. Penzenstadler, H. Femmer, A generic model for sustainability with process- and product-specific instances, Proceedings of the 2013 Workshop on Green in Software Engineering, Green by Software Engineering, GIBSE 2013, 2013, pp. 3–7.
[24] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, C.C. Venters, Sustainability design and software: the Karlskrona manifesto, Proceedings of International Conference on Software Engineering, 2015, pp. 467–476.
[25] A. Noureddine, A. Bourdon, R. Rouvoy, L. Seinturier, A preliminary study of the impact of software engineering on GreenIT, Proceedings of the 1st International Workshop on Green and Sustainable Software, GREENS 2012, 2012, pp. 21–27.
[26] K. Erdélyi, Special factors of development of green software supporting eco sustainability, Proceedings of IEEE 11th International Symposium on Intelligent Systems and Informatics, SISY 2013, 2013, pp. 337–340.
[27] J. Taina, Good, bad, and beautiful software – in search of green software quality factors, CEPIS UPGRADE 11 (2011) 22–27.
[28] B. Penzenstadler, V. Bauer, C. Calero, X. Franch, Sustainability in software engineering: a systematic literature review, Proceedings of IET Seminar Digest, 2012, pp. 32–41.
[29] P. Bozzelli, Q. Gu, P. Lago, A Systematic Literature Review on Green Software Metrics, VU University, Amsterdam, 2013.
[30] P. Bourque, R.E. Fairley, Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2014 Version 3.0.
[31] G.A. García-Mireles, M.A.M. De La Rubia, F. García, M. Piattini, Methods for supporting management of interactions between quality characteristics, Proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2014, 2014, pp. 93–100.
[32] S. Barney, K. Petersen, M. Svahnberg, A. Aurum, H. Barney, Software quality trade-offs: a systematic map, Inform. Softw. Technol. 54 (2012) 651–662.
[33] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requir. Eng. 11 (2006) 102–107.
[34] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: an update, Inf. Softw. Technol. 64 (2015) 1–18.
[35] G.A. García-Mireles, M.Á. Moraga, F. García, M. Piattini, Approaches to promote product quality within software process improvement initiatives: a mapping study, J. Syst. Softw. 103 (2015) 150–166.
[36] P. Berander, L.-O. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson, S. Kågström, D. Milicic, F. Mårtensson, K. Rönkkö, Software Quality Attributes and Trade-offs, Blekinge Institute of Technology, 2005.
[37] B. Kitchenham, S.L. Pfleeger, Software quality: the elusive target [special issues section], IEEE Softw. 13 (1996) 12–21.
[38] M. Unterkalmsteiner, T. Gorschek, A.K.M.M. Islam, C.K. Cheng, R.B. Permadi, R. Feldt, Evaluation and measurement of software process improvement: a systematic literature review, IEEE Trans. Softw. Eng. 38 (2012) 398–424.
[39] ISO/IEC: Systems and Software Engineering – Software Life Cycle Processes – Redline. ISO/IEC 12207:2008(E) IEEE Std 12207-2008 – Redline1-195 (2008).
[40] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (2008) 833–859.
[41] K.S. Vallerio, L. Zhong, N.K. Jha, Energy-efficient graphical user interface design, IEEE Trans. Mob. Comput. 5 (2006) 846–859.
[42] A. Kansal, F. Zhao, Fine-grained energy profiling for power-aware application design, SIGMETRICS Perform. Eval. Rev. 36 (2008) 26–31.
[43] J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, J.N. Mazón, Integrating sustainability in decision-making processes: a modelling strategy, Proceedings of 2009 31st International Conference on Software Engineering – Companion Volume, ICSE 2009, 2009, pp. 207–210.
[44] C. Calero, M. Piattini, Green in Software Engineering, Springer International Publishing, 2015, pp. 1–327.
[45] S.A. Koçak, G.I. Alptekin, A.B. Bener, Integrating environmental sustainability in software product quality, Proceedings of CEUR Workshop, 2015, pp. 17–24.
[46] I. Moura, G. Pinto, F. Ebert, F. Castor, Mining energy-aware commits, Proceedings of IEEE International Working Conference on Mining Software Repositories, 2015, pp. 56–67.
[47] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, J. Clause, An empirical study of practitioners' perspectives on green software engineering, Proceedings of the 38th International Conference on Software Engineering, ACM, Austin, Texas, 2016, pp. 237–248.
[48] S.A. Chowdhury, A. Hindle, Characterizing energy-aware software projects: are they different? Proceedings of 13th Working Conference on Mining Software Repositories, MSR 2016, 2016, pp. 508–511.
[49] O. Gordieiev, V. Kharchenko, M. Fusani, Evolution of software quality models: green and reliability issues, Proceedings of CEUR Workshop, 2015, pp. 432–445.
