Proceedings of the Pakistan Academy of Sciences:
A. Physical and Computational Sciences 54 (1): 71–87 (2017)

ISSN: 2518-4245 (print), 2518-4253 (online)

Pakistan Academy of Sciences

Research Article

Risks Mitigation Practices for Multi-Sourcing Vendors in Green Software Development

Muhammad Salam*, and Siffat Ullah Khan

Department of Computer Science and Information Technology,
University of Malakand, Dir (L), Pakistan

Abstract: Green and sustainable software development is the cry of the day and vendors are constantly striving to develop such software that have less hazardous impact on environment, economy and human beings. However developing green software in the context of software multi-sourcing is not a risk free activity. Software development multi-sourcing vendor organizations have focused on the adaptation of green practices in software development projects. In our previous study we have identified eight critical risk factors (CRFs) via systematic literature review (SLR) process, in the development of green and sustainable software. These CRFs are: 'lack of green RE practices', 'high power consumption', 'high carbon emission throughout the software development', 'poor software design (architectural, logical, physical and user interface)', 'lack of ICTs for coordination and communication', 'high resources requirements', 'lack of coding standards', and 'lack of green software development knowledge'. The proactive management of the identified risks might allow software development multi-sourcing vendor organizations to develop green and sustainable software successfully. In this study we have presented the identified 76 practices for addressing the aforementioned eight critical risk factors. The practices were extracted from sample of (N=102) research papers via SLR process. We have validated the identified 76 solutions/practices from 108 relevant experts in software development multi-sourcing industry via questionnaire survey. The findings of this study may help vendor organizations to address/mitigate the CRFs using the identified solutions in order to develop green and sustainable software in multi-sourced software projects.

Keywords: Green software multi-sourcing, risk mitigation, solutions/practices, systematic literature review (SLR), industrial survey

1. INRODUCATION

Green and sustainable software has been defined in the literature [1, 2] as "the software, whose direct and indirect negative impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software are minimal and/or which has a positive effect on sustainable development". Where Green and Sustainable Software Engineering is the art of developing green and sustainable software engineering process [3].

Currently much work has been done to obtain green and sustainable software in general [2, 4-8]. The software which has a longer life time is considered sustainable software. According to [9] the term sustainable applies to both longer life and greener aspects of software. Our aim in this study is specifically focused on the development of green and sustainable software in multi-sourced software projects, which is explained in the subsequent paragraphs.

In order to transfer the general concept of sustainability into the computer systems (hardware as well as software) the term 'green computing' or 'sustainable computing' or 'green IT' is used [10, 11]. Green computing can be defined as the practice of maximizing the efficient use of computing resources to minimize negative impact on environment [12, 13]. In other words, green computing or green IT is the study and practice of designing, manufacturing, using and disposing of computing resources efficiently and effectively with minimal environment damages [14].

Lo and Qian [15] defined green computing as "environmentally sustainable computing which studies and practices virtually all computing

Received, June 2016; Accepted, March 2017
*Corresponding author: Muhammad Salam; Email: msmdir@gmail.com

efficiently and effectively with little or no impact on the environment". However, Tushi and Bonny [16] defined green computing as " the practice of implementing policies and procedures that improve the efficiency of computing resources in such a way as to reduce the energy consumption and environmental impact of their utilization". The goal of green IT is to yield as less possible waste throughout the lifecycle of IT including (development, operation and disposal) [17].

The best use of information and communication technologies (ICTs) is to manage enterprise activities in eco-friendly manner comprising its product, services and resources throughout their life. The principal objective of green IT approach is to minimize the energy consumption, uphold the operational costs and minimize environmental impacts [16]. Though, it is noteworthy that there are two aspects of Green IT, primarily, IT can be the reason of ecological problems, and otherwise it can be used to resolve ecological problems [16, 18]. Literature [19] reveals that the term 'green IT' and 'green computing' are the same. Infrastructure plays an important role in the development of software in eco-friendly manner i.e. use meeting rooms with natural lights, avoid the use of air-conditioners, minimize the traveling and use modern ICT tools, establish paperless offices, use the concept of cloud computing in the software development houses [6].

Till recently, the larger portion of the efforts done in the era of 'Green IT' were linked to hardware, concentrating mostly on improving the hardware energy efficiency. Thus it is obvious that research work needs to focus on the software as well within green IT [10, 17]. The tendency has been changed in the last few years, and research on the new theme of green software is emerging. In this study we have focused on the developmnt of green software in multi-sourced software projects.We have identified practices/solutions for addressing the crtical risk factorsin order to develop green and sustainable software in multi-sourced projects.

Software outsourcing is the allocation of software processes to external (offshore) professionals in order to reduce cost, improve quality, and minimize the development time [20, 21]. There are three components of outsourcing i.e. client, vendor, and the project itself [22]. The organization outsourcing the software processses is referred to as the client, the organization that develops the software and makes decisions is called vendor, and the scope of the software development work is captured in a project.Multi-sourcing is a modern paradigm in outsourcing which offers the benefits of using multiple vendors for the development of software in a shorter time span. In multi-sourcing, client(s) outsource their software development work to multiple vendors. Software development multi-sourcing is a modern global software engineering paradigm for developing high quality software at minimum cost and time in low wages countries by contracting out the software development work to multiple vendors [23, 24].

Green methods and practices are getting prominence in software development multi-sourcing as well [25]. According to [26] software development is a perilous process and is vulnerable to risks from the initial phase till final stage. A number of researchers have worked on the identification and management of risks in the software development in general [27-31]. A risk has been defined in the literature as "a risk is a potential future harm that may arise from some present action" OR "Risk represents an undesirable event or a negative outcome to the expected result" [32]. Regardless of the significance and importance of green software development, little empirical research has been conducted on the identification and management of risk factors in the development of green and sustainable software. This study focuses on discovering solutions/practices for addressing eight critical risk factors that affect green and sustainable software development in multisourced projects.

2. BACKGROUD OF STUDY

Penzenstadler et al. [33] conducted a systematic mapping study to deliver an overview of the present knowledge on sustainability and software engineering. The authors of the study pointed out that the topic of software engineering for sustainability has acknowledged wide spread attention in the community of software engineering. Owing to the fact of being a comparatively new research area, little empirical research is available in practice on software engineering and sustainability. The authors have also highlighted that solutions/practices for green

software development are still immature. Penzenstadler's findings propose that more empirical results are needed on software engineering for sustainability.

Becker et al. [34] have presented Karlskrona Manifesto for sustainability design. The study combines and enlarges the present understanding of sustainability concerns inside the software engineering (SE) community. The manifesto reveals and levels out a number of communal misinterpretations with respect to SE and sustainability. The values of sustainability design pretence new challenges to research on sustainability and software engineering.

