Shola Oyedeji

SOFTWARE SUSTAINABILITY BY DESIGN

ACTA UNIVERSITATIS LAPPEENRANTAENSIS 885

Shola Oyedeji

SOFTWARE SUSTAINABILITY BY DESIGN

Dissertation for the degree of Doctor of Philosophy to be presented with due permission for public examination and criticism in the room 1316 at Lappeenranta-Lahti University of Technology LUT, Lappeenranta, Finland on the 5th of December, 2019, at noon.

Acta Universitatis
Lappeenrantaensis 885

Supervisors Adjunct Professor Birgit Penzenstadler
LUT School of Engineering Science
Lappeenranta-Lahti University of Technology LUT
Finland

Professor Jari Porras
LUT School of Engineering Science
Lappeenranta-Lahti University of Technology LUT
Finland

Reviewers Dr. Colin C. Venters
Department of Computer Science
School of Computing and Engineering
University of Huddersfield
United Kingdom

Professor Patricia Lago
Software Engineering
Vrije Universiteit Amsterdam
Netherlands

Opponents Assistant Professor Nelly Condori-Fernandez
Computer Science
University of Coruña
Spain

Dr. Colin C. Venters
Department of Computer Science
School of Computing and Engineering
University of Huddersfield
United Kingdom

ISBN 978-952-335-456-2
ISBN 978-952-335-457-9 (PDF)
ISSN-L 1456-4491
ISSN 1456-4491

Lappeenranta-Lahti University of Technology LUT
LUT University Press 2019

Abstract

Shola Oyedeji
Software Sustainability by Design
Lappeenranta 2019
84 pages
Acta Universitatis Lappeenrantaensis 885
Diss. Lappeenranta-Lahti University of Technology LUT
ISBN 978-952-335-456-2, ISBN 978-952-335-457-9 (PDF), ISSN-L 1456-4491, ISSN 1456-4491

In our current world, software impacts almost everything; it connects people and forms the cornerstone for the economy, as such, has an impact on sustainability and the emerging sustainable development goals (SDGs). Currently, sustainability is a concept with different interpretations and perceptions in the software engineering community. Software sustainability design, development and measurement are evolving and require more research in software engineering. There are only a few concrete guidelines, measures, tools and examples for software architects, developers, requirement engineers and companies to use in the design, development and measurement of software sustainability efficiently and effectively.

This research aims to explore how best to guide and support stakeholders (requirement engineers, software architects, developers, and companies) in the design, development and measurement of software systems, based on sustainability dimensions (economic, social, individual, environment and technical) in software engineering. This work will serve as the first step towards alleviating the challenge of understanding what sustainability means in software design, development and measurement for different stakeholders.

This research has been conducted using the design science research methodology to identify and design solutions (artefacts) for the problems of sustainability in software design, development and measurement. These artefacts are the Sustainable Business Goal Question Metric (S-BGQM), the Software Sustainability Design Catalogue (SSDC), the Framework for Sustainability of Software System Design (FSSSD) and the Template for Software Sustainability Requirement Best Practice documentation.

The overall outcome from this research is tailored towards supporting sustainability in software design and development practices. Output from this research provides the building block to foster more research investigation on tools and methods to support shift in stakeholders' mindsets towards adopting sustainability in a way that translates into software design decisions and practices.

Keywords: software sustainability, software sustainability requirement, software sustainability design, software sustainability perceptions, software sustainability analysis, software sustainability measures, karlskrona manifesto principles, software sustainability measurement

Acknowledgements

I want to say big thanks to my supervisors, Adjunct Professor Birgit Penzenstadler and Professor Jari Porras, for providing me with good insight, supervision and support during all my research activities for this dissertation.

The foundation for my research in this dissertation was provided by Professor Ahmed Seffah who hired me initially as his PhD student. I appreciate all your efforts in supporting and guiding me to understand how best to translate my ideas into good research.

I acknowledge and thank reviewers of this dissertation, Dr. Colin C. Venters and Professor Patricia Lago for all your valuable comments and feedbacks which helped me improve this dissertation.

I would like to express gratitude to Mariam Abdulkareem, Bilal Naqvi, Victoria Palacin, Andrey Sultan, Dr. Antti Knutas, Dr. Annika Wolff, Ola Mikhail Adisa and Dr. Janne Parkkila for always been there to discuss my research and assist me throughout this journey of PhD.

My master program Erasmus Mundus Joint Master Degree (EMJMD) in Pervasive Computing and Communications for Sustainable Development (PERCCOM) provided me the opportunity to further my studies for PhD. I would like to thank Professor Eric Rondeau for selecting me into PERCCOM program, Jean-Philippe Georges, Thierry Divoux, Francis Lepage, Professor Olaf Droegehorn, Professor Karl Andersson, Dr. Josef Hallberg, Professor Gérard Morel, Ah-Lian Kor, Professor Karl-Erik Michelsen and other PERCCOM lecturers for impacting me with knowledge of sustainability in ICT. All my friends in Cohort 1, 2, 3 and 4 from the PERCCOM program, you rock and I love you all.

I appreciate the support and assistance of my colleagues at the Software Engineering Unit and our boss Professor Kari Smolander. Special thanks to Tarja Nikkinen, Ilmari Laakkonen and Petri Hautaniemi for the administrative and technical support.

This dissertation would not be complete today without the support of caring friends and family in Finland and abroad: Ibrahim & Bilikisu, Larry & Munifah, Ayo & Nike, Moshood Afolabi, Mahmoud El-sebaie, Hicham Benkeltoum, Ezeanowi Nnaemeka Celestine, Alex Dankwah, Imtiaz Ahmed, Kuburat Abdulkareem, Rose Alshawwaf, Agnes Asemokha, Misbah Mustapha, Ibrahim Adebayo Ola, Mehar Ullah, Amin Esmaeili, Obi Chike Hilary, Alharith Asim Surij, Moses Irunokhai, Muhammad Ahsan, Mahdi Merabtene, Abdelrahman Azzuni, Fasasi Olufemi, Banji Seun, Abass Abolaji Adeniji, Anar Bazarhanova, Niklas Kolbe, Dimitar Minovski, Kola Adebayo, Ashraf Abdo, Md Anowarul Abedin, Ornela Bardhi, Dagnachew Azene Temesgene, Melanie Pittumbur, Sumeet Thombre, Rajeshwari Chatterjee, Jonathan Pucher, Stefanos Georgiou, Julien Da, Khan Mohammad Habibullah, Ahmad Azwan Ja'afar, Samuri Firusi, Mishael Akpabio, Nurul Haida Akhir, Moy Zulaikha, UshaDevi Balakrishnan, Futri Najla Saleh, Nuru Salihu and all my gym buddies.

To my dad, mum, siblings, uncles and aunties, I want to say big thanks for always listening to my struggles, supporting and believing in me throughout this research towards my dissertation.

Finally, big thanks to my lovely caring wife, Diajeng Rahmawati for your understanding and patience, especially during all those late nights in office. I appreciate all your efforts in making this process easy for me and those encouragements over the years.

Shola Oyedeji
November 2019
Lappeenranta, Finland

This thesis is dedicated to my parents (Sule & Fausat Oyedeji), Diajeng's parents (Firman & Utami Anwar), Eksannudin Elias, Dr. Kalaivani Chellappan, Norlini Ramli and all my family members for their endless love and support

Contents

List of publications

This thesis is based on the following publications; the publications are titled as Publications I - V.

I. S. Oyedeji, A. Seffah, and B. Penzenstadler, 2017. Sustainability quantification in requirements informing design. *Proceedings of 6th International Workshop on Requirement Engineering for Sustainable System co-located with the 25th International Conference on Requirements Engineering.*

II. S. Oyedeji, A. Seffah, and B. Penzenstadler, 2018. A catalogue supporting software sustainability design. *Sustainability*, Vol. 10, no. 7, pp. 1–30.

III. S. Oyedeji, A. Seffah, and B. Penzenstadler, 2018. Classifying the measures of software sustainability. *Proceedings of the 4th International Workshop on Measurement and Metrics for Green and Sustainable Software Systems co-located with 12th International Conference on Empirical Software Engineering and Measurement*

IV. S. Oyedeji and B. Penzenstadler, 2018. Karlskrona manifesto: Software requirement engineering good practices. *Proceedings of the 7th International Workshop on Requirements Engineering for Sustainable Systems co-located with the 26th International Conference on Requirements Engineering.*

V. S. Oyedeji, B. Penzenstadler, M. O. Adisa, and A. Wolff, 2019. Validation study of a framework for sustainable software system design and development. *Proceedings of 6th International Conference on ICT for Sustainability, ICT4S.*

Author's contribution to publications

I. The dissertation candidate is the principal author and investigator of this publication, who conceived the idea and discussed it with the co-authors.

II. The dissertation candidate is the main author of this publication, who conceived the idea of the publication based on discussion with the third author (main supervisor), and carried out the planning and execution of data collection through literature reviews, analyses and reporting under the supervision of the second and third authors.

III. The dissertation candidate is the principal author and investigator of this publication, under the supervision of the second and third author.

IV. The dissertation candidate is the main author of the publication and carried out the research investigation and reporting after discussions with the second author, who supervised the research process.

V. The dissertation candidate is the principal author and investigator of the publication, conducted the planning and execution of the case study and documented the results and findings.

Nomenclature

EF	Energy efficiency
EU	European Union
FS	Functional suitability
FSSSD	Framework for Sustainability of Software System Design
GHG	Global greenhouse gas
ICT	Information and communications technology
KMSD	Karlskrona manifesto for sustainability design
PE	Performance efficiency
PUE	Power usage effectiveness
S-BGQM	Sustainable business goal question metric
SDGs	Sustainable development goals
SDLC	Software development lifecycle
SE	Software engineering
SSDC	Software sustainability design catalogue

1 Introduction

Sustainability is now one of the world's major challenges (Tilbury *et al.*, 2002; Ehrenfeld, 2008; United Nations, 2013). The importance of sustainability in all aspects of human lives and development is further highlighted by the collection of 17 sustainable development goals (SDGs) (United Nations, 2015). These SDGs indicates the need for global action towards sustainability and software has a role to play. Software is a key factor and catalyst for all economic activities using information and communications technology (ICT) and a major driver linking all sectors. Currently, there has been limited research investigations and solutions on how these SDGs can be achieved through ICT (Wu *et al.*, 2018), which requires a global multi-disciplinary efforts with joint collaboration of companies in various industries.

ICT itself contributes an estimated 2% of global CO_2 emissions and is accountable for approximately 8% of the European Union's (EU) electricity consumption (Calero and Piattini, 2015), but the percentage of emissions induced by software-intensive systems is much higher. As stated in an Ericsson report, ICT can help reduce global greenhouse gas (GHG) emissions for companies by 15% (Ericsson, 2014) and software is a key component in the reduction. Currently, there are some companies using sustainable development for software innovations by creating new opportunities to lower costs, add value and gain competitive advantages (Calero and Piattini, 2015). There is also an increasing growth in the percentage of organisations redesigning their entire business models to incorporate sustainability according to a Microsoft report, as well as an IBM global CEO study on sustainability (Microsoft, 2015; IBM, 2010; Nidumolu, Prahalad and Rangaswami, 2013).

Sustainability has been on the agenda of many companies for decades, but their environmental, social and governance activities are disconnected from their core strategy because they lack understanding on how to integrate sustainability into their business models (Bonini and Görner, 2011). Software affects all facets of our lives and is a driver for sustainability and greening in companies (Kocak, 2013). However, the 'How' and 'Where' to apply each of the sustainability dimensions and how to evaluate the impact on software applications is still a challenge for many companies during software design (Kocak, 2013; Oyedeji, Seffah and Penzenstadler, 2018a).

The problem of long-term thinking is now a concern in software design, with focus on different research angles for holistic consideration of sustainability in software system during design and the environment in which it will operate (Becker, 2014). Software engineering (SE) and software designers have an important role to play in promoting sustainability through the design of sustainable software systems. The way we design and consequently use software systems has a significant impact and can greatly influence human perceptions of sustainability (Mahaux, Heymans and Saval, 2011). Although design is a central phase of any software development process (Freeman, 1980), there has been limited research on software sustainability design. Professionals' perception of sustainability affects the way sustainability has been applied in software development

(Groher and Weinreich, 2017) because different lifestyles, values and practices affect how sustainability is treated (Ilstedt, Eriksson and Hesselgren, 2017).

The challenge of sustainability in SE is that most research currently does not cover the full software development life cycle phases to show how sustainability can be an integral part of each development phase. A systematic mapping study (Wolfram, Lago and Osborne, 2017) shows the classification of relevant publications on sustainability in SE. Based on the results of the systematic mapping study (see Figure 1), the distribution of research efforts on each knowledge area according to SWEBOK (Bourque and Fairley, 2014) indicates that not all software development life cycle phases are proportionally addressed, from the software project definition phase to the user requirements definition, system requirement definition, analysis and design, development, integration and testing, implementation and finally maintenance.

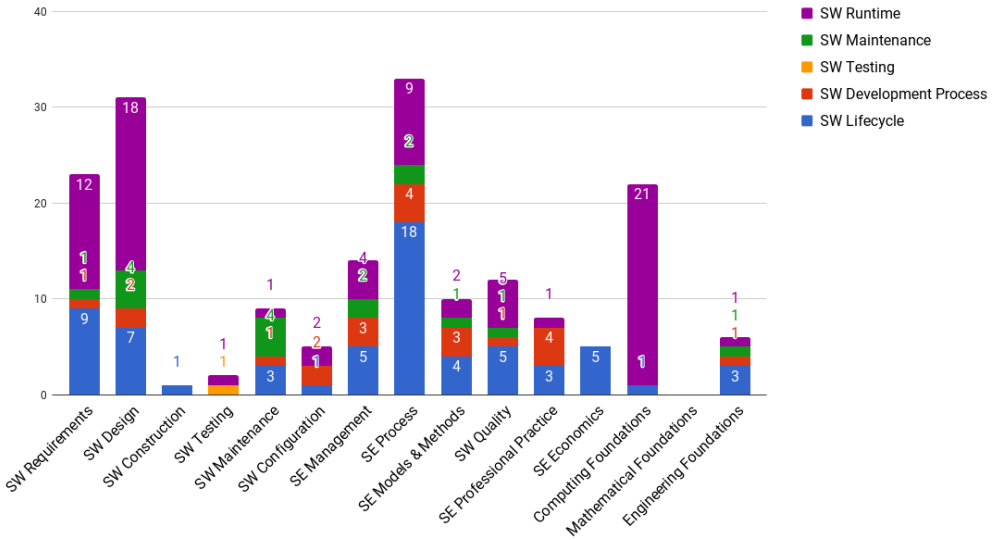


Figure 1. Distribution of sustainability research publications, according to each knowledge area (Wolfram, Lago and Osborne, 2017)

One of the main problems of sustainability in software design is that for software designers, even with a systems approach, there are only a few existing tools for sustainability. Instead, designers must learn to patch a series of disparate sustainability understandings to address the multiple dimensions of sustainability during software design and development (Shedroff, 2009). Furthermore, the challenge for most companies is that there is little understanding of how sustainability can be applied by software and requirement engineering professionals to facilitate sustainability design as an established part of the software development process: specifically, the requirements

engineering and design processes (Mahaux and Canon, 2012; Chitchyan *et al.*, 2016; Jannat, 2016).

The problem of 'How' and 'Where' to apply sustainability in software design and challenge of understanding in what way sustainability can impact positively in software design by stakeholders necessitated this research. Stakeholders in the context of this research are software architects, developers, designers, requirement engineers, researchers and companies.