[50] M.Á. Moraga, M.F. Bertoa, Green software measurement, in: C. Calero, M. Piattini (Eds.), Green in Software Engineering, Springer, Cham, 2015, pp. 261–282.
[51] V. Cordero, I.G.R. De Guzmán, M. Piattini, A first approach on legacy system energy consumption measurement, Proceedings of the 2015 IEEE 10th International Conference on Global Software Engineering Workshops, ICGSEW 2015, 2015, pp. 35–43.
[52] M. Couto, T. Carcão, J. Cunha, J.P. Fernandes, J. Saraiva, Detecting anomalous energy consumption in android applications, in: F.M. Quintão Pereira (Ed.), Programming Languages LNCS, vol. 8771, Springer, Cham, 2014, pp. 77–91 SBLP 2014.
[53] H. Field, G. Anderson, K. Eder, EACOF: a framework for providing energy transparency to enable energy-aware software development, Proceedings of the ACM Symposium on Applied Computing, 2014, pp. 1194–1199.
[54] C. Zhang, A. Hindle, A green miner's dataset: mining the impact of software change on energy consumption, Proceedings of the 11th Working Conference on Mining Software Repositories, ACM, Hyderabad, India, 2014, pp. 400–403.
[55] A. Noureddine, S. Islam, R. Bashroush, Jolinar: Analysing the energy footprint of software applications (Demo), Proceedings of the 25th International Symposium on Software Testing and Analysis, ISSTA 2016, 2016, pp. 445–448.
[56] S.J. Chinenyeze, X. Liu, A. Al-Dubai, DEEPC: dynamic energy profiling of components, Proceedings of the International Computer Software and Applications Conference, 2016, pp. 186–191.
[57] C. Kin Keong, K. Tieng Wei, A.A. Abd Ghani, K.Y. Sharif, Toward using software metrics as indicator to measure power consumption of mobile application: a case study, Proceedings of 2015 9th Malaysian Software Engineering Conference, MySEC 2015, 2015, pp. 172–177.
[58] Å.G. Dahlstedt, A. Persson, Requirements interdependencies: state of the art and future challenges, in: A. Aurum, C. Wohlin (Eds.), Engineering and Managing Software Requirements, Springer, Berlin, Heidelberg, 2005, pp. 95–116.
[59] A. Van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley Publishing, 2009.
[60] H. Zhang, J. Li, L. Zhu, R. Jeffery, Y. Liu, Q. Wang, M. Li, Investigating dependencies in software requirements for change propagation analysis, Inf. Softw. Technol. 56 (2014) 40–53.
[61] A. Salado, R. Nilchiani, The concept of order of conflict in requirements engineering, IEEE Syst. J. 10 (2016) 25–35.
[62] G. Lami, L. Buglione, Measuring software sustainability from a process-centric perspective, Proceedings of the 2012 Seventh International Conference on Software Process and Product Measurement (IWSM-MENSURA) and 2012 Joint Conference of the 22nd International Workshop on Software Measurement, IEEE, 2012, pp. 53–59.
[63] I. Manotas, L. Pollock, J. Clause, SEEDS: a software engineer's energy-optimization decision support framework, Proceedings of the 36th International Conference on Software Engineering, ACM, Hyderabad, India, 2014, pp. 503–514.
[64] R. Perez-Castillo, M. Piattini, Analyzing the harmful effect of god class refactoring on power consumption, IEEE Softw. 31 (2014) 48–54.
[65] C. Sahin, L. Pollock, J. Clause, How do code refactorings affect energy usage? Proceedings of International Symposium on Empirical Software Engineering and

Measurement, 2014, pp. 1–10.
[66] J.J. Park, J.E. Hong, S.H. Lee, Investigation for software power consumption of code refactoring techniques, Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE, 2014, pp. 717–722.
[67] M. Gottschalk, J. Jelschen, A. Winter, Refactorings for energy-efficiency, in: J.M. Gomez, M. Sonnenschein, U. Vogel, A. Winter, B. Rapp, N. Giesen (Eds.), Advances and New Trends in Environmental and Energy Informatics, Springer, Cham, 2016, pp. 77–96.
[68] E.A. Jagroep, J.M.v.d. Werf, S. Brinkkemper, G. Procaccianti, P. Lago, L. Blom, R.v. Vliet, Software energy profiling: comparing releases of a software product, Proceedings of the 38th International Conference on Software Engineering Companion, ACM, Austin, Texas, 2016, pp. 523–532.
[69] I. García-Rodríguez De Guzmán, M. Piattini, R. Pérez-Castillo, Green Software Maintenance, in: C. Calero, M. Piattini (Eds.), Green in Software Engineering, Springer, Cham, 2015, pp. 205–229.