In the study conducted by Weyns et al. [35] pointed out that integrating runtime adaptation and evolution is vital for the sustainability of software systems. This approach encompasses two complementaries. The first component is AdEpS model that defines the two combined processes to handle change, concerning doubts: adaptation management to preserve goals and evolution management to deal with goal changes. The second component is: three main engineering standard to design software systems that follows the AdEps model: design for meta-variability and inconsistency, examining, and controlled change. For each standard, the authors have showed new ideas for understanding the level of constituent models and languages.

Raturi et al. [36] have presented a Sustainability Non-Functional Requirements (SNFR) framework. The proposed framework can assist the software requirements engineers to classify and elicit sustainability requirements for the system to be made. The authors outline a roadmap which help the software requirement engineers to implement the theoretical NFR framework as quality factors.

Kern et al. [37] have focused on the green and sustainable software engineering process.The authors advised to have a close look at the software products life cycle. The life cycle contains three portions: (i) distribution and development of the software products; (ii) the usage phase (iii) disposal and deactivation of the product. In order to create green and sustaianble software, the developer should focus on the processes during the software development life cycle. Further, the authors have presented a model (process model) for the orgnization and development of green and sustainable software. The authors recommended that sustainability aspects should be considered in software engineering processes

Bartenstein et al. [38] have introduced 'GREEN STREAMS' in order to address energy efficiency of data-intensive software. GREEN STREAMS deliver an effective and practical approch to save the energy of data-intensive software. The proposed programming model/paradigm delivers appropriate language abstractions for creating data-intensive software, it also provides the perfect structure for efficient energy management. As a future work, the authors have planned to spread GREEN STREAMS to upkeep dynamic flexibility.

Easterbrook [39] presented the role of software researchers and practitioners in the climate change. According to the author, climate change is an importnat and urgent issue, and there is need of mobilization and efforts in many disciplines to address this problem. The author has identified three main areas where work can be done: software for understanding climate change; software for supporting the global joint decision making; and software that helps in reducing the green house gas of current technology.

Moraga et al. [40] have focused on studying measurement within the context of green software. Nowadays individuals have such a standard of life that puts the future generations resource at risk. However, there is a rising awareness of this issue in the civil societies. In the domain of software engineering, one of the core drives for assessing (measuring) has risen owing to the increasing interests in this theme. The main objective of the measurement is to improve the project, process or the product itself. In this study the authors have focused on the features allied to the product. The authors have considered a set of 192 measures recommended by different authors and have chosen 74 measures relevent to the product greenability (software greenability). They have argued that these 74 measures can be classified as regards the greenability.

Naumann et al. [41] have focused on the meaning of sustainable software and sustainable software engineering. Moreover, the authors propose a model of sustainable software as well as sustainable software engineering. Though, the study delivers only a short overview of the model.

The authors argued that software plays a key role in the ICT sustainability, that is why, the authros have considered particularly how to make software engineering process and software product itself more sustainable.

Lago [42] presented opportunities and challenges for sustainable software development. According to the author software play a key role in supporting our society. Consequently, environmental sustainability has become a major factor in the operation and development of software system. The author pointed out key areas regarding green and sustaianble software development. These includes: software energy efficiency, green software, measuring software energy consumption in practice, sustainable architectural design, environmental-friendly and cloud ready software.

Arakelyan et al. [43] have suggested design choices that possibly improve appropriation of the software and allows for the sequence to movement efficiently from one phase to the other. The authors intends to promote and improve the proposed design choices as a future work by performing a more detailed literature review along with expert ratings.

A study was conducted by Beghoura et al. [44] on measurement of the green software requirements. The study recommends a clear definition of green software requirements. The authors proposed an approach to launch an energy profiling tool to find the energy consuming lots of code. The recommended assessment tool defines the green efficiency by seeing the energy consumption as the key feature to be considered throughout the development phase.

Ignacio et al. [45] have worked and emphasized on green software maintenance and have attempted to predict a definition and possible practices for green software maintenance.

Betz et al. [46] have worked on sustainable software system engineering. They argue that sustainability management is one of the key issue of present that is why public and private administrations are keen interested in the “sustainable” practices and solutions. The study further pointed out that, there is a dearth of existing practices and solutions for sustainable development. For this purpose the authors proposed a conceptual model in order to incorporate sustainability features in a business development. Moreover, the study proposed that, to incorporate sustainability traits in the domain of software engineering, sustainability requirements should be measured throughout the software development life cycle.

Hayri et al. [47] have worked on the energy measurement of software at runtime and identified that ICTs are liable for nearby 2% of the global greenhouse gas productions. Further, the usage of mobile devices is repeatedly increasing. Because of the Internet and the cloud computing, consumers are using ever more software applications which producing more greenhouse gas. Therefore, a significant question is "in what ways can we decrease or limit the energy intake connected to ICTs and, in specific, connected to software?". Most of the suggested solutions focused only on the hardware aspect, though in recent years the software facets have also become significant.

Li et al. [48] have worked on green software from the business requirements point of view. The authors have pointed out that research on the new theme of green software is still at nascent stage. Initial research issues, problems, and methodological practices have been suggested; however widespread acceptance of green software is not yet fully implemented.

Rahma et al. [49] have focused on the development of a generic sustainable software model. The authors argue that sustainability is becoming an interesting topic in the domain of software engineering. In order to cover the different dimensions of sustainability, the authors projected a Generic Sustainable Software Star Model (GS3M). The proposed model covers a “complete” outlook of sustainable software. The different dimensions that the proposed model covers are: social, individual, environmental, economic, and technical. The authors have defined corresponding values for each sustainability dimension. The proposed model can be used to assess software projects sustainability.

Penzenstadler et al. [50] worked on green requirements engineering. The authors argue that ecological sustainability can be implemented to software systems in two ways, either as green through software systems green in software systems. The study demonstrates a checklist based method that determines how to incorporate the aim of ecological sustainability from the very first stages.

The explanation is exemplified by a case study on a sharing car system.

Kim et al. [51] worked on architectural sustainability with respect to non-functional requirements and have discussed that sustainability of software designs has gained more consideration to deal with the factors affecting architectural modifications (changes), for example, requirements modifications, technical changes, and modifications in business approaches and objectives. However, it is argued that there is a limited work done on architectural sustainability. In their study, the authors presented a new method for dealing with architectural sustainability (with respect to non-functional requirements changes) through defensive architectural designs erected upon the joined use of architectural designs and architectural strategies.

Jetley et al. [52] have pointed that software industry needs to integrate the premium practices of software engineering in their software application development procedure to optimize quality and cost. Though, this needs the factual set of methodologies and tools that satisfies the requirements of the software industry. Whereas, there have been a limited up-to-date software engineering methodologies, tools and techniques. The authors of tha aforementioned study have highlighted some challenges faced by the software industry while implementing software engineering methodologies/practices in the application development. Moreover, the study emphasized that there is a need for further research and efforts which support the adaptation of software engineering and methodologies and principles in the software industry.