This research focuses on the sustainability practices used in designing and developing software within software engineering which is usually called sustainable software engineering. Sustainable software engineering is a process which 'aims to create reliable, long-lasting software that meets the needs of users while reducing environmental impacts'(Amsel *et al.*, 2011). This research also supports software engineering for sustainability which focuses on how software can help and support sustainability while in use (Oyedeji, Seffah and Penzenstadler, 2018b).

The overall research is directed towards providing support and guidance to stakeholders in the integration of sustainability during software design, development and measurement through the use of a sustainability framework, catalogue and requirement template. The following points are the goals of this thesis:

- The main goal of this research is to create artefacts for software sustainability design that can guide and support stakeholders to easily adopt and institutionalise sustainability in their mainstream software development and management processes, assess the cost-benefit objectively while creating a business model associated with the sustainability of their software system.
- The second goal is to improve software sustainability design practices through software design decisions that will translate into sustainability in software design.

1.1 Research Questions

The research questions are centred on the question of '***How to guide and support software developers in the design, development and measurement of software sustainability'.***

The research questions and Table 1 describe each research question according to the publications used in this thesis:

- **Research Question 1 (RQ1):** How to elicit software sustainability requirements in software design?
- **Research Question 2 (RQ2):** How to measure and evaluate software system sustainability?
- **Research Question 3 (RQ3):** How to record good practices for software sustainability design and development?

Table 1. Publications' relation to research questions

No	Publications	RQ1	RQ2	RQ3
1	Sustainability Quantification in requirements Informing Design	X	X	
2	A Catalogue Supporting Software Sustainability Design	X	X	
3	Classifying the Measures of Software Sustainability	X	X	
4	Karlskrona Manifesto: Software Requirement Engineering Good Practices	X		X
5	Validation Study of a Framework for Sustainable Software System Design and Development	X	X	X

1.2 Research Contribution

This thesis provides a building block to advance the state of the art in software sustainability design, development and measurement through the identification and documentation of different research gaps, challenges and problems for software sustainability design in academia and its application in industry.

Also, Software Sustainability Design Catalogue (SSDC) is proposed as a guideline for stakeholders. Another core contribution from this thesis is the development of the Framework for Sustainability of Software System Design (FSSSD) for software architects and developers, which addresses each software development life cycle phase with different sustainability goals, concepts, methods, tools, measures and indicators.

Further, a method for documenting software sustainability requirements' best practices and a template for documenting these practices were created for reuse by both experienced and novice stakeholders, researchers and governmental agencies interested in software sustainability design.

1.2.1 **Overview of Research Contribution**

The five publications in this research contributed to software sustainability design in different phases of the software development lifecycle (SDLC). These SDLC phases, for better categorisation of the research contributions, are grouped into Requirements, Design and Development, Measurement, Documentation and Validation. Figure 2 summarises the research contribution based on the categorisation of all publications.

- **Requirement**: The requirement stage covers the project definition, user requirements and system requirements phases. Research output from Publication 1 (software sustainability requirements and sustainable business goal question metrics [S-BGQM]) contributed to facilitating software sustainability requirement as a core part of the three SDLC phases during software design and development.

- **Design and development:** The grouping for design and development includes the analysis, Design and Development phases of SDLC. Research output from Publication 2 contributed to this stage. First, the SSDC is a tool to educate stakeholders on how to design better sustainable software systems through the sustainability guidelines. Second, FSSSD, a derivative from SSDC, serves as a guide and support for stakeholders during the software design by using sustainability goals, aided by different sustainability concepts, methods and tools to facilitate software sustainability by design.
- **Measurement**: Publication 3 presents several measures advocating software sustainability and green software, based on the four software sustainability perceptions (sustainability in software development, software for sustainability, green software systems and Sustainability of software ecosystems), to support stakeholders during the integration and testing SDLC phases. The measures presented show different practices currently used in software sustainability evaluation. This measures can be used for software sustainability measurements during the integration and testing phases.

Documentation: This stage covers documentation during SDLC phases. Publication 4 presents research on collecting and disseminating software sustainability requirement elicitation best practices using a Template. Feedback from stakeholders in Publication 4 shows the template can serve as a useful tool in recording best sustainability practices during software design and development.

- **Validation**: Publication 5 provides results from case studies for the validation of FSSSD, based on the foundation laid in Publications 1- 4. The outcome of FSSSD in case studies shows that stakeholders need different tools to guide and support them during software sustainability design, development and measurement. The early feedback from the case studies highlights the usefulness of Framework for Sustainability of Software System Design, because it persuaded stakeholders to rethink their software development project with regards to sustainability. The feedback also indicates a challenge for those interested in software sustainability design. One of the major challenge is the lack of a central repository where sustainability has been exemplified in different software designs for stakeholders to learn and improve their understanding of sustainability during software design, development and measurement.

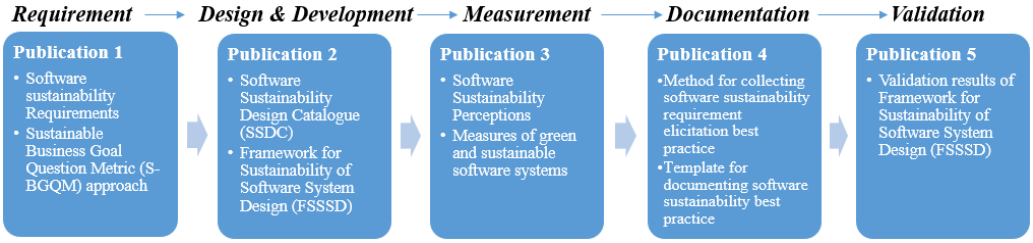


Figure 2. Overview of research contribution

2 Background

This chapter addresses the overall background research on software sustainability design, development and measurement. The general key concepts of this thesis, such as the meaning and definition of sustainability, sustainability pillars, software sustainability dimensions and sustainability in SE, are presented in this chapter.

The key research contributions in the field of software sustainability design from different authors and challenges of sustainability in software design, development and measurement are also detailed in this chapter. Overall, this chapter describes the research gap in software sustainability design and development, with a research statement on areas that require additional research.

2.1 Sustainability

Sustainability is a concept with a particular characteristic: that is, its meaning depends strongly on the context of the application (Becker, 2014). Sustainability has become a popular concept, with values expressed in research, academia, industry and government (Wolfram, Lago and Osborne, 2017). Today, sustainability values from the environmental, economic, and social dimensions such as healthy environment, vibrant economy and equitable society are those we all aspire to achieve. We aim to do so through policies, infrastructure, technological artefacts, systems, social and cultural development, human welfare and community building.

Sustainability is based on the premise that everything humans require for their survival and well-being depends, either directly or indirectly, on the natural environment (Marsh, 1864; National Research Council, 2011). Sustainability is regularly expressed as how the biological system endures and remains diverse and productive, but in the 21st century, sustainability now refers to the need to develop sustainable models necessary for both the human race and planet Earth to survive. In 2000, the Earth Charter stated that sustainability is a global society founded on respect for nature, universal human rights, economic justice and a culture of peace. Sustainability has gained worldwide recognition because of the following elements (Degrees, 2019):

- **The need for conservation and energy**: Advances and growth in economies and energy came at the cost of environmental degradation. This has led to different initiatives towards how to slow or prevent pollution, conserve natural resources and protect the environment.
- **Developing and maintaining a sustainable society**: Sustainable society is based on equal access to health care, nutrition, clean water, shelter, education, energy, economic opportunities and employment, the pursuit of quality life, social justice for all and harmony with the natural environment.
- **Supporting sustainable business:** Business patterns that require long-term practices that encourage respect for the environment, welfare and well-being of

employees, improved profitability, reduced costs, create innovation and increase market share.

- **Advances in sustainable technology and development:** The pervasive nature of technological advancement also brings the challenge of adverse effects on sustainability. There is a need to position new technologies with rural and urban infrastructure grounded around environmentally sound practices to support a sustainable, healthy and happy population.
- **Investigating climate change:** The way we live, produce and use natural resources has negatively impacted climate change. Debates, discussion and research are occurring worldwide regarding government policies on how we live, produce and use natural resources, and also the necessary corporate and individual actions for positive climate change.

Sustainability is now a worldwide goal for our planet because of the continuous degradation and depletion of natural resources, particularly the resources required for human existence, good health and good quality of life. The reasons for sustainability are now recognised worldwide and show the importance of sustainability in all aspects of our lives.

2.1.1 Brief History of Sustainability and Sustainability Definitions

The word 'sustainability' was originally coined from forestry and it means never harvesting more than what the forest yields in new growth (Wiersum, 1995). The first use of sustainability as a word in the European context was in 1713, in the book *Sylvicultura Oeconomica* by German forester and scientist von Carlowitz, used as Nachhaltigkeit (German language) which means sustainability (Heinberg, 2010; Kuhlman and Farrington, 2010). According to Heinberg (Heinberg, 2010), sustainability is a relative term that can be used as a frame of reference for the duration of prior civilisations, ranging from hundreds to thousands of years. Sustainability became a widely used term after the Brundtland Report from the United Nations World Commission on Environment and Development and its definition of sustainable development (UN General Assembly, 1987). The Brundtland report defined sustainable development as "development that meets the needs of the present without compromising the ability of future generations to meet their own needs." (UN General Assembly, 1987). This definition of sustainable development highlights the long-term characteristics of sustainability and ethical responsibility for fairness between present and future generation. Sustainability is however different from sustainable development because sustainability is a foundational concept for sustainable development and the sustainable development goals (Diesendorf, 2000).

Sustainability as a concept has been defined in different ways to ensure equality, quality of life, a safe environment free from toxic pollution, and continuous human existence in peace and harmony (Ben-Eli, 2015). The following are some definitions of sustainability:

1. **Sustainability** is a vision for the world in which current and future humans are reasonably healthy; communities and nations are secure, peaceful and thriving; there is economic opportunity for all; and the integrity of the life-supporting biosphere is restored and sustained at a level necessary to make these goals possible (Cortese and Rowe, 2000).
2. **Sustainability** is a dynamic equilibrium in the process of interaction between a population and the carrying capacity of its environment, such that the population develops to express its full potential without producing irreversible, adverse effects on the carrying capacity of the environment upon which it depends (Ben-Eli, 2015).
3. **Sustainability** is the long-term viability of a community, set of social institutions, or societal practice (Britannica.com). Here, community refers to people with common interests living in a particular area.
4. **Sustainability** is also defined as the ability to continue a defined behaviour indefinitely with consideration of the environment, society and economy (Thwink.org, 2019).

The common aspect from these definitions of sustainability shows people are core part for achieving sustainability as they form a society. These definitions support economic prosperity for all and healthy environment for continuous growth, provide values and goals that every society should have to achieve a good quality of life for all living species and ensure harmony among them. In order to continue to live as a society for current and future generations, sustainability values from the social, economic and environmental are required to ensure human evolution does not lead to depletion of resources.

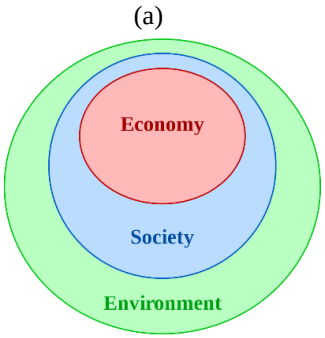
2.1.2 Sustainability Pillars

The Brundtland report of the World Commission on Environment and Development (‘Our Common Future’); (UN General Assembly, 1987), the European Information Technology Observatory in 2002 (EITO, 2002) and the World Summit on Social Development in 2005 (United Nations, 2005) identified three major areas as the core of sustainability development, namely, economic development, social development and environmental protection. These three pillars (Figure 3 and Figure 4) formed the corner points for different research efforts towards sustainability in different disciplines, including SE.

- **Economic pillar**: This means preserving and increasing economic capital without negative impact on the social and environmental pillars (UN General Assembly,

1987; EITO, 2002;United Nations, 2005). Protecting business investment and ensuring that business activities support viable business practices to aid collective equity are goals of the economic pillar.

- **Environmental pillar**: Promoting activities that will minimise negative impacts on the environment through operational efficiency and safeguarding natural resources from depletion (UN General Assembly, 1987; EITO, 2002; United Nations, 2005). The goal of the environmental pillar is preserving the earth's resources so humans can survive and evolve, with prosperity for current and future generations.
- **Social pillar**: Promoting social equity, trust and harmony among all living species (UN General Assembly, 1987; EITO, 2002; United Nations, 2005). The social pillar goals support community building on fairness, justice, good quality of life, security, health and continuous access to resources, irrespective of social class.



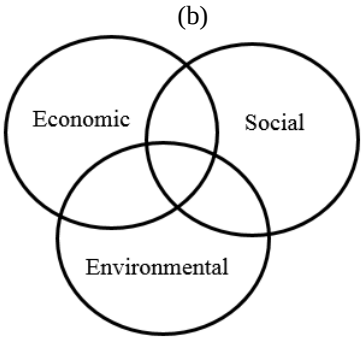


Figure 3. (a) Nested view of sustainability pillars. Figure 4. (b) Venn diagram of sustainability pillars

The three interdependent and jointly reinforcing pillars of sustainability are commonly called the 'triple bottom line' (EITO, 2002), a term coined by John Elkington in 1997 to set economic, social and environmental performance goals and objectives. The nested view of sustainability pillars (Figure 3) and Venn diagram (Figure 4) can be interpreted as ignoring the intrinsic, immutable relationships existing between each of the pillars. By characterising the pillars as independent systems, the model falls into a reductionist epistemological trap which fails to account for the inherent interactions between "the parts, the whole." Addressing issues associated with each pillar in isolation will lead to prioritising one pillar over the other (Moir and Carter, 2012).

Extending the pillars of sustainability, Goodland (Goodland, 2002) presents the types of sustainability as human (maintaining human capital, such as health, education and access to services), social, economic, and environmental. Linking sustainability to software systems (Penzenstadler and Femmer, 2013) argue that sustainability dimensions are individual, social, economic, environmental and technical. The additional technical dimension, suggested by (Penzenstadler and Femmer, 2013), offers support for the long-

term evolution of technical systems. Section 2.2 covers more details of the sustainability dimensions.

In today's information age, where software has the potential to drive most SDGs with example of infrastructure/medical diagnosis software and sustainable development influences ICT policies for software systems (EITO, 2002), there is a need for research on understanding how sustainability can be a core part of software design and development. The SDGs namely: No Poverty, Zero Hunger, Good Health and Well-being, Quality Education, Gender Equality, Clean Water and Sanitation, Affordable and Clean Energy, Decent Work and Economic Growth, Industry, Innovation, and Infrastructure, Reducing Inequality, Sustainable Cities and Communities, Responsible Consumption and Production, Climate Action, Life Below Water, Life On Land, Peace, Justice, and Strong Institutions, Partnerships for the Goals (United Nations, 2015) needs more research on how ICT can support international cooperations achieve sustainable development with the use of software system (Wu *et al.*, 2018). Figure 5 summarises some of the most important agreements and treaties for sustainable development and technological policy convergence.