[70] J. Jelschen, M. Gottschalk, M. Josefiok, C. Pitu, A. Winter, Towards applying reengineering services to energy-efficient applications, in: T. Mens, A. Cleve, R. Ferenc (Eds.), Proceedings of 2012 16th European Conference on Software Maintenance and Reengineering, 2012, pp. 353–358.
[71] S. Abdulsalam, D. Lakomski, Q. Gu, T. Jin, Z. Zong, Program energy efficiency: the impact of language, compiler and implementation choices, Proceedings of the 2014 International Green Computing Conference (IGCC), 2014, pp. 1–6.
[72] G. Procaccianti, H. Fernández, P. Lago, Empirical evaluation of two best practices for energy-efficient software development, J. Syst. Softw. 117 (2016) 185–198.
[73] M. Rashid, L. Ardito, M. Torchiano, Energy consumption analysis of algorithms implementations, Proceedings of International Symposium on Empirical Software Engineering and Measurement, 2015, pp. 82–85.
[74] L. Corral, A.B. Georgiev, A. Sillitti, G. Succi, Method reallocation to reduce energy consumption: an implementation in Android OS, Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, Gyeongju, Republic of Korea, 2014, pp. 1213–1218.
[75] D. Li, W.G.J. Halfond, An investigation into energy-saving programming practices for android smartphone app development, Proceedings of 3rd International Workshop on Green and Sustainable Software, GREENS 2014, 2014, pp. 46–53.
[76] G. Agosta, M. Bessi, E. Capra, C. Francalanci, Automatic memoization for energy efficiency in financial applications, Sustain. Comput. – Inform. Syst. 2 (2012) 105–115.
[77] A. Noureddine, A. Rajan, Optimising energy consumption of design patterns, Proceedings of 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015, pp. 623–626.
[78] C. Sahin, F. Cayci, I.L.M. Gutiérrez, J. Clause, F. Kiamilev, L. Pollock, K. Winbladh, Initial explorations on design pattern energy usage, Proceedings of 2012 1st International Workshop on Green and Sustainable Software, GREENS 2012, 2012, pp. 55–61.
[79] I. Manotas, C. Sahin, J. Clause, L. Pollock, K. Winbladh, Investigating the impacts of web servers on web application energy usage, Proceedings of 2013 2nd International Workshop on Green and Sustainable Software, GREENS 2013, 2013, pp. 16–23.
[80] G. Procaccianti, P. Lago, W. Diesveld, Energy efficiency of ORM approaches: an empirical evaluation, Proceedings of International Symposium on Empirical Software Engineering and Measurement, 2016, pp. 1–10.
[81] J. Yu, H. Han, H. Zhu, Y. Chen, J. Yang, Y. Zhu, G. Xue, M. Li, Sensing human-screen interaction for energy-efficient frame rate adaptation on smartphones, IEEE Trans. Mob. Comput. 14 (2015) 1698–1711.
[82] A. Rajan, A. Noureddine, P. Stratis, A study on the influence of software and hardware features on program energy, Proceedings of International Symposium on Empirical Software Engineering and Measurement, 2016, pp. 1–10.
[83] M. Sabharwal, A. Agrawal, G. Metri, Enabling green IT through energy-aware software, IT Prof. 15 (2013) 19–27.
[84] B. Penzenstadler, From Requirements Engineering to Green Requirements Engineering, in: C. Calero, M. Piattini (Eds.), Green in Software Engineering, Springer, Cham, 2015, pp. 157–186.
[85] B. Penzenstadler, J. Mehrabi, D.J. Richardson, Supporting physicians by RE4S: evaluating requirements engineering for sustainability in the medical domain, Proceedings of the 4th International Workshop on Green and Sustainable Software, GREENS 2015, 2015, pp. 36–42.
[86] C. Bomfim, W. Nunes, L. Duboc, M. Schots, Modelling sustainability in a procurement system: an experience report, Proceedings of 2014 IEEE 22nd International Requirements Engineering Conference, RE 2014, 2014, pp. 402–411.
[87] A. Rodriguez, B. Penzenstadler, An assessment technique for sustainability: applying the IMAGINE approach to software systems, Proceedings of CEUR Workshop Proceedings, 2013, pp. 1–8.
[88] M. Mahaux, P. Heymans, G. Saval, Discovering sustainability requirements: an experience report, in: D. Berry, X. Franch (Eds.), Requirements Engineering: Foundation for Software Quality, REFSQ 2011, Springer, Berlin, Heidelberg, 2011, pp. 19–33 6606 LNCS.
[89] A. Raturi, B. Penzenstadler, B. Tomlinson, D. Richardson, Developing a sustainability non-functional requirements framework, Proceedings of 3rd International Workshop on Green and Sustainable Software, GREENS 2014, 2014, pp. 1–8.
[90] A. Marcus, J. Dumpert, L. Wigham, User-experience for Personal Sustainability Software: Determining Design Philosophy and Principles, Springer, Berlin, Heidelberg, 2011, pp. 172–177 Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).