Dick et al. [53] have worked on the green software engineering with agile methods. The authors have proposed a model that mixes green computing features into software engineering procedures with agile methods to deliver green and sustainable software.

Ardito et al. [54] have conducted a survey on presented guidelines and data for reducing energy intake of the information system i.e. the authors have provided various energy efficiency guidelines including: infrastructure, application, operating system, hardware, and network.

Lami et al. [55] have worked on sustainability from a software process viewpoint. The authors argued that ICTs considerably contributes to the production of global carbon dioxide. The same researchers have discussed this problem from different perspectives. In this way, they have addressed the software sustainability from a process centric perspective. For this purpose they defined set of procedures that denote the activities/actions to be executed to introduce and incorporate the culture of green software development in the organization.

Yuzhong et al. [56] have explored the challenges to software (system software) in data centers. The authors have summarized certain tendencies that affect the data centers efficiency. Moreover, they investigated the reasons of inefficiency of the software system. They discussed and presented the four key challenges of building energy efficient software system: (a) programming difficulty (b) extreme scalability (c) energy efficiency of the software (d) adaptation to soft architecture. The authors have also recommended some basic practices for addressing the identified aforementioned challenges.

The aforementioned studies have described a number of issues in green software development in general context. However there is a lack of well-defined solutions for addressing the critical risks in developing green and sustainable software in the context of multi-sourncing. In this paper,we have reported the identification of state-of-the-art practices/solutions for addressing eight critical risk factors in the development of green and sustainable software in multi-sourced projects.

3. RESEARCH METHODLOGY

We have used two research methodologies i.e. Systematic Literature Review (SLR) and Industrial Survey. For identification of solutions/ practices, we followed the systematic literature review guidelines recommended by Kitchenham [57]. Consequently, we presented the core phases of our review protocol i.e. planning, conducting, and reporting, whereas industrial survey has been conducted, in software multi-sourcing industry, for validation of the SLR findings and to find any new solution/practice apart from the SLR findings, if any.

3.1 Planning the Review

3.1.1. Research Questions and Research Objectives

The core objective of this resarch study is to find out state-of-the-art practices for addressing critical risk factors in the development of green and

sustaianble software in multi-sourecd software projects. To achive this goal, we outlined the following research questions (RQs):

RQ1: What are the practices/solutions (as identified in the literature) for mitigating the identified risk factors in the development of green and sustainable software?

RQ2: What are the practices/solutions (as identified in in real-world practice), for mitigating the identified risk factors in the development of green and sustainable software?

3.1.2. Search Strategy

To carry out this study we followed the procedures provided by Kitchenham [57, 58]. After the finalization of research objectives and research question, we defined a comprehensive search strategy to examine possible available empirical studiesaccording to the aims of this systematic review. We also finalized the online search venues for the execution of our search string. The list of online digital libraries is presented as follow:

(a) Science Direct http://www.sciencedirect.com/
(b) ACM http://dl.acm.org/
(c) IEEE Xplore http://ieeexplore.ieee.org/
(d) Springer Link http://link.springer.com/
(e) Google Scholar https://scholar.google.com.pk/

3.1.3. Search String

We have designed the following two search strings for searching our selected online digital libraries.We derived the search strings from our formulated research question presented in section 3.1.1.

λ1:("Green software" OR "sustainable software") AND ("practices" OR "solutions" OR "techniques") AND ("multi-sourcing")

λ2: Green software" OR "sustainable software") AND ("practices" OR "solutions" OR "techniques")

Where λ1denotes search stringto retrieve empirical studies regarding the practices for the development of green software multi-sourced software projects, while λ2 denotes search string to retrieve empirical studies regarding the practices for the development of green software in general context. The results of search string (λ1) were very poor and almost negligible. Consequently we decided to implement search string λ2 after detaileddiscussions with experts of the software engineering research group (SERG-UOM) at the university, to implement search string λ2. The search results of λ2 are showed in Table 1. The practices, identified through the SLR (using search string λ2), will be validated through empirical studies in multi-sourcing software industry to know whether these findings are applicable specifically, or can be adopted, in software multi-sourcing environment. The same approach for verifying the SLR findings via empirical study has been used by other researchers as well [59]. Moreover,limited numbers of empirical research studies have been conducted in the context of global software development in general and software multi-sourcing in particular [60].

3.2 Conducting the Review

In this section, we have presented the outcomes of the implementation of our finalized search string (λ2) retrieved form the selected digital libraries. The selected online venues were searched using search string (λ2) and considerable amount of studies were retrieved. The search results are presented in Table 1.

3.2.1. Study Selection

In the first phase of papers selection we selected papers on the basis of titles and abstracts that were relevant to our research question.The included and excluded papers in the first phase are shown in Table 1. In the second phase of publication selection we studied the full text of the primary selected papers and excluded irrelevant papers from the primary list. As a result, we got 44 relevant papers. Finally we merged the papers of previous SLR [61]with finally selected 44 papers and got a list of (N=102) papers.

3.2.2. Data Extraction

We followed the guidelines provided by Kitchenham [57] and successfully extracted 76 pratices/solutions for identified 08 risk factors from (N=102) research papers.

3.2.3. Data Synthesis

At the end of the data extraction phase we got a list of 161solutions/practices initially. After detailed analysis of the identified 161 practices we classified 76 practices for critical risk factors from the sample of 102 papers. These identified 76

Table 1. Search string (λ2) results.

S. No.	Data sources	Retrieved	Phase 1		Phase 2	
			Included	Excluded	Included	Excluded
1	Google Scholar	429	47	382	19	28
2	ACM	164	29	135	12	17
3	IEEE Xplore	114	20	94	04	16
4	Springer Link	149	23	126	06	17
5	Science Direct	16	04	12	03	01
6	Total	872	123	749	44	79

Table 2. Summary of software development companies and multi-sourcing professionals groups.

S. No.	Name of Software Development companies/IT Board	Date of request
1	Pakistan Software Export Board	November 2015
2	Khyber Pakhtunkhwa Information Technology Board	November 2015
3	Punjab IT Board	November 2015
4	NetSol Technologies	November 2015
5	System Pvt Ltd	November 2015
6	NextBridge, Islamabad	November 2015
7	IT Intellisense Peshawar, Pakistan	November 2015
8	Xeeonix Pvt Ltd	November 2015
9	parexons IT Solution	November 2015
10	Innovathings Pvt Ltd	November 2015
11	Relevant Professional Groups on Social networks	November 2015

Table 3. List of critical risk factors.