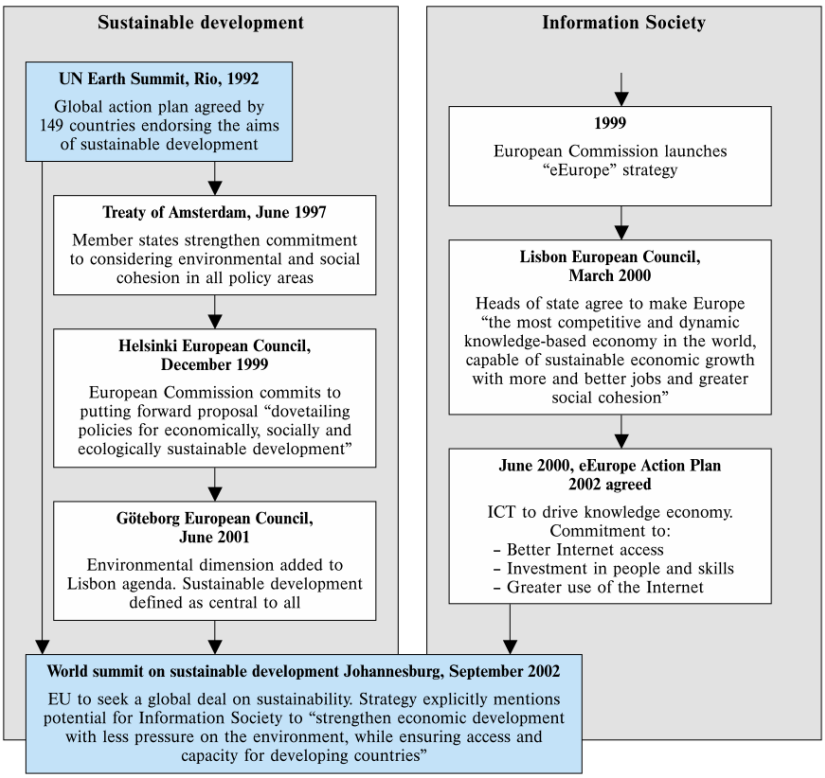


Figure 5. Sustainable development and technological policies convergence (EITO, 2002)

2.2 Sustainability in Software Engineering

Research on sustainability in SE has evolved with different research efforts from the requirements, design, development and measurement of software systems (Becker *et al.*, 2016 ;Wolfram, Lago and Osborne, 2017). Some key challenges of sustainability in SE include harmonising multiple research efforts into a central focal point to support and guide stakeholders interested in software sustainability design guidelines (Becker *et al.*, 2015), development (Wolfram, Lago and Osborne, 2017) and measurement (Albertao *et al.*, 2010; Bozzelli, Gu and Lago, 2013). This section covers software sustainability definitions, sustainability dimensions, research work on software sustainability design, development and measurement.

2.2.1 Software Sustainability Dimensions and Definitions

In SE, sustainability is categorised into five dimensions, namely, economic, environmental, social, individual and technical (Penzenstadler and Femmer, 2013), extending the three main pillars of sustainability (United Nations, 2005).

- **Economic sustainability** is about maintaining financial capital, assets and added value towards financial growth. For SE, the focus is on how to design and develop software systems in a cost-effective manner and ensure the safety of the stakeholders' long- and short-term investment from economic risk.
- **Individual sustainability** refers to the maintenance of individual human capital, human dignity, health, education and equal access to services. In the context of this research the individual dimension focuses on software architects and developers. For SE, individual sustainability means, 'How can software be created and maintained in a way that enables developers to be satisfied with their job over a long period?'
- **Social sustainability** is about the relationship between individuals, groups and maintaining social capital; the mutual trust structure in the societal communities; and the balance between conflicting interests. For SE, the main question is, 'What are the impacts of software systems and applications on the society?' (Example: communication, sense of belonging, interaction and equality).
- **Environmental sustainability** refers to the use and maintenance of natural resources, such as water, land, air, minerals and the ecosystem to improve the welfare of all living creatures (humans and animals). The environmental dimension is to ensure ecological integrity in which there is a balance in how natural resources are produced and used at a rate in which they can replenish themselves (Giovannoni and Fabietti, 2013). For SE, the question is 'How does software impact and affect the environment during / after development and maintenance?'

- **Technical sustainability** covers the fundamental goal of long-time usage of systems and their suitable evolution along with changing user requirements and environments. It is about the maintainability and evolution of systems over time. For SE, the question is 'How can software be designed and developed for easy evolution, maintainability, adaptability to changes in the future'?

The different dimensions of sustainability offer the means to decompose sustainability values for the engineering of software systems design, and also serve as means for categorisation during the evaluation of effective sustainability design and development, using analyses of the first, second and third order impacts to create a sustainable software system. This order of impacts are explained as follows (Erdmann *et al.*, 2004):

- First order impacts (immediate effects) are about the direct effects of the development and use of a software system.
- Second order impacts (enabling effects) are about the indirect impacts related to the effects of using the software system in its application domain.
- Third order impacts (structural effects) are the cumulative long-term effects resulting from accumulating first and second order impacts over time.

The meaning and understanding of 'sustainable software' varies in SE, based on the different domains of application and the stakeholders involved in the application; some consider technical sustainability, while others consider the higher impacts of software systems. Table 2 shows some software sustainability definitions and their corresponding sustainability dimensions.

Table 2. Software sustainability definitions

Author	**Definition**	**Sustainability Dimensions**
(Naumann *et al.*, 2011)	Sustainable software is a software in which the direct and indirect negative impacts on the economy, society, human beings and environments that result from development, deployment and usage of software are minimal or has a positive effect on sustainable development.	Environmental, technical, economic, social and individual
(Koziolek, 2011)	A software-intensive system is long-living if it must be operated for more than 15 years.	Technical
(Koziolek, 2011)	A long-living software system is sustainable if it can be cost-efficiently maintained and evolved over its entire life-cycle.	Economic, Technical

(Calero, Moraga and Bertoa, 2013)	Sustainability of a software product can be defined as the capacity of developing a software product in a sustainable manner	Environmental, Social, Technical
Seacord *et al.* (Seacord *et al.*, 2013)	Software sustainability, is the 'ability to modify software system based on customer needs and deploy those modifications,' which means software sustainability is the ability to modify systems based on user requirements.	Social, individual and technical
(Idio, 2014)	• Long-lasting software that relates to how well a piece of software will be able to cope with changes • Lean software that requires less hardware and reduces its power consumption (energy efficient) • Software for sustainable humans as software that induces sustainable human behaviour.	Technical, environmental, individual

The above definitions show different perspectives of software sustainability in SE in the design, development, maintenance and usage phases. According to Venters *et al.* (Venters *et al.*, 2014), there are four aspects to consider when supporting sustainability in SE:

1. Development process: The use of environmental, human and capital resources
2. Maintenance process: Continuous monitoring of quality and knowledge management
3. System production: Dedicated to the way resources are used during production activities to achieve system development goals
4. System usage: Consider the responsibility for environmental impact.

The use of sustainability dimensions as a key part of the aspects listed above includes encouragement of better thinking about how to incorporate sustainability into a software system along with open discussions on how each sustainability dimension should be treated during the development, maintenance processes, system production and system usage.

2.2.2 Sustainability in Software Design and Development

Sustainability has gained increasing attention in software design and development, especially from the requirement engineering domain (Mahaux and Canon, 2012; Penzenstadler, 2014; Oyedeji, Seffah and Penzenstadler, 2017) and other research topics (Ehrenfeld, 2008; Chitchyan *et al.*, 2015; Robillard, 2016; Spinellis, 2017). Requirements engineering, as a major phase of software system development, has an important role to

play in software sustainability design. The International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy) (Penzenstadler, Mahaux and Salinesi, 2014; Penzenstadler, Mahaux and Salinesi, 2015; Penzenstadler, Salinesi and Ruzanna, 2017) provided an anchor point for many researchers on software sustainability through the lens of requirements engineering.

One of the major drivers for sustainability during requirements engineering is the ability to discuss the benefits of sustainability for both end users and all stakeholders involved. The current requirements engineering methods and tools do not facilitate negotiation or discussion about sustainability requirements during software development, which usually leads to the omission of sustainability or consideration of only one dimension during the software design (Seyff *et al.*, 2018). Seyff and colleagues (2018) presented an adaptation of the EasyWinWin method and WinWin Negotiation Model to facilitate and stimulate negotiation among stakeholders and requirements engineers for sustainability requirements in software systems. Seyff and colleagues detected how each requirement affects each sustainability dimension: first order impacts (immediate), second order impacts (enabling) and third order impacts (structural). This can help support sustainability consideration during software requirement. However, there is a need for education and awareness to improve practitioners' knowledge about the concept of software sustainability by design in the professional environment, such as an understanding of software sustainability by design and the potential of applying sustainability in requirements and SE. Crowd-focused requirements engineering was used by Seyff *et al.* (2018) to support the evolution of software sustainability requirements to improve the awareness and understanding of sustainability in requirement engineering for researchers and interested stakeholders. This is one way of improving sustainability awareness among stakeholders in the requirements engineering domain.

The results of a study of requirements engineering practitioners shows that the attitudes and perceptions of software practitioners regarding sustainability are limited due to a narrow understanding of sustainability, poor organisational awareness about the positive opportunities for applying sustainability and the benefits that can be generated from it (Chitchyan *et al.*, 2016). Furthermore, another major challenge of sustainability in software requirements engineering is that there is no single reference point where different research on the application of sustainability in software requirement are gathered and exemplified (Chitchyan *et al.*, 2015), to support and guide requirement engineers on how to effectively elicit software sustainability requirements during software design and development. The Karlskrona Manifesto for Sustainability Design (KMSD) principles provides the basis for creating a reference point that can be applied during software design by different stakeholders (Becker *et al.*, 2015). These are the nine principles of KMSD:

1. **Sustainability is systemic**: Sustainability is never an isolated property. It requires transdisciplinary common ground of sustainability as well as a global picture of sustainability within other properties.

2. **Sustainability has multiple dimensions**: We have to include different dimensions into our analysis if we are to understand the nature of sustainability in any given situation.
3. **Sustainability transcends multiple disciplines**: Working in sustainability means working with people from across many disciplines, addressing the challenges from multiple perspectives.
4. **Sustainability is a concern independent of the purpose of the system**: Sustainability has to be considered even if the primary focus of the system under design is not sustainability.
5. **Sustainability applies to both a system and its wider contexts**: There are at least two spheres to consider in system design: the sustainability of the system itself and how it affects the sustainability of the wider system of which it will be part.
6. **System visibility is a necessary precondition and enabler for sustainability design:** Strive to make the status of the system and its context visible at different levels of abstraction and perspectives to enable participation and informed responsible choice.
7. **Sustainability requires action on multiple levels:** Seek interventions that have the most leverage on a system and consider the opportunity costs: whenever you are taking action towards sustainability, consider whether this is the most effective way of intervening in comparison to alternative actions (leverage points).
8. **Sustainability requires meeting the needs of future generations without compromising the prosperity of the current generation:** Innovation in sustainability can play out as decoupling present and future needs. By moving away from the language of conflict and the trade-off mindset, we can identify and enact choices that benefit both present and future.
9. **Sustainability requires long-term thinking**: Multiple timescales, including longer-term indicators in assessment and decisions, should be considered.

The Karlskrona Manifesto principles are driver for a broader discussion about sustainability in software design (Becker *et al.*, 2015) for different stakeholders (software practitioners, researchers, professional associations, educators, customers and users) in the SE community and industry to consider the different sustainability dimensions during software requirements engineering and development (Penzenstadler, 2015). The KMSD shows design is a big part of achieving software sustainability and also a key element in software sustainability design and development (Oyedeji, Seffah and Penzenstadler, 2018a;). Software design, as a key aspect of software development, can help to reduce energy consumption by 30% to 90% because software dictates what and how hardware functions, meaning that software can support real energy savings in any software system (Musthaler, 2014).

However, there is currently limited research on software sustainability design, even though design is the central phase of any software development (Freeman, 1980; Tate, 2005). The methods applied and practices in design and usage of a software system have a significant effect on sustainability in SE and can have a major influence on the users' perception of sustainability (Mahaux, Heymans and Saval, 2011). Software practitioners and stakeholders should identify different leverage areas for a better understanding of how software can act as a catalyst for transformational change towards sustainability (Penzenstadler and Venters, 2018). Leverage points in software systems, in which a change in one aspect of the software can positively impact the whole system's sustainability, can help software engineers to address issues of sustainability in a software system (Penzenstadler *et al.*, 2018).

A study of software design and development life cycle activity focusing on protection of the environment proposed a formula that can assist software architects and developers calculate software waste in order to promote the design and development of green software. The use of the proposed formula will aid computational and data efficiency (Erdélyi, 2013). Venters et al. (2018) presented some issues of sustainability in software architecture design, such as sustainability debt, cumulative effects of flawed architectural design choices over time, resulting in code smells, architectural brittleness, erosion, coupling and cohesion issues. The authors' work (Venters *et al.*, 2018) also provides a roadmap for open research challenges and issues in sustainable software architectures.

In addition, the concept of sustainability for software design and its integration into the existing catalogue of design quality attributes are needed to achieve sustainable software (Robillard, 2016) and sustainability should also be considered as a quality of software systems like security and usability (Lago *et al.*, 2015). This consideration will require a multidimensional and interdisciplinary approach (Chitchyan *et al.*, 2016; Penzenstadler, Tomlinson and Richardson, 2012; Bozzelli, Gu and Lago, 2013). Research work on formalising the design of sustainability into software systems based on the five sustainability dimensions is needed to develop official standards and models of sustainability in software design, development and measurement (Wolfram, Lago and Osborne, 2017).

Overall, this section describes some of the key challenges and problems of sustainability in software design and development. Based on current research, one of the major challenge of sustainability in SE and application of sustainability in a software project is lack of understanding about what sustainability means in software design and development. Another challenge is the lack of awareness, especially among practitioners about the benefits of sustainability in software design and how to formalise different dimensions of sustainability into software sustainability design and development.

Addressing these challenges will facilitate better software sustainability requirements, design and development from the different sustainability dimensions, which can improve the negative effects and impacts on software systems. This thesis uses the following research work as a building block to address these challenges: the KMSD (Becker *et al.*,

2015), requirements engineering design methods for software sustainability systems design and development (Penzenstadler, 2014; Chitchyan *et al.*, 2015; Penzenstadler, 2016).

2.2.3 Software Sustainability Measures and Measurement

Research on software sustainability measures and measurement is an area currently with limited research. There is the challenge for guidance on what sustainability measures can be effectively used in the measurement and evaluation of software systems considering the five sustainability dimensions (Albertao *et al.*, 2010;Calero, Bertoa and Angeles Moraga, 2013). Currently, there are few studies about 'what' aspect of software sustainability to measure and 'how' to measure it efficiently (Lami and Buglione, 2012). The lack of understanding of what and how to measure software sustainability has limited the complete adoption of sustainability in most software design and development projects. Another challenge of software sustainability measurement is that the management and planning of software sustainment are affected by the lack of consistently applied practical software sustainability measures (Seacord *et al.*, 2013).

There is need for software sustainability measures that addresses the different sustainability dimensions and software quality. In this regard, Albertao *et al.* (Albertao *et al.*, 2010) suggested the use of existing software quality attributes and measures as an indirect way to evaluate the economic, social and environmental sustainability of software projects. Gordieiev and colleagues (Gordieiev, Kharchenko and Fusani, 2016) proposed the use of measures associated with other software quality attributes as a way of evaluating green software. Some of the measures used in the evaluation of software sustainability and green software systems are detailed in Table 3.