[91] A. Marcus, J. Dumpert, L. Wigham, User-experience for Personal Sustainability Software: Applying Design Philosophy and Principles, Springer, Cham, 2014, pp. 583–593 Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).
[92] C. Zhang, A. Hindle, D.M. German, The impact of user choice on energy consumption, IEEE Softw. 31 (2014) 69–75.
[93] B. Penzenstadler, A Toolkit for SE for Sustainability – A Design Fiction, in: A. Marcus (Ed.), Design, User Experience, and Usability: Design Discourse. DUXU 2015. Lecture Notes in Computer Science, vol. 9186, Springer, Cham, 2015, pp. 634–643.
[94] K. Roher, D. Richardson, A proposed recommender system for eliciting software sustainability requirements, Proceedings of 2013 2nd International Workshop on User Evaluations for Software Engineering Researchers, USER 2013, 2013, pp. 16–19.
[95] T.R.D. Saputri, S.W. Lee, Incorporating sustainability design in requirements engineering process: a preliminary study, in: S. Lee, T. Nakatani (Eds.), Requirements Engineering Toward Sustainable World. APRES 2016, Springer, Singapore, 2016, pp. 53–67 671 CCIS.
[96] M.A. Beghoura, A. Boubetra, A. Boukerram, Green software requirements and measurement: random decision forests-based software energy consumption profiling, Requir. Eng. (2015), http://dx.doi.org/10.1007/s00766-015-0234-2.
[97] S. Ouhbi, J.L. Fernández-Alemán, A. Idri, J.R. Pozo, Are mobile blood donation applications green? Proceedings of 2015 10th International Conference on Intelligent Systems: Theories and Applications, SITA 2015, 2015, pp. 1–6.
[98] S. Abdulsalam, Z. Zong, Q. Gu, M. Qiu, Using the greenup, powerup, and speedup metrics to evaluate software energy efficiency, Proceedings of 2015 6th International Green and Sustainable Computing Conference, 2015, pp. 1–8.
[99] S.D. Ambrosio, S.D. Pasquale, G. Iannone, D. Malandrino, A. Negro, G. Patimo, A. Petta, V. Scarano, L. Serra, R. Spinelli, Mobile phone batteries draining: Is green web browsing the solution? Proceedings of 2014 International Green Computing Conference (IGCC), 2014, pp. 1–10.
[100] Chang, S.-W., Cheng, S.-W., Hsiu, P.-C., Kuo, T.-W., Lin, C.-W., Application behavior analysis in resource consumption for mobile devices, in Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM: Gyeongju, Republic of Korea. p. 1469–1474.
[101] F. Ahmed, H. Mahmood, A. Aslam, Green computing and software defects in open source software: an empirical study, Proceedings of 2014 International Conference on Open Source Systems and Technologies, ICOSST 2014, 2014, pp. 65–69.
[102] Aggarwal, K., Zhang, C., Campbell, J.C., Hindle, A., Stroulia, E., The power of system call traces: predicting the software energy consumption impact of changes, in Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, IBM Corp.: Markham, Ontario, Canada. p. 219–233.
[103] S.A. Koçak, G.I. Alptekin, A.B. Bener, Evaluation of software product quality attributes and environmental attributes using ANP decision framework, Proceedings of CEUR Workshop Proceedings, 2014, pp. 37–44.
[104] F. Rahman, C. O'Brien, S.I. Ahamed, H. Zhang, L. Liu, Design and implementation of an open framework for ubiquitous carbon footprint calculator applications, Sustain. Comput.: Inform. Syst. 1 (2011) 257–274.
[105] R. Amri, N. Bellamine Ben Saoud, Towards a generic sustainable software model, Proceedings of the 2014 4th International Conference on Advances in Computing and Communications, ICACC 2014, Institute of Electrical and Electronics Engineers Inc., 2014, pp. 231–234.
[106] J. Iivari, Why are CASE tools not used? Commun. ACM 39 (1996) 94–103.
[107] S. Naumann, E. Kern, M. Dick, T. Johann, Sustainable Software Engineering: Process and Quality Models, Life Cycle, and Social Aspects, in: L. Hilty, B. Aesbischer (Eds.), ICT Innovations for Sustainability. Advances in Intelligent Systems and Computing, vol. 310, Springer, Cham, 2015, pp. 91–205.
[108] K. Sierszecki, T. Mikkonen, M. Steffens, T. Fogdal, J. Savolainen, Green software: greening what and how much? IEEE Softw. 31 (3) (2014) 64–68.
[109] B.A. Kitchenham, D. Budgen, O. Pearl Brereton, Using mapping studies as the basis for further research – a participant-observer case study, Inf. Softw. Technol. 53 (2011) 638–651.
[110] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Inf. Softw. Technol. 55 (2013) 2049–2075.