S. No.	Risk Factors	Frequency	%	Practices
01	Lack of green RE practices	38	70	12
02	High power consumption (process, resources and the product itself)	37	68	16
03	High carbon emission throughout the software development	33	61	09
04	Poor software design (architectural, logical, physical and user interface)	32	59	11
05	Lack of ICTs for coordination and communication	30	55	07
06	High resources requirements	27	50	09
07	Lack of coding standards	22	40	10
08	Lack of green software development knowledge	19	35	02

practices are presented in the section 5 of this study.

4. CONDUCATION OF INDUSTRIAL SURVEY

As discussed in Section 3, that we initially conducted SLR and have identified 76 solutions/practices for addressing the critical risk factors in the development of green and sustainable software in the context of multi-sourcing. In order to address RQ2, we conducted questionnaire survey in software multi-sourcing industry to validate the findings of the SLR (identified solutions/practices) and to find any new solution/practice in addition to the SLR findings. We developed the questionnaire based on the inputs from the systematic literature review (SLR) findings i.e. identified solutions/practices. The piloting of the questionnaire was conducted through fellow members of the software engineering research group (SERG-UOM) and required modifications were made to the questionnaire accordingly. Throughout the questionnaire development process, we considered the input/feedback of fellow researchers and existing literature [62-64] .There are two main types of questionnaire format: Open format questionnaire and closed format questionnaire [65]. We have chosen a closed format questionnaire as a tool to gather self-reported data. However, in order to identify new factors from software multi-sourcing industry professionals in addition to the SLR findings, we also included some open ended questions in the questionnaire. In order to define the significance of identified solutions/practices, the respondents were inquired to note each practice's relative value on a 7-point Likert Scale (i.e. Extremely Satisfied, Moderately Satisfied, Slightly Satisfied, Neither, Slightly Dissatisfied, Moderately Dissatisfied, Strongly Dissatisfied). We have used three distinct format of the questionnaire for its distribution across the target population. These include online version, MS Word format (soft), and printed copy (hard copy). However mainly we have used the online survey because of many advantages of online survey over the traditional survey methods as discussed in [66]. Keeping in view all of the mentioned advantages [66] of online survey, we decided to go for online survey mostly. We have used Google survey tool in this research study.

4.1. Data Sources

In order to approach the target population, we sent an invitation letter for consent to various professionals/groups and software development companies as shown in Table 2. Apart from this we also invited various software companies and authors of industry papers to take part in the questionnaire survey. A total of 160 professionals from these mentioned groups showed their willingness in response to the invitation. Consequently we sent the questionnaire form (web link) to the experts. Finally we received 120 responses (filled questionnaires). After the filtration of 120 questionnaires through pre-defined quality criteria, 12 questionnaires/ responses were discarded and finally got 108 questionnaires as our final sample size with the response rate of 68% as shown in Figure 1. Among the final 108 respondents 62 were from the vicinity whereas 46 experts were from offshore countries.

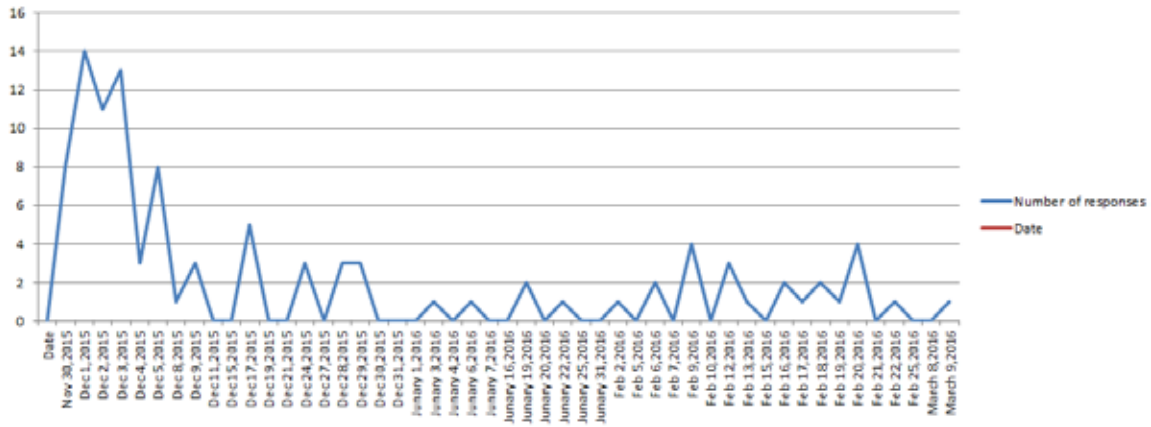

Fig. 1. Survey response rate.

4.2. Data Analysis

In order to analyse the collected data we have used frequency analysis, as it is suitable method for the management of qualitative data [67]. To find the occurrences of each solution/practice, we have used frequency as shown in Table 8 to Table 14. Frequencies can be used for numeric as well as ordinal/nominal data and are useful for comparing across group of variables or within groups of variables. Each solution/practice is analysed by counting its existence in the filled questionnaires. The relative significance of each solution/practice is identified by comparing the existences of one solution/practice against another solution/practice in the development of green and sustainable software in multi-sourced software projects.

5. RESULTS AND DISCUSSION

In this section we have presented the identified 76 practices/solutions for the eight critical risk factors (CRFs). These CRFs are presented in Table 3, while the practices for addressing these CRFs are presented in Table 4 to Table 11.

5.1. Practices for addressing CRF-1: 'Lack of green RE practices'

The data presented in Table 3 indicate that 'lack of green RE practices' is the first CRF (70%) in our findings. We have identified 12 solutions for addressing 'Lack of green RE practices' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 4.

5.2. Practices for Addressing CRF-2: 'High Power Consumption'

The data presented in Table 3 indicates that 'lack of green RE practices' is the 2nd CRF (68%) in our

Table 4. Practices for addressing 'Lack of green RE practices'.

S. No.	Solutions/practices for addressing the Critical Risk factor (CRF-1): 'Lack of green RE practices'	SLR %	Industrial Survey Extremely Satisfied %
CRF1-P-1	In order to meet the customer requirements green gap analysis tool should be used.	11	34%
CRF2-P-2	Define the Shelf life for the Software to be built keeping in view the current and future needs.	02	36%
CRF3-P-3	Update the members of the development team with current market trends.	01	37%
CRF4-P-4	The hardware requirements are chosen such that they should meet the requirements of the software.	01	43%
CRF5-P-5	Prepare proper documentation throughout the software development.	03	42%
CRF6-P-6	Identify functional and non-functional requirements.	03	47%
CRF7-P-7	Use of environment friendly hardware during the software development	01	39%
CRF8-P-8	Use of cloud infrastructure during requirement engineering phase.	01	41%
CRF9-P-9	In case of using cloud by client, vendors should ask from the client during the RE phase about the type of cloud (public, private, hybrid) to be adopted keeping in view the security issues.	04	43%
CRF10-P-10	Hold virtual meeting (online/video conferencing) with offshore workers and customers.	01	39%
CRF11-P-11	Involve end user throughout the requirements gathering and design.	01	45%
CRF12-P-12	Adopt the concept of green requirements engineering.	01	41%

Table 5. Practices for addressing 'High power consumption'.