Table 3. Software sustainability measures

Measure	**Description**	**Sustainability Dimension**
Energy efficiency (EF) (Johann *et al.*, 2012)	*EF*=UsefulWorkDone/UsedEnergy, where Useful Work Done is the total amount of completed task by a software module, and Used Energy is the total amount of energy used (Joule) in the process of completing the task	Technical, environment
Functional suitability (FS) (Gordieiev, Kharchenko and Fusani, 2016); Albertao *et al.*, 2010)	*FS* is measured using computational accuracy (CA), where CA is the total number of frequency of all inaccurate results based on user operation FS = A / T A= Number of cases encountered by users with a difference against reasonably expected results beyond allowable T= Operation time	Technical

Performance efficiency (PE) (Gordieiev, Kharchenko and Fusani, 2016)Albertao *et al.*, 2010)	PE is divided into the following: Time behaviour • *Response Time*: The amount of time taken to complete a task T = (time of gaining the result) - (time of command entry finished) • *Response Time Mean (RTM) -Mean Time:* the mean response time of the software system to finish a task or request. RTM= Tmean / TXmean Tmean = ∑(Ti) / N, (for i=1 to N) TXmean = required mean response time Ti= response time for i-th evaluation (shot) N= number of evaluations (sampled shots)	Technical, environmental, economic
Power usage effectiveness (PUE) (Rondeau, Lepage and Georges, 2015)	PUE = Total Facility Energy/IT equipment Energy where the Total Facility Energy and IT equipment Energy is measured in watts and converted to Joule	Environmental, technical
Maintainability (Gordieiev, Kharchenko and Fusani, 2016)Albertao *et al.*, 2010)	Analysability • Diagnostic function support X= A / B A= Number of failures which maintainer can diagnose (using the diagnostics function) to understand the cause-effect relationship B= Total number of registered failures • Failure analysis capability X=1- A / B A= Number of failures of which causes are still not found B= Total number of registered failures Testability • Availability of built-in test function X= A / B A= Number of cases in which maintainer can use the suitably built-in test function B= Number of cases of test opportunities • Re-test efficiency X= Sum(T) / N	Environmental, technical, economic

	T= Time spent to test to make sure whether the reported failure was resolved or not N= Number of resolved failures	
Software energy cost (Noureddine *et al.*, 2012)	Esoftware = Ecomp +Ecom +Einfra, where Ecomp is the computational cost (i.e. CPU processing, memory access, I/O operations). Ecom is the cost of exchanging data over the network, and Einfra is the additional cost incurred by the OS and runtime platform (e.g., Java VM)	Technical
Resource usage (Koçak, Alptekin and Bener, 2014)	The amount of CPU Usage, I/O Usage, Memory Usage, Storage Usage for completing a software task	Technical
Energy impact (Koçak, Alptekin and Bener, 2014)	Energy impact is the total energy consumption and the CO_2 emission based on the energy usage	Technical, Environmental
Energy efficiency (Speedup Greenup, Powerup, and) (Abdulsalam *et al.*, 2015)	Speedup=Tφ/To where Tφ is the total execution time of non-optimised code, and To is the total execution time of the soptimised code Greenup = Eel/Eon Assuming Pφ is the average power consumed by the non-optimised code and Po is the average power consumed by the optimised code Powerup =Po /Pφ= Speedup /Greenup	Environmental, Technical
Software Project's Footprint (Albertao *et al.*, 2010)	The effect of the number of resources used in software development projects, such as power, electricity, computers, fuel consumption for transportation, emissions and human resources	Economic, Social, Environmental

The measures proposed and used in the measurement of software sustainability (Table 3) shows that energy and energy efficiency has received the largest attention for measures applied in measuring and evaluating software sustainability. This is further highlighted by research compilation on different software sustainability measures and measures for green software (Bozzelli, Gu and Lago, 2013; Kern *et al.*, 2013). This might be due to the research attention green software has gained over the past ten years and the need to reduce energy cost for many companies (Oyedeji, Seffah and Penzenstadler, 2018b).

The previously referenced works have focused on specific dimensions of sustainability. In order to have a holistic measurement of software sustainability, there is a need to provide methodological frameworks and methods to create new measures of software sustainability. Such frameworks and methods should take into account the five

dimensions of sustainability (economic, social, individual, environment and technical) with the capability to extend current software measures.

3 Methodology

This chapter presents an overview of the research methods and processes applied in this research. A general description of the other research methods considered is provided with full details of design science, including the guidelines for design science methodology, design science research cycles and processes, case study research methods and components of case study research design. Details of how design science and case study research methods are applied in the publications are presented in this chapter.

Design science research methodology (Hevner *et al.*, 2004) was applied in this thesis because the aim of this research is to create new artifacts to improve problems of sustainability in software design and contribute new knowledge to software engineering practices. It also aims to improve the applicability of existing artifacts such as Karlskrona manifesto principles to solve identified problems during software sustainability design. Case study (Starman, 2013) was used in the demonstration and evaluation of artifacts created in this thesis.

3.1 Selection of Research Methods

The following are the research methods considered at the beginning of this thesis research with rational for selecting design science and case study.

Grounded Theory: This methodology originated with the work of Glaser and Strauss on the interactions between health care professionals and dying patients. It is define as the discovery of theory from data systematically obtained from social research (Glaser and Strauss, 1967). The main feature in grounded theory is the development of new theory through the collection and analysis of data about a phenomenon. However in this research the main aim is to create new artifacts and not to derive new theory which means grounded is not appropriate for this research.

Action Research: Action research is an emergent inquiry process that integrates theory and action to couple scientific knowledge with existing organizational knowledge and to address real organizational problems together with the people of the system under inquiry (Mohajan, 2018). It seeks transformative change through the simultaneous process of taking action and doing research, which are linked together by critical reflection. Action research is also a systematic and orientated around analysis of data whose answers require the gathering and analysis of data and the generation of interpretations directly tested in the field of action (Macdonald, 2012).This research method would have been suitable for this research, but this thesis was not based on active collaboration with companies where the researcher collaborate with participants (company staffs such as software developers and architects) to improve practice of software sustainability design in the companies. Rather artefacts were created within this thesis, evaluated and tested in companies using case study. Thus, action research was not applied in this thesis work.

Design Science Research: Design science research is a research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artefacts, thereby contributing new knowledge to the body of scientific evidence (Hevner and Chatterjee, 2010). This research method was selected because it provides research processes that best suit the research questions in this thesis and it supports the creation of artifacts that can be used to improve stakeholders (software architects, developers, requirement engineers, researchers and companies) understanding of sustainability in software design. Design science also support design iteration of artifacts to improve artefacts usability in tackling the research problems.

Design science research seeks to create innovations that define the ideas, practices, technical capabilities and products through which the analysis, design, implementation, and use of information systems can be effectively and efficiently accomplished (Hevner *et al.*, 2004). This research method is centred on the improvement of designed artefacts functional performance to solve identified problems. Table 4 details the design science research guidelines (Hevner *et al.*, 2004) applied in this thesis.

Table 4. Design science research guidelines adopted from (Hevner *et al.*, 2004)

Guideline	**Description**
Design as an Artefact	Design science research must produce a viable artefact in the form of a construct, a model, a method or an instantiation.
Problem Relevance	The objective of design science research is to develop technology-based solutions to important and relevant business problems.
Design Evaluation	The utility, quality and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.
Research Contributions	Effective design science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations and design methodologies.
Research Rigour	Design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.
Design as a Search Process	The search for an effective artefact requires utilising available means to reach desired ends while satisfying laws in the problem environment.
Communication of Research Design science	Design science research must be presented effectively to both technology-oriented as well as management-oriented audiences.

Case Study: A case study is an in-depth exploration from multiple perspectives of the complexity and uniqueness of a particular project, policy, institution, programme or system in real life (Starman, 2013). Case study was used in the validation of artifacts created within this research because it takes into account the context of the subject under investigation and requires the formation of questions in terms of ‘who’, ‘what’, ‘where’,

'how', and 'why' during the research investigation. For evaluating the artefacts created through the design science process, the method of a case study offers in-depth content that provides a complete picture for the whole situation. The goal of this thesis is to support and guide stakeholders in software sustainability design and development, case study allows for collecting feedback from natural setting and context which in this case is a company and university where software is designed and developed. The two case studies used allow discovery of the real problems in adopting sustainability during software design and development and explore means for solving those problems.

3.1.1 Design Science Research Cycles

The research work of (Hevner *et al.*, 2004) on design science did not include a detailed process for performing design science research. A new method, proposed in 2007, is called the design science research cycles. This new process is based on the information system (IS) research framework (Hevner *et al.*, 2004). It has overlays of three inherent research cycles (Relevance Cycle, Rigour Cycle and Design Cycle) for any design science research work (Hevner and Chatterjee, 2010).

The Relevance Cycle links the contextual environment of the research project to design science activities. The Rigour Cycle links the design science activities with the knowledge base of scientific foundations and finally, the Design Cycle iterates between the core activities of building and evaluating the design artefacts and the processes involve in the research. Figure 6 shows the refined design science cycles as a guide for carrying out any design science research project.

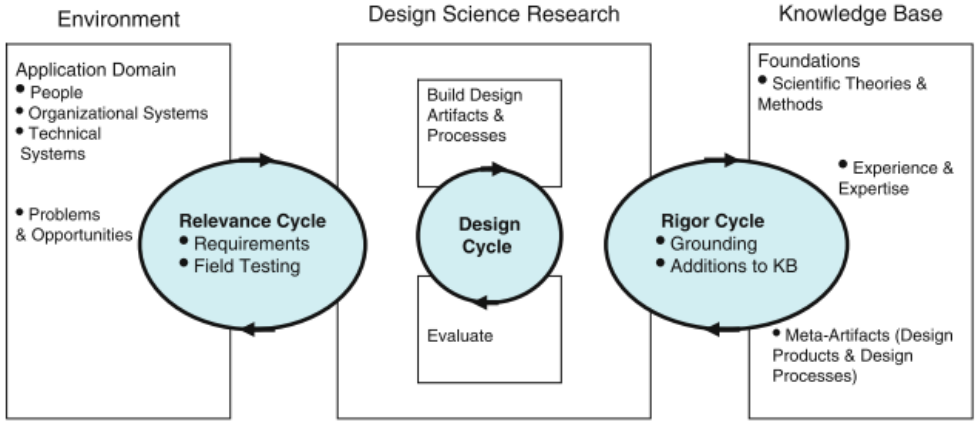


Figure 6. Design science research cycles (Hevner and Chatterjee, 2010)

Table 5 details how the design science research process and cycles were applied in this thesis, from the problem identification and motivation to the communication process.

Table 5. Design Science Research Process and Cycles Applied in Thesis

1. What is the research question? *Phase: Identification of problem and motivation (Relevance Cycle)*	The research question for this thesis is stated in Chapter 1.1.
2. What are the objectives of the solution? *Phase: Define objectives of solution (Relevance Cycle)*	The objectives are stated in Chapter 1 and also in Publications 1 and 2.
3. What is the designed artefact? *Phase: Design and development (Design Cycle)*	The artefacts created in this thesis are S-BGQM (Publication 1), SSDC and FSSSD (Publication 2), and Template for Software Sustainability Requirement Elicitation Best Practice (Publication 4). Details of the artefacts are in Chapter 4.
4. What were the design processes used in creating the artefacts? *Phase: Design and development (design cycle)*	After identification of the problems for software sustainability design through literature reviews and discussion with stakeholders in the industry, the first artefact was created in the first iteration (Publication 1) through an iterative design process; the second and third artefacts were designed (Publication 2). The fourth artefact was created after feedback from stakeholders on the second and third artefacts (Publication 4).
4. How was the artefact used in the application environment to test the usefulness? *Phase: Demonstration (Relevance cycle)*	The artefacts were used in case studies (Publication 5) and tested with industry experts (Publication 4).
5. How were the design process and artefacts grounded in the knowledge base? *Phase: Evaluation (Linking Design to Rigour Cycle)*	The design process for the artefacts was grounded in the KMSD (Becker *et al.*, 2015) and requirements engineering design methods for software sustainability systems design and development (Penzenstadler, 2014).
6. How were the artefacts evaluated? *Phase: Evaluation (Design Cycle)*	The artefacts were used in two case studies: one case study with sustainability as a principal factor, and the other without sustainability consideration. The purpose was to see if the artefact would guide and support stakeholders to consider sustainability in their software design and development project. Publication 5 describes the positive results and some of the identified areas to improve the artefact.
7. What new contribution is added to the knowledge base, informed by theory, method or literature? *Phase: Communication (Rigor cycle)*	New artefacts to support and guide software sustainability design and development have been created. Furthermore, the identification of new research gaps, detailed in Chapter 1 (introduction) and Chapter 2 (background) as problems, challenges of software

	sustainability design and discussion in Chapter 5.
8. Did this thesis address all research questions? *Phase: Complete review and evaluation of the research process (Relevance cycle)*	All research questions have been satisfactorily addressed, as detailed in Chapter 5 and Publications 1-5.
How has the process for designing, using and evaluating the artefacts been communicated? *Phase: Communicating design, application and evaluation results of artefacts. (Rigor Cycle)*	All the processes involved in the design, usage and evaluation of the artefacts have been published in conferences and journal (Publications 1-5).

3.1.2 Design Science Research Processes

This thesis is grounded in design science research (Hevner *et al.*, 2004) and design science research methodology (DSRM). Peffers *et al.* (2007) proposed the use of a six-step guide process during design science research. Figure 7 shows an overview of the six-step guide for design science research.

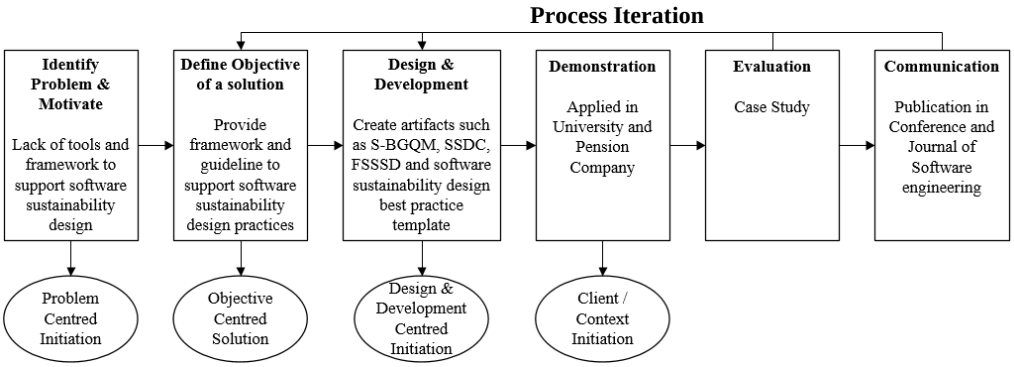


Figure 7. Modified DSRM process model adopted from (Peffers *et al.*, 2007)

1. **Identify problem and motivate**: This step involves defining the specific research problem and challenge in a clear way and justify the value of finding a solution to the problem. Publication 1 investigates different kinds of literature from SE and other similar domains to identify the current problems and research gaps of software sustainability design, development and measurement. The outcome from Publication 1 led to the definition of the main research question and the sub-research questions (RQ1, RQ2 and RQ3) in Chapter 1. Publication 1 details the problems and challenges of software sustainability requirement elicitation, design and development.

2. **Define objectives of the solution**: The main purpose and goals of the solutions to the identified problems are stated base on the current understanding of what is achievable. In this research, based on the identified problems from the first step which form the basis of the research questions, Publication 2 details the objectives for creating guidelines and framework to support software sustainability design, development and measurement of software sustainability.
3. **Design and development:** This stage involves the creation of artefacts, such as methods, constructs, theories and models. The first artefact in this thesis is the Sustainable Business Goal Question Metric (S-BGQM) approach to support the software sustainability requirements elicitation from Publication 1. From Publication 2, the second artefact, SSDC, is proposed to show how sustainability can be applied to different software systems. The third artefact, FSSSD, was created to be used in software design by software architects, developers and companies during software design and development. Finally, in Publication 4, the last artefact, a template to document best practices for the design of software sustainability, is created to ensure effective documentation of all best practices that will serve as guide for others interested in software sustainability design.
4. **Demonstration:** This step is about using the artefact to solve the problems mentioned above. This can be demonstrated through case studies, experiments or any activity that shows the application of the artefact. The demonstration of the proposed artefacts is described in Publication 5, regarding a validation study in which the artefacts were applied to case studies for different types of software systems to test effectiveness in solving problems of software sustainability design, development and measurement. Publication 4 also demonstrates the use of the best practice template for documenting software sustainability requirement elicitation.
5. **Evaluation:** The results from applying the artefacts to solve the identified problems were assessed to see how well the artefacts met the set objectives of the proposed solution by comparing the result of using the artefacts to the set objectives. Results and feedback from applying the S-BGQM approach to software design and development projects led to the design of new artefacts, such as SSDC and FSSSD, for better software sustainability design. Results from using SSDC and FSSSD are in Publication 5; the feedback is detailed in Chapter 5 under section 5.1 research evolution and results.
6. **Communication:** The documentation of the problem identification, relevance, artefacts, artefacts usefulness and effectiveness in solving the identified problem are communicated to the community of interested stakeholders or users. In this thesis, publications in conferences and journal served as a means of reporting and communicating research outcomes from different processes to get feedback from academia and industry. These publications are in the following sequence:

a. Sustainability Quantification in Requirements Informing Design: This publication summarises our investigations on sustainability during software design and development, focused on the requirements engineering processes.
b. Catalogue Supporting Software Sustainability Design: This publication introduces SSDC, which comprises a series of guidelines and a framework for the sustainability of software system design.
c. Classifying the Measures of Software Sustainability: This publication identifies and compiles the measures of green software and software sustainability from sustainability perceptions and dimensions.
d. Software Sustainability Good Practices: This publication explores the derivation of good practices by applying sustainability in software design and development.
e. Validation Study of a Framework for Sustainable Software System Design and Development: This publication presents the results for validating the proposed framework for the sustainability of software system design.

This thesis is based on an iterative design process for addressing the research questions for software sustainability design and development. Design science offers the right research method to address the challenges and problems identified from the research gaps. The design science process guides the process of designing the artefacts that can be applied to address the specified research questions. The design science research cycles support a quality iterative process for the design, application, evaluation and communication of artefacts.

3.1.3 Case Study Research Method

Case study design research method is an empirical inquiry that investigates a contemporary phenomenon within its real life context when the boundaries between phenomenon and context are not clearly evident, and in which multiple sources of evidence are used (Yin, 1984; Zainal, 2007).

There are five components of case study design according to Yin (Yin, 1984), namely: A study's questions, study proposition (if any), units of analysis, the logic linking data to the propositions, and criteria for interpreting the findings.

1. **Study Questions:** The first component of case study research design is the formation of questions in terms of 'who', 'what', 'where', 'how', and 'why', which provides the basis for the relevant research strategy to use. In Publication 5, the study question is about how to use SSDC and FSSSD to guide and support stakeholders during software design, development and measurement." The second

study question is on "why companies have difficulties in incorporating sustainability into their software design and development project."

2. **Study proposition:** This is about what should be examined with the scope of the study. The 'how' and 'why' only shows what an investigator is interested in, but does not point the investigator to what should be studied. In Publication 5, the proposition is that a software sustainability framework can help stakeholders in software design and development, and the second proposition is that the lack of concrete guidelines exemplifying the use of sustainability in software design affects the adoption of software sustainability design and development in companies.
3. **Units of analysis:** The third component is the fundamental problem of defining what the case is in case studies. A case can be an individual, implementation, company, entity, event, organisational change or programme. The proposition helps identify the case and guides the investigator on what data to collect instead of collecting data on everything. The definition of the unit analysis of a case is related to the way the initial research questions have been defined. Publication 5 details a case of implementing FSSSD in company settings with two case studies and the unit analysis for the case is the number of decisions and practices influenced by FSSSD during software design and development.
4. **The logic linking data to the propositions**: This component is about connecting data collected with the propositions in the case study. One approach is 'pattern-matching', in which several pieces of information from a case study are related to some theoretical proposition. The data collected during the application of SSDC and FSSSD were checked to see if it match the two propositions.
5. **Criteria for interpreting the findings**: In order to avoid confusion on interpreting findings, it is good to set conditions base on the logic linking data to the propositions used in understanding findings based on the unit of analysis. The conditions set are based on checking the usefulness and effectiveness of SSDC and FSSSD in supporting and guiding stakeholders during software sustainability design and development in the two case studies detailed in Publication 5.

4 Publication Overview

The research publications, as a core contribution to this thesis, are summarised in this chapter, focusing on all five publications. The five publications address the identification of different research challenges in software sustainability design, development and measurement. All the created artefacts are aimed towards supporting and guiding software requirement engineers, software architects, developers and companies for software sustainability design and development. Table 6 provides the list of publications, descriptions, outcomes and SDLC Phase were those outcomes are useful.

Table 6. Summary of Publications in Thesis

Publication	**Description**	**Outcomes**	**SDLC Phase**
Publication I	Study the factors affecting sustainability quantification in software development	Sustainable business goal question metric (S-BGQM) approach to software development	User requirement System requirement
Publication II	Investigated the challenges of sustainability in software design and development. Proposed the SSDC and pilot framework to assist software developers and managers in eliciting software sustainability requirements and measuring software sustainability	Software sustainability design catalogue (SSDC) Framework for Sustainability of Software System Design (FSSSD)	All SDLC phases
Publication III	Studied current software sustainability measures based on the five sustainability dimensions and categorised them into four perceptions (Sustainability in Software Development; Green Software Systems; Software for Sustainability; and Sustainability of the Software Eco System)	Categorisation of software sustainability understandings into four perceptions and measures associated to each perceptions	Integration and Testing
Publication IV	Explore how to document software sustainability requirements good and best practices during the design and development of software system guided by KMSD (Becker *et al.*, 2015)	Method for collecting and disseminating software sustainability requirement elicitation best practices Template for software sustainability	All SDLC phases

		requirements elicitation best practices	
Publication V	Validation study of SSDC and FSSSD using two case studies	Validation results of SSDC and FSSSD and challenges from stakeholders involve in the two case studies	All SDLC phases

4.1 Publication I: Sustainability Quantification in Requirements Informing Design

4.1.1 Research Objective

The main objective is to study different sustainability definitions and measures, and how those definitions relate to a software system in SE to generate software sustainability requirements and how to quantify sustainability in software design.

4.1.2 Relation to Thesis Research Question

Publication I investigated the first research question on how to elicit software sustainability requirements during software design and development. Software quality criteria, generated from the sustainability definitions in Table 4, shows some of the requirements for software sustainability. The second research question, how to measure and evaluate software system sustainability, is also explored in this publication using the S-BGQM approach to software design and development, which provides setting questions to characterise each sustainability goal for the software. The answers to the questions are then used to create metrics and indicators to evaluate software sustainability, based on the context of development.

4.1.3 Research Output and Contribution

The first contribution from this publication is the summary of different sustainability definitions and the software requirements identified from those definitions. Table 7 shows the sustainability definitions and software requirements.

Table 7 Sustainability Definitions and Software Requirements (Oyedeji, Seffah and Penzenstadler, 2017)

Author	Definition	Requirement
Idio (Idio, 2014)	Long-lasting and Lean software, software for sustainable humans	Energy efficiency, longevity and user experiences

Author	Definition	Requirement
Venters *et al.* (Venters *et al.*, 2014)	"Sustainability is the quality of being sustained. Longevity and maintenance are the two most important factors for understanding sustainability".	Longevity and maintenance
Heiko Koziolek (Koziolek, 2011)	"Long-living system that should last for more than 15 years and can be cost-efficiently maintained and evolved over its entire life cycle."	Longevity and maintenance
Seacord *et al.* (Seacord *et al.*, 2013)	Ability to modify a software system based on customer needs and deploy these modifications.	Modifiability
(Harris and Goodwin 2001)	"Sustainability as a system that must achieve fairness in distribution and opportunity, adequate provision of social services."	Accessibility
(Naumann *et al.*, 2011)	"Software whose direct and indirect negative impacts on the economy, society, human beings and environment that result from development, deployment and usage of the software are minimal."	Economic, environment, social and individual dimensions of sustainability
Tainter (Tainter, 2006)	To define sustainability in a specific context, the questions should be "to sustain what, for whom, how long and at what cost?"	Sustainability is a requirement within a certain context and requires specification of the context

The second contribution is the S-BGQM approach to encourage consideration of sustainability during software design and development, especially in the requirements and software testing phases. Figure 8 provides details of the S-BGQM approach during software design and development. S-BGQM is aimed at supporting both technical (requirement engineers, developers, testers, architects) and non-technical (business requirement personnel, business managers and project managers) stakeholders that incorporate sustainability into their software development or enhancement projects.

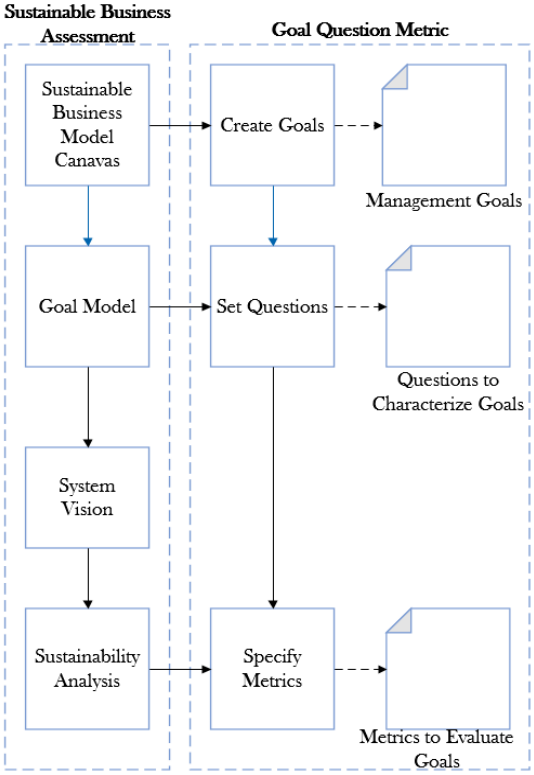


Figure 8. Sustainable business goal metric process flow (S-BGQM; (Oyedeji, Seffah and Penzenstadler, 2017)

4.2 Publication II: A Catalogue Supporting Software Sustainability Design

4.2.1 Research Objective

The research goal of this publication is to provide concrete guidelines that software architects and developers can apply effectively with support and guidance in the elicitation of software sustainability requirements, design, measuring and testing software sustainability against set requirements.

4.2.2 Relation to Thesis Research Question

The two contributions from this publication (SSDC and FSSSD) relate to the first and second research questions of this thesis on how to elicit software sustainability requirements and how to measure the software sustainability during design.

The SSDC provides a series of software sustainability guidelines to help stakeholders become involved in software design and development, based on the analysis of different software systems, to improve their understanding of sustainability in software design, which in turn can facilitate better software sustainability requirements and measurements.

The FSSSD was developed to guide and support stakeholders incorporate sustainability goals, requirements and measurements during software design and development. FSSSD, through its structures, guides developers on how to create software sustainability goals, elicit software sustainability requirements and identify the right software sustainability measures to evaluate the software.

4.2.3 Research Output and Contribution

The first main research output from this publication is the SSDC. The SSDC is a set of guidelines derived from the nine Karlskrona Manifesto principles, based on a cross-analysis of different systems. SSDC was created to aid sustainability integration in software design and offer a better understanding to software architects, practitioners and other stakeholders on sustainability in software design, development and measurement. Figure 9 shows how SSDC was created using the Karlskrona Manifesto principles, sustainability dimensions and the three orders of impact (first, second and third order impacts).

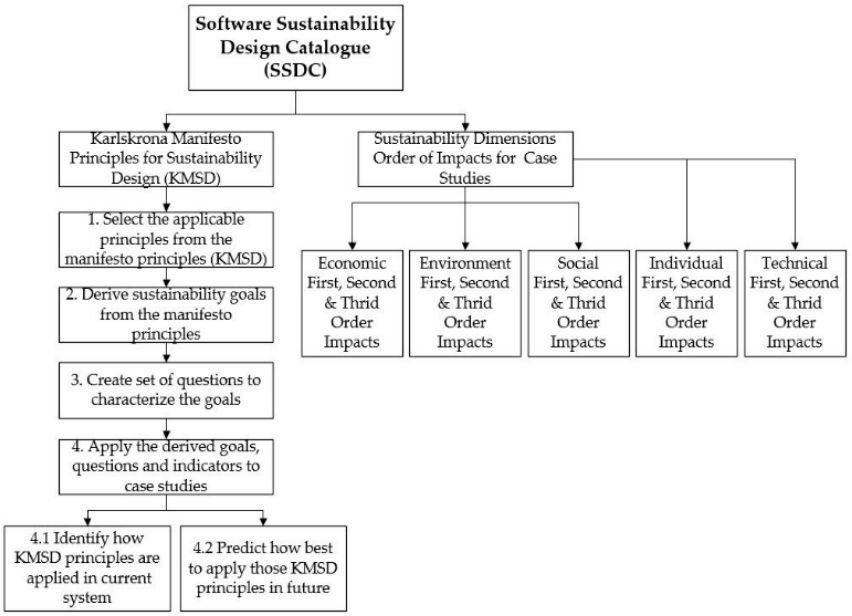


Figure 9. Structure and flow of the derivation of the SSDC (Oyedeji, Seffah and Penzenstadler, 2018a)

The second research output from this publication is the pilot FSSSD, based on SSDC, to guide and support stakeholders to use sustainability as a core metric during software design and development when covering the whole software development life cycle. Figure 10 shows the details of FSSSD, and Table 8 presents FSSSD in tabular form for better understanding.