S. No.	Solutions/practices for addressing the Critical Risk factor (CRF-2): 'High power consumption'	SLR %	Industrial Survey Extremely Satisfied %
CRF2-P-1	Install power management software to keep the computing devices on sleep mode when idle such as Joulemeter, vEC, Span etc.	12	43%
CRF2-P-2	Use LCD screen instead of CRT screen to save considerable amount of energy.	04	44%
CRF2-P-3	Use energy efficient programming paradigm.	03	43%
CRF2-P-4	Install latest computing equipment, if the budget permits.	02	42%
CRF2-P-5	Extend the shelf life of hardware through continuous upgradation.	02	29%
CRF2-P-6	Use of clean energy/green energy sources such as solar power.	04	38%
CRF2-P-7	Arrange online regular meetings throughout the software development in order to minimize travelling between the sites.	03	42%
CRF2-P-8	Use of cloud computing.	05	46%
CRF2-P-9	Use the concept of virtualization.	03	40%
CRF2-P-10	Use of green compiler.	01	30%
CRF2-P-11	The use of power estimation tools.	10	31%
CRF2-P-12	Avoid the use of ad-blocking software which consumes more energy.	01	29%
CRF2-P-13	Keep minimum possible data on webpage.	02	32%
CRF2-P-14	The use of appropriate user devices for online reading such as e-Reader.	01	24%
CRF2-P-15	The use of code optimization techniques and data compressions strategies.	03	33%
CRF2-P-16	Use paperless communication and switch off the computing devices manually when not under usage.	04	37%

Table 6. Practices for addressing 'High carbon emission throughout the software development'.

S. No.	Solutions/practices for addressing the Critical Risk factor (CRF-3): 'High carbon emission throughout the software development'	SLR %	Industrial Survey Extremely Satisfied %
CRF3-P-1	Use of carbon assessments tools throughout the software development such as CF metric.	07	33%
CRF3-P-2	Arrange online regular meetings throughout the software development in order to minimize travelling between the sites.	04	37%
CRF3-P-3	Use of carbon free energy/green energy sources such as solar power.	04	32%
CRF3-P-4	Use sensors and power management software	01	31%
CRF3-P-5	Use of green policies and framework such as code optimization.	05	33%
CRF3-P-6	Use low-powered and green labels hardware for software development.	03	33%
CRF3-P-7	Use of virtualization leads to lower carbon emission.	04	35%
CRF3-P-8	Use of cloud computing.	06	41%
CRF3-P-9	Use electronic mode of communication during the software development.	02	40%

findings. We have identified 16 solutions for addressing 'High power consumption' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 5.

5.3. Practices for Addressing CRF-2: 'High Carbon Emission throughout Software Development'

The data presented in Table 3 indicates that 'High carbon emission throughout the software development' is the 3rd CRF (61%) in our findings. We have identified 09 solutions for addressing 'High carbon emission throughout the software development' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 6.

5.4. Practices for Addressing CRF-2: 'Poor Software Design (Architectural, Logical, Physical and User Interface)'

The data in Table 3 indicates that 'High carbon emission throughout the software development' is the4th CRF (59%) in our findings. We have identified 11 solutions for addressing 'Poor software design (architectural, logical, physical and user interface)' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 7.

5.5. Practices for Addressing CRF-2: 'Lack of ICTs for Coordination and Communication'

The data in Table 3 indicates that 'Lack of ICTs for coordination and communication'' is the 5th CRF (55%) in our findings. We have identified 07 solutions for addressing 'Lack of ICTs for coordination and communication' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 8.

5.6. Practices for Addressing CRF-6: 'High Resources Requirements'

The data in Table 3 indicates that 'High resources requirements' is the 6th CRF (50%) in our findings. We have identified 09 solutions for addressing 'High resources requirements' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 9.

5.7. Practices for Addressing CRF-7: 'Lack of Coding Standards'

The data in Table 3 indicates that 'Lack of coding standards' is the 7th CRF (40%) in our findings. We have identified 10 solutions for addressing 'Lack of coding standards' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 10.

5.8. Practices for Addressing CRF-8: 'Lack of Green Software Development Knowledge'

The data in Table 3 indicates that 'Lack of green software development knowledge' is the 8th CRF (35%) in our findings. We have identified 02 solutions for addressing 'Lack of green software development knowledge' through SLR process initially. We have validated the identified solutions/practices from 108 experts in software development multi-sourcing industry via questionnaire survey as shown in Table 11.

6. LIMITATIONS

In this study we have identified and presented 81 practices/solutions for addressing eight critical risk factors (CRFs) in the development of green software. We have extracted these practices from a sample of (N=102) research papers successfully. However, there are some limitations that need to be documented in this study.

The first limitation is that, some of the authors of selected papers have not reported the original reasons why these practices were considered for green software development. We cannot overcome this threat on our own.

Similarly, another possible threat to validity is that, most of the selected studies were self-reported experiences, case studies, and empirical studies which might be the cause of publication bias.

The third limitation is small sample size of the study. We have selected 102 research papers for data extraction, representing large community of

Table 7. Practices for addressing 'Poor software design (architectural, logical, physical and user interface).

S.No	Solutions/practices for addressing the Critical Risk factor (CRF-4): 'Poor software design (architectural, logical, physical and user interface)'	SLR %	Industrial Survey Extremely Satisfied
CRF1-P-1	Use simple and reusable design	17	45%
CRF4-P-2	Use of energy metrics as a tool to predict the energy consumption in segments at the design stage.	16	25%
CRF4-P-3	Use of agile methods for efficient design and smart coding	16	33%
CRF4-P-4	Support the system architecture through i. Compact design of data structures and efficient algorithms ii. Design smart and efficient functionality that results in an efficient algorithm and fewer lines of code during implementation iii. Components should be reused if possible	16	34%
CRF4-P-5	The design should be flexible to accommodate the future changes easily.	16	34%
CRF4-P-6	Adopt ISO 14000 family of standards related to environmental management which assists the vendor organizations to minimize how their operations negatively affect regarding recyclability or disposal.	16	23%
CRF4-P-7	Use of efficient algorithm to reduce complexity and energy consumption. e.g. encryption algorithm such as Advanced Encryption Standard (AES) consumes less energy than Data Encryption Standard (DES).	05	31%
CRF4-P-8	Avoid repetitive change in design	01	36%
CRF4-P-9	Use of modularization strategies.	10	35%
CRF4-P-10	Use of low level programming languages and avoid use of byte code.	10	27%
CRF4-P-11	Improve usability of the user interface of the software by using simple interface.	03	45%

Table 8. Practices for addressing 'Lack of ICTs for coordination and communication'.