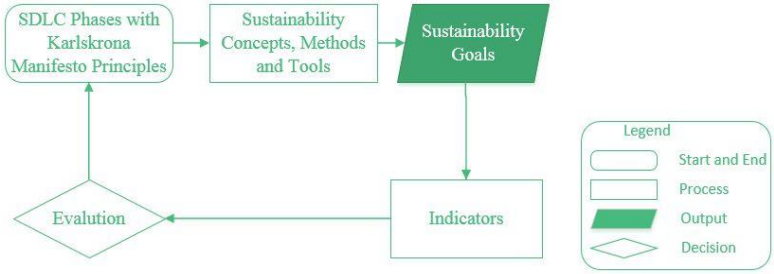


Figure 10. FSSSD (Oyedeji, Seffah and Penzenstadler, 2018a)

Table 8. Contents of the FSSSD (Oyedeji, Seffah and Penzenstadler, 2018a)

SDLC phases and KMSD principles	**Sustainability goals**	**Sustainability concepts, Methods and Tools**	**Indicators**
Phase 1. Project definition, P1, P2 and P3	Design for sustainable efficiency, reusability	Flourishing Business Canvas	Carbon footprint, material footprint, end of life footprint
Phase 2. User requirements definition, P2	Increase sustainability awareness among users	Sustainability requirement template	Total number of sustainability requirements, priority assign to sustainability requirements
Phase 3. System requirements definition, P4, and P5	Design for efficiency, sustainability awareness and interoperability	Sustainability requirement template, goal model	Total number of system goals relating to sustainability dimensions
Phase 4. Analysis and design, P2, P4, P6 and P8	Design for reuse and efficiency, localisation, interoperability	Lifecycle sustainability assessment, social return on investment,	Number of first-, second- and third order impacts of the system identified

		sustainability analysis radar chart	
Phase 5. Development, P2 and P4	Design for reuse, design for module replicability, design for efficiency, sustainability awareness, efficiency and design for easy service and maintenance	Biomimicry	Number of coding choices influenced by sustainability, number of features (functions) added to systems to inform users about sustainability through functions like eco feedback
Phase 6. Integration and testing, P2 and P4	Design for easy assembly and disassembly and design for durability	Sustainability analysis radar chart, life cycle sustainability assessment	How much information from sustainability analysis chart was used during integration and testing, such as the number of systems functions tested against sustainability concerns such as the first order (immediate) impact, possible second order (enabling) and potential third order (structural) impacts to the system
Phase 7. Implementation, P5 and P7	Design for easy use, design to induce conscious sustainability awareness, design to educate users about sustainability and design for easy recycling	Sustainability analysis radar chart	The priority assigned to sustainability by developers and the system owners/users during or after implementation
Phase 8. Sustainment and maintenance, P9	Proper design for serviceability, design for easy replacement of code modules and design for continuous user engagement through sustainability awareness	Life cycle sustainability assessment, sustainability analysis radar chart	Number of improvements to the system based on sustainability requirements, either from users' feedback or developers

These are brief explanation of the sustainability concepts and tools used in FSSSD as shown in Table 8:

1. **The Flourishing Business Canvas (Sustainable Business Canvas):** is a visual design tool that embeds a common language to enable more effective collaboration by stakeholders deemed relevant to designing the economic, social and environmental aspects of an organization's business model. The tool aims at maximizing positive and avoiding negative impact on society and nature. Therefore, sustainability is integrated into the core business. (Enterprise, 2019)
2. **Sustainability Requirement Template:** provide stakeholders a way of categorizing software requirements into the five sustainability dimensions. The template foster better thinking on how software requirements relates to each sustainability dimension and provide an avenue to understand the requirement categorization by stakeholders (Oyedeji and Penzenstadler, 2019).
3. **Goal Model**: provides a holistic grouping of software application goals into business goal, usage goal and system goals to identify conflicts early in order to resolve them with consideration of the five sustainability dimensions (Penzenstadler, 2016).
4. **Life Cycle sustainability assessment (LCSA):** refers to the evaluation of all environmental, social and economic negative impacts and benefits in decision-making processes towards more sustainable products throughout their life cycle (Guinée, 2016).
5. **Social Return on Investment (SROI):** is a process of understanding, measuring and reporting on the social, environmental and economic value that is being created by an organisation (Cohen, Robbins and Denault, 2012). This is calculated using: Net present value of benefits / Net present value of investment
6. **Sustainability Analysis Radar Chart:** This is a chart that presents the effects and impacts of software system considering the five sustainability dimensions with the first order impact (immediate effect), second order impacts (enabling effect) and third order impacts (structural effect) (Becker *et al.*, 2016b).
7. **Biomimicry**: is a science of studying the designs, processes, and phenomena in nature as a source of inspiration for human creations. It recognizes nature as a model for us to emulate in our designs, measures to evaluates design and a mentor from which to learn (Benyus, 1997). Biomimicry (from bios, meaning life, and mimesis, meaning to imitate) is a design discipline which studies nature's ideas that can be imitated in design process to solve human problems (Mann, 2007).

4.3 Publication III: Classifying the Measures of Software Sustainability

4.3.1 Research Objective

This publication aims to study the different proposed and suggested measures of software sustainability to identify different measures for green software, software sustainability and the perceptions of software sustainability measures in SE.

4.3.2 Relation to Thesis Research Question

The research output from this publication relates to the second research question about how to measure and evaluate software system sustainability. The measures compiled in this publication highlight the kinds of measures currently being applied for software sustainability and green software. This publication also raises some research needs about software sustainability measurement and the need for a framework to ground the derivation of new software sustainability measures with a clear interpretation based on the general software measurement theory.

4.3.3 Research Output and Contribution

The first main contribution is the categorisation of software sustainability understandings into four perceptions, namely, sustainability in software development (development), software for sustainability (usage), green software systems (focused impact) and sustainability of software ecosystems (net effect).

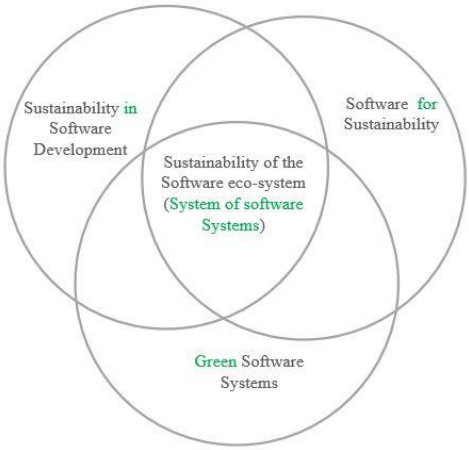


Figure 11. Sustainability perceptions for SE (Oyedeji, Seffah and Penzenstadler, 2018b)

Figure 11 shows the four perceptions and their relation. Out the four perceptions, the green software system has received the highest number of related research works on measures for evaluating green software systems.

The second contribution is the compilation of different measures used in evaluating green and sustainable software systems based on the software development life cycle, the four sustainability perceptions and the five sustainability dimensions. Publication 3 shows what measures are used in evaluating green and sustainable software systems and what formula were applied, with a definition of each measure.

4.4 Publication IV: Karlskrona Manifesto - Software Requirement Engineering Good Practices

4.4.1 Research Objective

The goal of this publication is to explore the derivation of good and best practices during software design and development by applying the Karlskrona Manifesto principles in software sustainability requirements elicitation and documenting these best practices in a template to aid dissemination among interested stakeholders.

4.4.2 Relation to Thesis Research Question

The contribution from this publication covers the third research question on how to record good practices for software sustainability design and development. The method for documenting software sustainability requirements best practices and the template proposed in this publication can serve as a step towards encouraging different interested parties to document and share how sustainability was used in their software projects, both in academia and industry, to increase the knowledge base on software sustainability design and development. This can assist and educate novice stakeholders interested in software sustainability on what to do and how to do it during software sustainability design, development and measurement.

4.4.3 Research Output and Contribution

The research contribution from this publication is a method for collecting and disseminating software sustainability requirement elicitation best practices and providing a template for the documentation. Long-term usage of the template for documentation of different software sustainability requirement best practices will serve as a central point to educate software developers on how sustainability is treated in different software projects and contexts, which can help the rethink of how software is currently designed and

developed in the future. Table 9 addresses details of the template for documenting software sustainability requirement elicitation best practices.

Table 9 Template for software sustainability requirements elicitation of best practices (Oyedeji and Penzenstadler, 2018)

Element	Description
Title	Which title best describes best practice?
Date	In what month and year is the 'good practice' published or documented?
Authors	Who wrote the good practice document?
Target Audience	Who is the target group? To whom is this document useful?
Objective	What is the goal or aim of the best practice?
Location	What is the geographic location in which this practice can be applied for a software system (country, region, town or village)? Examples: system for a country, state, province health care system or banking system or a commercial software application
Stakeholders	Beneficiaries of this best practice? Who are the users, institutions and implementing agencies of the best practice?
Methodology	What methodology was used in documenting best practice? What were the process and steps involved?
Selected Karlskrona Manifesto principles	What are the principles that served as a guide for creating best practices for requirements elicitation?
Requirements	What were the requirements used in best practice? How was sustainability considered in the requirement?
Validation	How was best practice validated? Did best practice fulfil the best practice criteria?
Impact	What was the impact in the application of best practice?
Lessons Learnt	What is the key take-away from the application of best practice?
Sustainability	What are the dimensions of sustainability covered in best practice application?
Contact Details	What are the contact details of those responsible for best practice?

4.5 Publication V: Validation Study of a Framework for Sustainable Software System Design and Development

4.5.1 Research Objective

The goal of this publication is to present the validation results for the SSDC and FSSSD by applying it in the case of studies for software design, development and measurement.

4.5.2 Relation to Thesis Research Question

The contribution from this publication relates to three research questions of this thesis because the validation study results show SSDC and FSSSD provides the necessary guidance and support to create software sustainability goals, elicit software sustainability requirements and facilitate the measurement of software based on sustainability measures.

4.5.3 Research Output and Contribution

The research contribution from this publication is the validation of applying SSDC and FSSSD in the two case studies during software design, development and measurement.

The first case study where SSDC and FSSSD were applied is about how to design and develop a pension benefit tracker application. This case study does not have sustainability as a core motive during the project initiation, but the application of FSSSD shows that the project stakeholders made changes to some of the software requirements and added new ones to improve the overall efficiency of the pension benefit tracker application. The usage of FSSSD in this case study also shows the benefits of sustainability consideration in software design and development not only for the software but also the stakeholders. This is because the developers' satisfaction during development (individual dimension of sustainability) was considered as a yardstick for evaluating the software project by the company’s project manager.

The second case study is about an application to display energy usage and carbon emission through activities with the university staff and students driven by the university’s sustainability goals. FSSSD provided the necessary guidance and support in assisting the stakeholders in creating sustainability goals methodically, elicit sustainability requirements for the project and incorporate these goals into the design and development of the application in each software development life cycle.

5 Results and Evaluation

This chapter presents a summary of the results that describes the evolution of the research, based on different design iterations, by explaining how and why the artefacts designed and used in this thesis were created. The validity of the design processes, threat to validity and the research limitations are also explained.

5.1 Research Evolution and Results

The first research stage (Publication 1 and Chapter 4, Section 4.1) of this thesis focused on the requirements engineering domain by exploring sustainability definitions about software systems; the key elements of requirements in the form of software quality were derived from those definitions of sustainability. The identified sustainability dimensions for software sustainability requirements in Publication 1 shows there is no consensus on what software sustainability requirements should be during design and development. This is corroborated by research results in (Venters *et al.*, 2017).

The first challenge of software sustainability requirements during software design and development, as noted in Publication 1, is how to elicit software sustainability requirements methodically. The second challenge is on how stakeholders can be guided from software project initiation to final product delivery and how to derive necessary software sustainability goals that can facilitate software sustainability design. The third challenge is on how to evaluate software system, based on the elicited software sustainability requirements. How to identify indicators, measures and benchmarks for the proper evaluation of software sustainability requirements to ensure sustainability issues are addressed throughout the whole software development life cycle.

The S-BGQM (first artefact) approach is proposed in Publication 1 to address these challenges by supporting stakeholders with processes that would help identify key sustainability goals of a software system and specify software sustainability requirements. Identify areas within the context of the application that the software will impact, set questions that will characterise the goals, specify indicators or measure to evaluate the sustainability goals of the software system related to each of the software sustainability requirements. S-BGQM approach also supports sustainability analysis of software system to provide insights on major first, second and third order impacts of the software system within the five sustainability dimensions.

The S-BGQM approach was useful to support software sustainability requirements and evaluation during software design and development but does not offer a complete guide for the whole software development life cycle. Without a complete guide covering the whole software development life cycle on software sustainability design, development and measurement, it becomes difficult for stakeholders to adopt sustainability in the software design and development process. Publication 2 explores how the Karlskrona Manifesto principles on sustainability design (Becker *et al.*, 2015) can be used as a guide

in all phases of the software development life cycle, as shown through the development of a second artefact (SSDC) and a third artefact, (FSSSD) in Publication 2.

Publication 2 describes SSDC as a guide for the integration of sustainability during software design and encourages sustainability design as a core part of software design practices based on the sustainability analysis of different kinds of software systems. FSSSD, which was derived from the understanding of software sustainability design through the SSDC guidelines, serves as a framework to address the problem of holistic guidance and support covering the whole software development life cycle. This will ease the challenge of software sustainability design, development and measurement. Table 10 details the interpretation of all the Karlskrona Manifesto principles on sustainability design to each phase of the software development life cycle and the sustainability design goals in each phase. The nine Karlskrona Manifesto principles and design goals are presented in Table 10.

Table 10. Publication 2 summary of the Karlskrona Manifesto principles in relation to SDLC phases and sustainability goals

SDLC Phases	Karlskrona Manifesto principles	Design Goals
Phase 1: Project Definition	**P1-** This principle ensures that the project initiation considers sustainability in the overall project definition from the beginning. **P2-** Software sustainability has different dimensions that must be considered from the beginning for better project management with different stakeholders. **P3-** Software projects usually involve stakeholders from different domains, incorporating their sustainability concerns and proper management of those concerns from multiple perspectives, will help the incorporation of sustainability in the software.	Design for sustainable efficiency, reusability, transferability, ensure a good working environment for developers and work satisfaction.
Phase 2: User Requirements Definition	**P2-** It is important to take note of user requirements regarding each sustainability dimension to have better sustainability analysis during the analysis and design phase.	Increase sustainability awareness among requirement engineers and users during requirement gathering.
Phase 3: System Requirements Definition	**P4-** During elicitation of system requirements, requirement engineers should consider sustainability concerns for the system during the requirements definition, even when it is not a core part of the user requirements. **P5-** Cross-evaluate the consequential impacts of system sustainability requirements and the environment in which the system will function.	Design for efficiency, sustainability awareness and interoperability.
Phase 4. Analysis and Design	**P2-** Applying this principle provides a blueprint for system evaluation from all sustainability	Design for reuse and efficiency, localisation, interoperability

	dimensions (economic, environment, social, individual and technical). **P4-** At this phase, the principle helps to encourage the analysis of system design based on sustainability in order to facilitate better sustainable system design. **P6-** Application of this principle enables a better visual and visible overview of the system from different levels of abstraction. **P8-** This principle provides a better understanding during analysis to make choices that will help the potential users of the system in both the present and future when the system evolves.	
Phase 5. Development	**P2-** This principle will encourage developers to consider different sustainability dimensions during this phase, especially technical, social and individual dimensions. **P4-** Encourage the search for better avenues to make systems sustainable from the development perspective (developers) and also the functions of the system that support longevity.	Design for reuse, design for module replicability, design for efficiency, sustainability awareness, efficiency, design for easy service and maintenance
Phase 6. Integration and Testing	**P2-** Provides integration and a sustainability template for the test team that can be used to test the system for all sustainability dimensions based on the sustainability requirement output from phases 2, 3 and 4. **P4-** Application of this principle will support the consideration of sustainability in this phase, even if the primary focus of the system is not about sustainability.	Design for easy assembly and disassembly, design for durability
Phase 7. Implementation	**P5-** Provides beforehand reasoning for the development team to consider the sustainability of the system, its production environment and when pushing it live for use. **P7-** Based on Principle 5 (P5), this principle aid consideration of seeking the involvement of different stakeholders to make the actualisation of the system sustainability possible in the production environment and when pushed live.	Design for easy use, design to induce conscious sustainability awareness, design to educate users about sustainability, design for easy recycling
Phase 8. Sustainment/ Maintenance	**P9-** This principle, at this stage, helps create conscious awareness so that when the system is in a live environment, there will be a continuous evaluation to assess system sustainability and think of ways to optimise and improve the system's sustainability from all dimensions.	Proper design for serviceability, design for easy replacement of code modules, design for continuous user engagement through sustainability awareness

Publication 2 provides two key artefacts that support software sustainability design, development and measurement with discussion about sustainability requirements during software development, which is usually not supported by most requirement gathering method as stated by (Seyff *et al.*, 2018). All the research outcome from publication 2 also contributes to promoting software sustainability design which presently has limited research as shown in the research work of (Wolfram, Lago and Osborne, 2017). However, one challenge noted in Publication 2 is the lack of measures for software sustainability design and a knowledge base of how software sustainability measures have been applied in the industry through transfer of research knowledge from academia to practice in industry.