S. No.	Solutions/practices for addressing the Critical Risk factor (CRF-5): 'Lack of ICTs for coordination and communication'	SLR %	Industrial Survey Extremely Satisfied
CRF5-P-1	Use latest ICTs for communication such as email, Skype, Viber, IMO etc.	05	57%
CRF5-P-2	Prepare and maintain the software documents in electronic format (E-format).	05	47%
CRF5-P-3	Avoid frequent visits instead use modern communication tools.	01	35%
CRF5-P-4	Use of video conferencing for meetings with other co-workers during the software development.	06	40%
CRF5-P-5	The use of E-reading devices such as e-Reader.	01	
CRF5-P-6	Perform data management, data transmission, and data compilation in green and sustainable fashion.	02	32%
CRF5-P-7	Establish paperless offices.	01	35%

green software. A higher sample size could deliver more accurate and robust results.

Similarly, another limitation of the study is: we have designed the following two search strings as shown below.

λ1: ("Green software" OR "sustainable software") AND ("practices" OR "solutions" OR "techniques") AND ("multi-sourcing")

λ2: Green software" OR "sustainable software") AND ("practices" OR "solutions" OR "techniques")

Where λ1 denotes search string to retrieve empirical studies regarding the practices for the development of green software multi-sourced software projects, while λ2 denotes search string to retrieve empirical studies regarding the practices for the development of green software in general context. The results of search string (λ1) were very poor and almost negligible. Consequently we decided, to implement search string λ2 after detailed discussions with experts of the software engineering research group (SERG_UOM) at the university, to implement search string λ2. The practices, identified through the SLR (using search string λ2), have been validated through empirical studies in multi-sourcing software industry via questionnaire survey. Lastly, our search strategy may have missed out some relevant papers which are not a systematic omission.

Secondly, we have conducted online questionnaire survey in the software development multi-sourcing industry to validate the findings of the SLR and to find any new solution/practice apart from the identified ones. Finally we got 108 questionnaires as the final sample with response rate of 68%. Among the final 108 respondents 62 were from the vicinity whereas 46 experts were from offshore countries. It would be better if we should have involved more offshore professionals instead of the local professionals but it was not possible due to limited resources and time at this stage. Due to limited number of responses from foreign experts, one should be careful while generalizing the results.

7. CONCLUSIONS

In this study we have presented the identified 76 practices/solutions for addressing the aforementioned eight critical risk factors (CRFs). The solutions/practices are extracted from sample of (N=102) research papers via SLR process. We have validated the identified solutions/practices from 108 experts in the software development multi-sourcing industry. The findings of this study can help vendor organizations to address the CRFs and to evaluate their readiness for the development of green and sustainable software in multi-sourced projects.

For CRF-1: 'Lack of green RE practices' we have identified 12 practices as presented in Table 4, for CRF-2: 'High power consumption' we have identified 16 practices as shown in Table 5, for CRF-3: 'High carbon emission throughout the software development' for this factors we identified 09 practices as presented in Table 6, for CRF-4: 'Poor software design (architectural, logical, physical and user interface)', we have identified 11 practices as shown in Table 7, for CRF-5: 'Lack of ICTs for coordination and communication' we have identified 07 practices as presented in Table 8, for CRF-6: 'High resources requirements', we have identified 09 practices as shown in Table 9, and for CRF-7:'Lack of coding standards', we have identified 10 practices/solutions as presented in Table 10 and for CRF-8: 'Lack of green software development knowledge', 02 solutions/practices have been identified as shown in Table 11.

We have validated the identified 76 practices/solutions from 108 experts in software development multi-sourcing industry via questionnaire survey. The findings of this study help vendor organizations to address the CRFs in order to evaluate their readiness for the development of green and sustainable software in multi-sourced software projects.

However, we recommend more empirical studies on green and sustainable software development specific in the context of software development multi-sourcing. This will increase confidence in our findings and will support software development multi-sourcing vendor organizations to develop green software in multi-soured projects.

The eventual goal of this study is to develop 'Green Software Multi-Sourcing Readiness Model' from vendor's perspective that will assist software multi-sourcing vendor organizations in developing green and sustainable software in multi-sourced projects. This paper contributes only

Table 9. Practices for addressing 'High resources requirements'.

S. No.	Solutions/practices for addressing the Critical Risk factor (CRF-6): 'High resources requirements'	SLR %	Industrial Survey Extremely Satisfied
CRF6-P-1	Deploy virtualization of server resources.	06	38%
CRF6-P-2	Utilize cloud services for both software and hardware.	04	37%
CRF6-P-3	Use of resource saving default configurations.	02	31%
CRF6-P-4	Sustainable use of the resources.	03	41%
CRF6-P-5	Use the concept of power aware computing.	02	38%
CRF6-P-6	Use of energy efficient/green resources.	01	44%
CRF6-P-7	Deploy mechanism for measurement of the energy consumed by the nodes.	01	32%
CRF6-P-8	Use of software engineering standards during the software development such as CMMI etc.	01	43%
CRF6-P-9	Save resources through the use of teleconferencing, e-Reader device, paperless communication, and use of power-saving devices.	05	40%

Table 10. Practices for addressing 'Lack of coding standards'.

S. No.	Solutions/practices for addressing the Critical Risk factor (CRF-7): 'Lack of coding standards'	SLR %	Industrial Survey Extremely Satisfied
CRF7-P-1	Follow professional coding conventions while programming in order to improve the software maintainability.	02	35%
CRF7-P-2	Use of efficient software techniques in coding. i.e. multi-threading, vectorization.	05	40%
CRF7-P-3	Avoid hardware-specific Programming Interface (API's).	01	26%
CRF7-P-4	Avoid using ad-hoc programming approach.	04	23%
CRF7-P-5	Avoid bad smells in coding such as duplicate code, long methods, data clumps, and shotgun surgery etc.	04	33%
CRF7-P-6	Use of automated tools such as automatic code generation tools and automatic code review tools.	02	33%
CRF7-P-7	Establish energy efficient coding by writing clean code, documenting code, less number of code and use of pair-programming.	10	30%
CRF7-P-8	Use of modularization strategies.	10	35%
CRF7-P-9	Use of energy aware compilers to analyse software programs at run time and reshape software source code by applying several green aspects during code transformation.	10	25%
CRF7-P-10	Use of low level programming languages and avoid use of byte code.	10	27%

Table 11. Practices for addressing 'Lack of green software development knowledge'.

S. No.	Solutions/practices for addressing the Critical Risk factor (CRF-8): 'Lack of green software development knowledge'	SLR %	Industrial Survey Extremely Satisfied
CRF8-P-1	Arrange special training for the development teams regarding green and sustainable software development.	03	42%
CRF8-P-2	Update the members of the development team with current market trends.	01	37%

one component to our proposed model [24]. We have adopted a similar research design in our previous work [68, 69].

8. REFERENCES

1. Dick, M., S. Naumann. & N. Kuhn. A model and selected instances of green and sustainable software. In: *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustain-ability Resilience*. Springer, p. 248-259 (2010).
2. Naumann, S., M. Dick, E. Kern., & T. Johann. The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems* 1: 294-304 (2011).
3. Mahmoud, S.S., & I. Ahmad. A green model for sustainable software engineering. *International Journal of Software Engineering and Its Applications* 7: 55-74 (2013).
4. Naumann, S., E. Kern & M. Dick. Classifying green software engineering - The GREENSOFT model. In: *Proceedings of the 2nd Workshop Energy Aware Software-Engineering and Development (EASED@ BUIS)* 4:13-14 (2013).
5. Singh., V. Kumar, & D. Vander Meer. Estimating the energy consumption of executing software processes. In: *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing,* p. 94-101 (2013).
6. Shenoy, S. Sanath, & R. Eeratta. Green software development model: An approach towards sustainable software development. In: *Annual IEEE India Conference (INDICON),* p. 1-6 (2011).
7. Sissa, G. Green software. *UPGRADE. The European Journal for the Informatics Professional* 11: 53-63 (2010).
8. Lago, P., R.Kazman, N. Meyer, M. Morisio., H. A. Müller, & F. Paulisch. Exploring initial challenges for green software engineering: Summary of the first GREENS workshop, at ICSE 2012. *ACM SIGSOFT Software Engineering Notes* 38: 31-33 (2013).
9. Ray, S. Green software engineering process: moving towards sustainable software product design. *Journal of Global Research in Computer Science* 4: 25-29 (2013).
10. Calero, C. & M. Piattini. *Green in Software Engineering*. Springer (2015).
11. Samiksha, R.S. & M. Chavan. Green Computing: An essential trend for secure future. In: *Proceedings of National Conference on Emerging Trends: Innovations and Challenges in IT,* p. 19: 20 (2013).
12. Cai, Y. Integrating sustainability into undergraduate computing education. In: *Proceedings of 41st ACM Technical Symposium on Computer Science Education*, Wisconsin, USA, p. 524-528 (2010).
13. Harmon R. R., & N. Auseklis. Sustainable IT services: Assessing the impact of green computing practices. In: *International Conference on Management of Engineering & Technology, PICMET, Portland,* p.1707 - 1717 (2009).
14. Cai, S., X. Chen, & I. Bose. Exploring the role of IT for environmental sustainability in China: An empirical analysis. *International Journal of Production Economics* 146: 491-500 (2013).
15. Lo, C.T.D., & K. Qian. Green computing methodology for next generation computing scientists. In: *IEEE 34th Annual Computer Software and Applications Conference (COMPSAC)* 250-251 (2010).
16. Tushi, B.T. *An Archival Analysis of Green Information Technology: The Current State and Future Directions.* Doctoral dissertation, Queensland University of Technology, Queensland, Australia (2015).
17. Erdelyi, K. Special factors of development of green software supporting eco sustainability. In: *Intelligent Systems and Informatics (SISY), IEEE 11th International Symposium, Subotica,* p. 337-340 (2013).
18. Harmon, R., H. Demirkan, N. Auseklis, & M. Reinoso. From green computing to sustainable IT: developing a sustainable service orientation. In: *System Sciences (HICSS), 43rd Hawaii International Conference*, p.1-10 (2010).
19. Donnellan, B., C. Sheridan & E. Curry. A capability maturity framework for sustainable information and communication technology. *IT Professional* 13: 33-40 (2011).
20. Babar, M.A., J.M. Verner, & P. Nguyen. Establishing and maintaining trust in software outsourcing relationships: An empirical investigation. *The Journal of Systems and Software* 80: 1438–1449 (2007).
21. Qu, G., L. Shen & X. Bao. Vendors' team performance in software outsourcing projects: From the perspective of transactive memory systems behavioral characteristics. *Nankai Business Review International* 5: 290-308 (2014).
22. Power, M.J. *The outsourcing handbook how to implement a successful outsourcing process*. Kogan Page Publishers London, (2006)
23. Kehal, H. *Outsourcing and Offshoring* In: *The 21st Century: A Socio-Economic Perspective*. Lgi Global (2006).
24. Salam, M. & S.U. Khan. Green software multi-sourcing readiness model (gsm-rm) from vendor's perspective. *Science International (Lahore)* 26: 1421-1424 (2014).
25. Khan, R.U. & S.U. Khan. Green IT-Outsourcing Assurance Model. In: *Global Software Engineering Workshops (ICGSEW), IEEE 8th International Conference,* p. 84-87 (2013).