Publication 3 investigates how different software sustainability measures have been identified, created, applied and evaluated in different research setups. This resulted in the compilation of different software sustainability and green software measures, based on the four sustainability perceptions. From the four identified perceptions of sustainability in SE, the green software system has the largest number of measures out of the compiled measures in Publication 3. In particular, the energy efficiency measure, which is the most used measure in evaluating software sustainability and greenness. Most of the measures also focus on the environmental and technical dimensions of sustainability only, with few measures in the social, individual and economic dimensions. According to (Lami and Buglione, 2012), there are few studies about 'what' aspect of software sustainability to measure and 'how' to measure it efficiently. This is evident in the compiled software sustainability measures in Publication 3.

One reason identified for the lack of measures covering all sustainability dimensions in Publication 3 is the problem of understanding the different scales of software sustainability measures in the different sustainability dimensions and the interpretation of these measures by stakeholders. For example, EU Directive 92/75/EC which established energy consumption labelling scheme for energy efficiency of refrigerator, categorised the labelling using A+, A++ and A+++, which means A++ is better than A+, and A+++ is better than A++, based on energy consumption of the refrigerator. For software sustainability measures during design and development, how could we have a similar scale of measurement and the right interpretation of these scales of measurement for software sustainability design and measurement? A good start can be documenting different software sustainability design, and development projects to record what kind of sustainability requirements and measures were applied, such as how the requirements and measures were created and what interpretation and scale of measurement were associated with these measures. This can then provide a central reference point to show how measures of software sustainability are exemplified and to provide a good start for grounding those measures in general software measurement theories. Requirements in software sustainability design and development are key to having measures for evaluating the software system, based on those requirements.

Hence, Publication 4 researched how to create a method for documenting and disseminating software sustainability requirement elicitation best practices and provided

a template for easy documentation. The criteria for validating the best practice consist of the following six, namely:

1. **Effective and successful**: The best practice has demonstrated relevance to achieve an objective effectively and efficiently, and has been used successfully with a positive outcome and impact.
2. **Environmentally, economically, individually, and socially sustainable:** The best practice meets current requirements and needs without compromising the capacity to address future needs.
3. **Technically feasible**: Easy to understand, learn and apply in other similar cases or scenarios.
4. **Inherently participatory**: Supports a participatory approach to facilitate joint decision-making and actions.
5. **Replicable and adaptable:** Can easily be replicated in other projects and adaptable in a different context with similar objectives.
6. **Reducing disaster/crisis risks**, if applicable: Supports risk reduction and crisis to facilitate resilience.

The template for documenting the requirement elicitation best practices uses the Karlskrona Manifesto principles for sustainability design as a guide for the requirements and the different sustainability dimensions. As shown in a study of requirements engineering practitioners that practitioners perception regarding sustainability are limited due to a narrow understanding of sustainability (Chitchyan *et al.*, 2016). And also there still is no single point of reference for either RE researchers or practitioners where the work on sustainability is gathered and exemplified (Chitchyan *et al.*, 2015). Overtime, the usage of this template as a documentation of best practice during software sustainability design can provide more examples to improve practitioners understanding for applying sustainability in software design and also contribute as a reference point. Table 11 shows an example of the template from Case Study 2 in Publication 5 about the energy usage and carbon emission display for university staff and students. This best practice documentation was not included in the Publication 5 because it was not available at the time of the publication. Publication 4 also provides a sample of how this template was used.

Table 11. Best practice documentation from Case Study 2 (Sustainability Awareness via Energy Data Display) in Publication 5

Element	**Description**
Title	Develop sustainability awareness in energy display application for the public
Date	12/08/2018
Authors	Mistretta Tom – Devinez Alexandre
Target Audience	Engineers / Developers
Objective	• Create awareness about sustainability requirements in a project • Encourage the development of ideas around sobriety

Location	Applicable worldwide
Stakeholders	Engineers / Developers / Users
Methodology	• Discussion among software development team on what sustainability means to them by going through the Karlskrona Manifesto principles, FSSSD and SSDC • Dialogue about which requirements can better influence users' awareness of sustainability • Dialogue about which requirements can better teach users to improve their daily habits, influenced by the information shown to them • Discussion of how to integrate sobriety awareness requirement in the project • Find a way to make the project attractive to users
Selected Karlskrona Manifesto principles	Principle 6: System visibility is a necessary precondition and enabler for sustainability design. Principle 7: Sustainability requires action on multiple levels. Principle 8: Sustainability requires meeting the needs of future generations without compromising the prosperity of the current generation. Principle 9: Sustainability requires long-term thinking.
Requirements	**Functional Requirement** REQ 1 – Interactivity (users must be able to interact with the application) • The interface must be simple to catch the user's attention. • Users can make actions on the interface with energy data and dynamically get eco feedback. REQ 2 – Display Information • The users should be able to understand the displayed data and information. • Energy usage data and carbon emission information should be displayed to users in relation to road distance between LUT University in Lappeenranta and other cities within Finland (this will provide a better understanding to users regarding their impact). REQ 3 – Community (users must be able to share ideas on sustainability and advice to the user community group) • Provide users with a sustainability challenge every week, dynamically based on energy usage to help users develop a sense of belonging with the idea of sustainability beyond the university. This can make them become more curious and choose to change their habits.
Validation	Engineers, developers and some end users validate these requirements with the best practice criteria.
Impact	Promote sustainability and sobriety awareness
Lessons Learnt	1. Test results from user interaction with the prototype design show users gain a sense of pride if their advice and suggestions help reduce energy usage in the community section 2. The prototype test result also shows the best way to influence public behaviour is to present energy and carbon emission information in relation to what users can easily relate to, which

	can offer better understanding for the public about their impact on the environment. This approach is why the equivalent of CO_2 emission, based on energy usage data, has been presented in the form of distance between one city and another to explain the impact on sustainability. This will encourage a change in users' habits over time instead of telling them to change their habits based on high energy usage data displays or CO_2 emissions.
Sustainability Dimensions	The requirements in this template cover the following: • Social sustainability • Environmental sustainability • Individual sustainability
Contact Details	mistrettatomjulien@gmail.com , devinez.alexandre@gmail.com

Publications 1- 4 provided the problems and challenges for stakeholders in engineering software sustainability requirements, design and development. The publications also presented all designed artefacts to address the problems described above and the challenges of software sustainability design. To check the validity of the artefacts, its applicability and usefulness, Publication 5 details a validation study through a case study on the proposed design artefacts.

Publication 5 details the application of SSDC and FSSSD, which was created based on guidelines from the SSDC in two case studies. The result from the application of the framework shows it assisted and guided stakeholders on how to derive sustainability goals at each phase of the SDLC, supported through the use of different software sustainability concepts, tools and methods. Based on the sustainability goals in each SDLC phase, stakeholders were able to specify sustainability requirements, identify the measures and indicators that will be useful in evaluating the sustainability of software systems and applications. Table 12 shows a sample of how the sustainability requirement template from the FSSSD was used in the pension application case study. The sustainability requirement template provided a way to understand how stakeholders involved in the case study viewed each sustainability dimensions in relation to the software requirements.

Table 12. Sustainability Requirement Template (Oyedeji and Penzenstadler, 2019)

Requirement	**Sustainability Dimension**	**Sustainability Dimension and Explanation**
The pension tracker application should be accessible online via web at any branch	Economic and Technical	It will save us money of using interstate courier to send, receive and track pension applications. (economic) To achieve this, a good functional system with no down time that will satisfy user needs is required (technical)

The application should have ability to enable Managers, pensioners and other stakeholders check application status	Technical, individual and social	Ease of use (individual) and also allows everyone using the system to be up to date about pension application status (Technical and social)
Provide automatic status communication and notification at each stage of benefit application	Individual and Social	It will keep clients (pension applicants) up to date about their application (individual and social)
Allow bulk or single file upload	Individual and Technical	More options to reduce time spent in uploading application files (individual, technical)
Provide SMS authorization from managers in benefit department	Individual	Provide ease of processing and approval for managers (individual)
Send Incomplete documentation notification to benefit department staff	Individual and economic	Reduce time of processing the pension application (individual, economic)
Provide email and SMS notification as an option for all users	Individual	Provide more options to increase user preference because some users might not have access to email (individual)
Provide option of different display to magnify fonts for users with visual problems	Individual	This promote inclusiveness especially with users with visual problem (individual)
Provide option to preview pension application and save electronically	Individual	Reduce amount of error in applications and saves time of double work (individual)
Add a tag message below each notification "Save the planet from environmental waste, print only when needed"	Environmental	Promote sustainability awareness among staff and clients (pension applicants)
Provide energy report for system usage	Environmental and Technical	This will enable users track the amount of energy consumed by the application and discuss how we can improve it

As shown in Table 13, presenting direct statements of stakeholders involved in using SSDC and FSSSD for the two case studies, the framework provided the necessary guide and support to make stakeholders rethink how to design and develop software systems and using sustainability principles as a core guide in software design. Overall, the feedback from the stakeholders in the two case studies shows that the application of SSDC and FSSSD was useful. However, there were some challenges during the application of FSSSD due to the stakeholders' lack of understanding of what sustainability means in software design and development. Using the SSDC guidelines during a discussion with

stakeholders and providing more explanation of how to use the tools and methods provided in FSSSD, stakeholders were able to adopt it in their software design and development project. Table 12 presents a summary of direct statements from stakeholders in the two case studies.

Table 13. Direct quotes, feedback and comments from participants and stakeholders in the two case studies (Oyedeji *et al.*, 2019)

Role	**SDLC Phase**	**Positive**	**Challenges**
CTO	Project Definition	1. FSSSD and SSDC provides new insight for software project with consideration of sustainability 2. Guidelines in SSDC and FSSSD introduces new methods for evaluating our applications, especially the environmental and individual dimensions of sustainability 3. The Sustainable Business Canvas brings new factors into software project definition with sustainability concepts and dimensions as guides	1. Very difficult to understand how to apply some of the sustainability concepts because it is new to my team and me 2. We have the challenge to find concrete examples online to see how sustainability was applied to software project definition, especially in industry 3. It was challenging to give my staff the additional task of reading the Framework manual to understand how to apply it
Software developer, Project coordinator	User requirement definition	1. The sustainability requirement template was useful as a guide during the requirement gathering because it provides us with means of discussing sustainability with users and categorising user requirements based on sustainability dimensions	1. It was difficult at first to understand how to explain the different dimensions of sustainability to key stakeholders (users) during discussion gathering requirements on how to improve the existing system
System analyst, software developer	System requirements Definition	1. I was able to learn new things about how sustainability can influence defining system requirements and identifying new system requirements using the FSSSD 2. The goal model diagram is a good tool to break down sustainability goals based on requirements for business, usage and system goals 3. The goal model diagram made it easy to explain, discuss and improve the project goals and system requirements using the business, usage and system goal diagram	1. The only issue is a lack of examples to show how sustainability has been used in different software requirement elicitations at the beginning when using FSSSD, but after a couple of meetings discussing sustainability with the research guy, things became clearer 2. Some of the research, especially about sustainability in system requirements I saw on Google from some researchers, are too complex to apply
System analyst, Programm	Analysis and Design	1. The sustainability goals and suggested tools from FSSSD were a good starting point to	1. Brainstorming how to connect the first, second and third order impact in each of the

ers, Software developer		guide us during the analysis and design phase 2. The sustainability analysis radar chat was a new interesting tool because it shows some new requirements to add after brainstorming each of the first, second and third impacts	sustainability dimensions was not easy because each of us has different views on what is the right thing to put, but eventually, we looked at some of the examples provided by the researcher guy in using FSSSD.

5.2 Evaluation of Validity in Design Process

The seven guidelines for design science research (Hevner *et al.*, 2004) provide researchers with better knowledge and understanding of a design problem, the solution required in building and the application of the artefact. These seven guidelines were used in this thesis (detailed in Chapter 3) and serve as the evaluation for validity and quality of the design science process.

1. **Design as an Artefact:** The result of design science research is a useful artefact designed to address a particular organisational problem or challenge. A clear description of the artefact must be stated effectively to enable better implementation and application in the appropriate domain or context. Artefacts created in design science research include constructs, models and methods applied in the development and use of ISs.
   Applied in this research process: This thesis produced four artefacts (see Publications 1, 2 and 4) to address the challenges of software sustainability design and development. Chapter 3 details how the design science process for creating the artefacts was conducted (see Section 3.1) and Chapter 4 presents the results for each artefact.
2. **Problem Relevance:** The goal of research in information system is to gain knowledge and understanding that can aid the development and implementation of a technology-based solution to solve crucial problems. Design science approaches this goal through the creation of artefacts relevant to solve the problem in the application context.
   Applied in this research process: A literature review of scientific articles was conducted as well as discussions with industry stakeholders to investigate the problems of how to guide and support software developers in the design, development and measurement of software sustainability. The result of a literature study for this problem is summarised in Chapter 2 (Section 2.2) and Publications 1- 4.
3. **Design Evaluation:** The efficacy and quality of a design artefact must be rigorously demonstrated through an efficient evaluation method. Evaluation is an

important part of the design research process. The application context or environment sets the requirements upon which the created artefacts are evaluated. The evaluation requires specification of the right metrics and indicators for evaluation, such as consistency, completeness, usefulness, performance, reliability, accuracy, fit in the organisation or application context, functionality and other metrics for quality evaluation. The design evaluation method includes Observational (case study, field study) used in Publication 5; Analytical (static analysis, architecture analysis, optimisation and dynamic analysis), Experimental (controlled experiment, simulation), Testing (functional [black box]), Structural [white box] testing), and Descriptive (informed argument and scenarios) used in Publications 1 and 2.

Applied in this research process: The design of the artefacts was evaluated based on its usefulness and performance in guiding and supporting stakeholders during software design and development, as detailed in Chapter 4 (Sections 4.1 and 4.2), in which scenarios were used, and in Section 4.5 (Publication 5) detailed how case study was conducted.

4. **Research Contributions:** Effective design science research must provide a clear contribution to the areas of the design artefact, design construction knowledge (foundations), and design evaluation knowledge (methodologies). The main assessment of any design science research is based on new contributions, which are categorised into novelty, generality and significance of the design artefacts.

   Applied in this research process: The research contributions from this thesis are detailed in Chapter 1, Section 1.2, and Publication 1- 5.

5. **Research Rigour:** Rigour addresses how research is conducted. Design science research is based on the application of the rigorous method in the creation and evaluation of design artefacts. Furthermore, designed artefacts are often components of a human-machine problem-solving system. For such artefacts, knowledge of behavioural theories and empirical work are necessary to construct and evaluate such artefacts. Constructs, models, methods and instantiations must be exercised within appropriate environments.

   Applied in this research process: The research in this thesis involves the use of sustainability manifesto principles, known as the Karlskrona Manifesto principles for sustainability design (Becker *et al.*, 2015) and requirement engineering design methods for software sustainability (Penzenstadler, 2014)(Penzenstadler, 2016) in the design of the artefacts.