26. Hijazi, H., S. Alqrain., H. Muaidi., & T. Khdour. Risk factors in software development phases. *European Scientific Journal* 10:3 (2014).
27. Patil, S. & R. Ade. A software project risk analysis tool using software development goal modelling approach. In: *Information Systems Design and Intelligent Applications,* p. 767-777 (2015).
28. Persson, J.S. & B.R. Schlichter. Managing risk areas in software development offshoring: A cmmi level 5 case. *Journal of Information Technology Theory and Application* 16: 5-24 (2015).
29. Hua, W. & Y. Longyong. Software risk assessment method based on Fuzzy neural network. In: *International Conference on Computer Science and Intelligent Communication,* p. 159-162 (2015).
30. Elzamly, A. & B. Hussin. Classification and identification of risk management techniques for mitigating risks with factor analysis technique in software risk management. *Review of Computer Engineering Research* 2: 22-38 (2015).
31. Prikladnicki, R. & M.H. Yamaguti. Risk management in global software development: A position paper. In: *Third International Workshop on Global Software Development* (2004).
32. Chou, D.C. Risk identification in green IT practice. *Computer Standards & Interfaces* 35: 231-237 (2013).
33. Penzenstadler, B., A. Raturi., D. Richardson, C. Calero, H. Femmer, & X. Franch. Systematic mapping study on software engineering for sustainability (SE4S). In: *Proceedings of 18th International Conference on Evaluation and Assessment in Software Engineering ACM, UK,* p. 1-14 (2014).
34. Becker, C., R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, & N. Seyff, & C.C. Venters. Sustainability design and software: The karlskrona manifesto. In: *Proceedings of the 37th International Conference on Software Engineering-Volume 2, IEEE Press,* p. 467-476 (2015).
35. Weyns, D., M. Caporuscio, B. Vogel, & A. Kurti. Design for sustainability = Runtime adaptation∪ evolution. In: *Proceedings of 2015 European Conference on Software Architecture Workshops, ACM* 62: 1-7 (2015).
36. Raturi, A., B. Penzenstadler, B. Tomlinson, & D. Richardson. Developing a sustainability non-functional requirements framework. In: *Proceedings of the 3rd International Workshop on Green and Sustainable Software, ACM,* p. 1-8 (2014).
37. Kern, E., S. Naumann, & M. Dick. Processes for green and sustainable software engineering processes for green and sustainable software engineering. In: *Green in Software Engineering, Springer International Publishing,* p. 61-81 (2015).
38. Bartenstein, T. W. & Y. D. Liu. Green streams for data-intensive software. In: *Proceedings of the 2013 International Conference on Software Engineering, IEEE Press*, p. 532-541 (2013).
39. Easterbrook, S.M. Climate change: a grand software challenge. In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, ACM* 99-104 (2010).
40. Moraga, M. A. & M.F. Bertoa. Green software measurement. In: *Green in Software Engineering, Springer,* p. 261-282. (2015).
41. Naumann, S., E. Kern., M. Dick., & T. Johann. Sustainable software engineering: process and quality models, life cycle, and social aspects. In: *ICT Innovations for Sustainability, Springer,* p. 191-205 (2015).
42. Lago, P. Challenges and opportunities for sustainable software. In: *Proceedings of the Fifth International Workshop on Product LinE Approaches in Software Engineering, IEEE Press,* p. 1-2 (2015).
43. Arakelyan, A. & D. Lamas. Situating a design space for sustainable software appropriation. In: *International Conference on Human-Computer Interaction, Springe.* p. 665-673 (2014).
44. Beghoura, M.A., A. Boubetra, & A. Boukerram. Green software requirements and measurement: random decision forests-based software energy consumption profiling. *Requirements Engineering* 1-14 (2015).
45. de, G., I.G. Rodríguez, M. Piattini, & R.P. Castillo. Green software maintenance. In: *Green in Software Engineering, Springe,* p. 205-229 (2015).
46. Betz, S. & T. Caporale. Sustainable software system engineering. In: *IEEE Fourth International Conference on, Big Data and Cloud Computing (BdCloud),* p. 612-619 (2014).
47. Curran, R., N. Wognum, & M. Borsato. Transdisciplinary Lifecycle analysis of Systems. In: *Proceedings of 22nd ISPE International Conference on Concurrent Engineering, IOS Press,* p. 20-23 (2015).
48. Li, F., S. Qanbari, M. Vögler, & S. Dustdar. Constructing green software services: From service models to cloud-based architecture. In: *Green in Software Engineering, Springer,* p. 83-104 (2015).
49. Amri, R. & N.B.B. Saoud. Towards a Generic Sustainable Software Model. In: *Fourth International Conference on Advances in Computing and Communications (ICACC),* p. 231-234 (2014).
50. Penzenstadler, B. Infusing Green: Requirements engineering for green in and through software systems. In: *RE4SuSy@ RE,* p. 44-53 (2014).
51. Kim, D.K., J. Ryoo, & S. Kim. Building sustainable software by preemptive architectural design using tactic-equipped patterns. In: *Ninth International Conference on Availability, Reliability and Security (ARES),* p. 484-489 (2014).

52. Jetley, R. & A. Nair & P. Chandrasekaran, & A. Dubey. Applying software engineering practices for development of industrial automation applications. In: *2013 11th IEEE International Conference on Industrial Informatics (INDIN),* p. 558-563 (2013).
53. Dick, M., J. Drangmeister, E. Kern, & S. Naumann. Green software engineering with agile methods. In: *2nd International Workshop on Green and Sustainable Software (GREENS),* p. 78-85 (2013).
54. Ardito, L. & M. Morisio. Green IT-Available data and guidelines for reducing energy consumption in IT systems. *Sustainable Computing: Informatics and Systems* 4: 24-32 (2014).
55. Lami, G., F. Fabbrini., & M. Fusani. Software sustainability from a process-centric perspective. In: *Systems, Software and Services Process Improvement*. Springer, p. 97-108 (2012).
56. Sun, Y., Y. Zhao, Y.Song, Y.Yang., H. Fang, H. Zang, Y. Li, & Y. Gao. Green challenges to system software in data centers. *Frontiers of Computer Science in China* 5: 353-368 (2011).
57. Kitchenham, B. & C. Charters. *Guidelines for Performing Systematic Literature Reviews in Software Engineering.* Joint Report, Keele University and Durham University (2007).
58. Kitchenham, B. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33: 1-26 (2004).
59. Korkala, M. & F. Maurer. Waste identification as the means for improving communication in globally distributed agile software development. *Journal of Systems and Software* 95: 122-140 (2014).
60. Sriram, R. & S.K. Mathew. Global software development using agile methodologies: A review of literature. In: *2012 IEEE International Conference on Management of Innovation and Technology (ICMIT),* p. 389-393 (2012).
61. Salam, M. & S.U. Khan. Systematic Literature Review Protocol for Green Software Multi-sourcing with Preliminary Results. *Proceeding Of The Pakistan Academy Of Sciences* 52: 285–300 (2015).
62. Linaker, J., S.M. Sulaman, R. M. Mello, & H. Martin. *Guidelines for Conducting Surveys in Software Engineering.* https://lup.lub.lu.se/search/ publication/ 5366801 (2015).
63. Ahmad, A. & S.J. Kolla. *Effective Distribution of Roles and Responsibilities in Global Software Development Teams*. School of Computing Blekinge Institute of Technology, Karlskrona Sweden (2011).
64. Khan, S.U. & M.K. Niazi. A preliminary structure of software outsourcing vendors' readiness model. In: *Doctoral symposium at 11th International Conference on Product Focused Software Development and Process Improvement (PROFES 2010) Limerick, Ireland, ACM,* p. 76-79 (2010).
65. Khan, S.U. & M.I. Azeem. Intercultural challenges in offshore software development outsourcing relationship: An Empirical study. *Proceedings of the Pakistan Academy of Sciences* 53: 75-88 (2016).
66. Heiervang, E. & R. Goodman. Advantages and limitations of web-based surveys: evidence from a child mental health survey. *Social Psychiatry and Psychiatric Epidemiology* 46: 69-76 (2011).
67. Ali, S. & S.U. Khan. Empirical investigation of risk factors for establishing software outsourcing partnership from vendor's perspective. *Proceedings of the Pakistan Academy of Sciences* 52: 315–328 (2015).
68. Ali, S. & S.U. Khan. Software Outsourcing Partnership Model: An Evaluation Framework for Vendor Organisations. *The Journal of Systems & Software* 117: 402-425 (2016).
69. Khan, S.U. *Software Outsourcing Vendors Readiness Model (SOVRM).* PhD thesis, Keele University, UK (2011).