6. **Design as a Search Process:** Design science is inherently iterative. The search for the best or optimal design is often intractable for realistic information system problems. Heuristic search strategies produce feasible, good designs that can be implemented in the application environment. Design is essentially a search

process to discover an effective solution to a problem. Problem-solving can be viewed as utilising available means to reach desired ends while satisfying laws existing in the environment

Applied in this research process: The research questions were specified without a pre-stated solution method to address the research questions. Results from literature reviews and discussions with stakeholders were used to set requirements for the artefact design detailed in Publications 1-3 and also in Chapter 4.

7. **Communication of Research:** Design science research must be presented both to technology-oriented and management-oriented audiences. Technology-oriented audiences need sufficient detail to enable the described artefact to be constructed (implemented) and used within an appropriate organisational context.
Applied in this research process: The research results from this thesis were published in conferences and journal to reach a wider community of interested stakeholders.

5.3 Threat to Validity

This section details threat to validity in the research focusing on the construct, internal, external validity and issue of reliability.

Construct Validity: Threat to construct validity is related to what extent the operational measures that were studied really represent what the researcher have in mind (Runeson and Höst, 2009). The research questions are tailored to support the aim of investigating how to support and guide software sustainability design practices. Publication 1, 2 and 3 provided some of the challenges in software sustainability design practices and proposed the use of artifacts such as SSDC and FSSSD. Publication 4 presents a template for documenting software sustainability design best practice validated with industry experts such as software developers, product tester /integration engineer, requirement engineer and business analyst. To improve the construct validity, Publication 5 details how artifacts from this research were applied in case study, the result from the case study further helped to increase validity of the results from this research.

Internal validity: This is about the validity of results within, or internal to, a study. When the researcher is investigating whether one factor affects an investigated factor, there is a risk that the investigated factor is also affected by a third factor (Runeson and Höst, 2009). Research from literature were first used for the investigation of stakeholder's challenges in software sustainability design practices. The results from literature were combine with feedbacks from stakeholders to improve internal validity. Publication 4 and 5 details the combination of literature and feedbacks from stakeholders.

External Validity: This aspect is about how findings from research can be generalized and to what extent the findings are of interest to other people outside the investigated case (Runeson and Höst, 2009). Case study involving two different kinds of case were used

for validation of the artifacts in publication 5. This is to enable analytical generalization based on the results from the two case studies. Publication 4 also presents a template that was validated by stakeholders from varying companies which shows interest of stakeholders from different backgrounds. One limiting factor especially in the validation of artifacts is that only two case studies were used. Applying more case studies might also improve variety of feedbacks that can help to establish the general factors affecting sustainability in software design in the industry and help improve the artifacts to support software sustainability design practices.

Reliability: This aspect is concerned with to what extent the data and analysis are dependent on the specific researchers (Runeson and Höst, 2009). To mitigate the issue of reliability which can occur due to researcher's bias, triangulation among different researcher involve in the publications was applied. According to (Creswell and Miller, 2000) triangulation is an approach to ensure reliability of findings. During each stage of research from literature review, design of artifacts and validation of artifacts, discussion were held between researcher and supervisors to improve reliability of findings and results.

5.4 Limitation of Research

This research is carried out with the constraint of designing and developing artefacts to support and guide stakeholders in software sustainability design, development and measurement. Based on this scope, the research is limited to software sustainability design and development, covering only the SDLC phases.

The second limitation of this thesis is the number of design iterations for creating the last two artefacts (FSSSD and Template for Software Sustainability Requirement Elicitation Best Practice). The design process would have benefited from a third design and evaluation test iteration to include more software sustainability tools and methods to support stakeholders in using the framework and also to improve the template based on stakeholder feedback. However, the design iteration is currently ongoing, and the results will be published in a conference or journal article.

The final notable limitation is the number of case studies used in the evaluation of the design artefacts (SSDC, FSSSD and Template for Software Sustainability Requirement Elicitation Best Practice). In this thesis two case studies were used to show proof of concept. More industrial case studies would have provided more information and a wider context of application on how to improve the artefacts and create more extensive software sustainability design which can increase awareness for interested stakeholders.

6 Conclusion

Sustainability in software design, development and measurement is an area that requires concentrated effort in providing concrete guidelines for sustainability in software design. The design science research and case study were applied in this thesis to investigate how to support and guide stakeholders in software sustainability design, development and measurement. This chapter highlights the identified problems, proposed solutions and future research work.

One of the major issue identified from this thesis is the lack of understanding from practitioners in the industry about what sustainability means in software design and development. This issue is associated with a crucial challenge from academic research regarding the provision of common ground on how sustainability can be objectively engineered into software design and development. Specify practical cases that exemplify how processes, methods, tools and sustainability concepts are used for software sustainability engineering, from the requirement phase of SDLC to integration and maintenance phase. SSDC and FSSSD was created to address this issue in software sustainability design and development.

Furthermore, there is a challenge of what sustainability design principles to use during software design. The KMSD serves as a base for solving this problem through the proposed high-level and abstract principles. In order to make the KMSD principles applicable in a quantifiable way during software design, there is a need for research that translates those principles into software sustainability design guidelines. The SSDC guidelines are an example of using the KMSD to provide software sustainability design guidelines.

Also, the measures and measurement of software sustainability is an area with less research attention in providing metrics, indicators and measures that can be used in the evaluation of software sustainability design and measurement. There is a need for a software sustainability measurement framework that will provide a different scale of measurement and interpretation of the scales of software sustainability measures.

Overall, the research output from this thesis using the design science process are the created artefacts, namely, the Sustainable Business Goal Question Metric approach (Publication 1), SSDC and FSSSD, for Sustainability of Software System Design (Publication 2) and Template for Software Sustainability Requirement Elicitation Best Practice (Publication 4) to address the aforementioned problems in this thesis and the research questions.

Finally, in order to fully achieve software sustainability by design, there is a need for a knowledge base with practical examples of how sustainability has been applied in different software projects especially in industry. This can serve as a backbone for interested stakeholders to use and learn how to apply sustainability during software design and development.

6.1 Addressing Research Questions

- **Research Question 1 (RQ1):** How to elicit software sustainability requirements in software design?
  The S-BGQM approach, SSDC and FSSSD artefacts provide a structural guide on how stakeholders can elicit software sustainability requirements during software design. The application of the three artefacts is detailed in Chapter 5, Section 5.1, and Publication 1, 2 and 5.
- **Research Question 2 (RQ2):** How to measure and evaluate software system sustainability?
  FSSSD provides support and guidance for creating and selecting indicators, metrics and measures to evaluate software sustainability, as summarised in Publication 5 and Chapter 4. Publication 3 also details measures that have been used in the measurement of software sustainability.
- **Research Question 3 (RQ3):** How to record good practices for software sustainability design and development?
  The template for Software Sustainability Requirement Elicitation Best Practice, detailed in Chapter 5 and Publication 4, is a good example of how to document software sustainability design and development best practices.

6.2 Future Research

The artefacts created in this thesis will benefit from more extensive and comparative evaluation to improve their usefulness in supporting and guiding stakeholders through software sustainability design and development. Using more case studies from different application environments and contexts will provide a better understanding of the effects of applying the artefacts in software sustainability design. Transfer of research to industry via company collaboration in future will improve the current artefacts and enable a more user-friendly framework to support software sustainability design practices.

There is also a need for an additional design iteration to incorporate feedbacks on the current design artefacts to improve their effectiveness and efficiency during software sustainability design, development and measurement. This additional iteration will also improve the overall understanding of software sustainability design in the long term.

Currently, there is not enough scientific data on how to objectively create software sustainability measures with different scales of measurement and the interpretation of those scales of measurement. Hence, there is a need for a framework or model that can support the grounding of software sustainability measures in general software measurement theories.

Furthermore, there is need for interdisciplinary research towards grounding the meanings and perception of software sustainability design in research and theory with a common ground that consolidates existing research and extend current understanding software design in software engineering. In addition, to foster better application of different research theories and proposed methods for software sustainability design, a cross disciplinary collaboration with industry for transfer of research to practice will further enhance software sustainability design with feedback from practitioners in industry. This collaboration can also help to provide a common understanding for the role of stakeholders in software sustainability design both in research and practice.

References

Abdulsalam, S. *et al.* (2015) 'Using the Greenup, Powerup, and Speedup metrics to evaluate software energy efficiency', in *Proceedings of Sixth International Green and Sustainable Computing Conference (IGSC)*. doi: 10.1109/IGCC.2015.7393699.

Albertao, F. *et al.* (2010) 'Measuring the Sustainability Performance of Software Projects', *2010 IEEE 7th International Conference on E-Business Engineering*, pp. 369–373. doi: 10.1109/ICEBE.2010.26.

Amsel, N. *et al.* (2011) 'Toward sustainable software engineering (NIER Track)', *2011 33rd International Conference on Software Engineering (ICSE)*, pp. 976–979. doi: 10.1145/1985793.1985964.

Becker, C. (2014) 'Sustainability and longevity: Two sides of the same quality?', in *Proceedings of Third International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)*, pp. 1–6. doi: 10.13140/2.1.2214.0800.

Becker, C. *et al.* (2015) 'Sustainability Design and Software: The Karlskrona Manifesto', *Proceedings of the 37th International Conference on Software Engineering*, 2, pp. 467–476. doi: 10.1109/ICSE.2015.179.

Becker, C. *et al.* (2016a) 'Requirements: The key to sustainability', *IEEE Software*, 33(1), pp. 56–65. doi: 10.1109/MS.2015.158.

Becker, C. *et al.* (2016b) 'Requirements: The key to sustainability', *IEEE Software*, 33(1), pp. 56–65. doi: 10.1109/MS.2015.158.

Ben-Eli, M. (2015) 'Sustainability: definition and five core principles', *Sustainability Science*, 13(5), pp. 1337–1343. doi: 10.1007/s11625-018-0564-3.

Benyus, J. M. (1997) 'Biomimicry: Innovation Inspired by Nature', *Governance International Journal Of Policy And Administration*, (2002), p. 308. Available at: http://books.google.com/books?id=mDHKVQyJ94gC&pgis=1.

Bonini, S. and Görner, S. (2011) 'The business of sustainability : Putting it into practice', *Insights & Publications*, p. 6. Available at: www.mckinsey.com.

Bourque, P. and Fairley, R. E. (Dick) (2014) 'Guide to the Software Engineering Body of Knowledge SWEBOK, 3rd edition', *IEEE Computer Society Press*. Available at: https://cs.fit.edu/~kgallagher/Schtick/Serious/SWEBOKv3.pdf%0A.

Bozzelli, P., Gu, Q. and Lago, P. (2013) 'A systematic literature review on green software metrics', *VU Technical Report*. Available at: http://www.sis.uta.fi/~pt/TIEA5_Thesis_Course/Session_10_2013_02_18/SLR_Green Metrics.pdf.

Calero, C., Bertoa, M. F. and Angeles Moraga, M. (2013) 'A systematic literature review for software sustainability measures', in *Proceedings of the 2nd International Workshop on Green and Sustainable Software*, pp. 46–53. doi: 10.1109/GREENS.2013.6606421.

Calero, C., Moraga, M. A. and Bertoa, M. F. (2013) 'Towards a Software Product Sustainability Model', in *WSSSPE'1: First workshop on sustainable software for science: practice and experiences, SC'13, Denver, CO, USA.*, p. 4. Available at: http://arxiv.org/abs/1309.1640.

Calero, C. and Piattini, M. (2015) 'Introduction to Green in software engineering', *Green in Software Engineering*, pp. 1–327. doi: 10.1007/978-3-319-08581-4.

Chitchyan, R. *et al.* (2015) 'Evidencing sustainability design through examples', in *Proceedings of the Fourth International Workshop on Requirements Engineering for Sustainable Systems co-located with the 23rd IEEE International Requirements Engineering Conference (RE 2015)*, pp. 45–54.

Chitchyan, R. *et al.* (2016) 'Sustainability design in requirements engineering: state of practice', *38th International Conference on Software Engineering Companion (ICSE '16)*, pp. 533–542. Available at: http://eprints.hud.ac.uk/28747/.

Cohen, N., Robbins, P. and Denault, J.-F. (2012) 'Social Return on Investment [pdf] Available at : http://www.socialvalueuk.org/app/uploads/2016/03/The%20Guide%20to%20Social%20Return%20on%20Investment%202015.pdf [Accessed on 20-04-2019]', (January). doi: 10.4135/9781412973793.n132.

Cortese, A. D. and Rowe, D. (2000) 'Higher Education and Sustainability Overview [pdf] Available < https://uwosh.edu/sirt/wp-content/uploads/sites/86/2017/08/Definitions-of-Sustainability.pdf>[Accessed on 03-01-2019]'.

Council, N. R. (2011) *Sustainability and the U.S. EPA. National Research Council. Washington, DC [online] Available < https://doi.org/10.17226/13152> [Accessed on 02-02-2019]*, *National Academies Press*. The National Academies Press. doi: 10.5860/choice.49-4423.

Creswell, J. W. and Miller, D. L. (2000) 'Determining validity in qualitative inquiry', *Theory into Practice*, pp. 1–130. doi: 10.1207/s15430421tip3903_2.

Degrees, S. (2019) 'What is Sustainability. Sustainability Degrees. [online] Available < https://www.sustainabilitydegrees.com/what-is-sustainability/> [Accessed on 29-04-2019]'.

Diesendorf, M. (2000) 'Sustainability and sustainable development', in Dunphy, D, Benveniste, J, Griffiths, A and Sutton, P (eds) Sustainability: The corporate challenge of the 21st century, Sydney: Allen & Unwin, chap. 2, 19-37', pp. 19–37. doi:

10.4324/9781315442044-11.

Ehrenfeld, J. R. (2008) *Sustainability by Design: A Subversive Strategy for Transforming Our Consumer Culture*. Yale University Press New Haven and London.

EITO, E. I. T. O. (2002) 'The impact of ICT on sustainable development', pp. 250–283.

Enterprise, F. (2019) 'Flourishing Business Canvas [onlin] Available at: http://www.flourishingbusiness.org/the-toolkit-flourishing-business-canvas/ [Accessed on 14-03-2019]', p. 2016.

Erdélyi, K. (2013) 'Special factors of development of green software supporting eco sustainability', *Proceedings of IEEE 11th International Symposium on Intelligent Systems and Informatics -SISY 2013*, pp. 337–340. doi: 10.1109/SISY.2013.6662597.

Erdmann, L. *et al.* (2004) 'The Future Impact of ICTs on Environmental Sustainability', *IPTS Publications*, p. 68. Available at: ftp://ftp.jrc.es/pub/EURdoc/eur21384en.pdf.

Ericsson (2014) 'Technology for Good [pdf] Available online <https://www.ericsson.com/assets/local/about-ericsson/sustainability-and-corporate-responsibility/documents/2015-corporate-responsibility-and-sustainability-report.pdf> [Accessed on 30-11-2017]'. doi: 10.1089/SUS.2013.9868.

Freeman, P. (1980) 'The Central Role of Design in Software Engineering: Implications for Research', *Software Engineering: Research Directions*, pp. 121–132.

Giovannoni, E. and Fabietti, G. (2013) 'What Is Sustainability? A Review of the Concept and Its Applications', (2010), pp. 21–41. doi: 10.1007/978-3-319-02168-3.

Glaser, B. G. and Strauss, A. L. (1967) *The Discovery of Grounded Theory Strategies for Qualitative Research*.

Goodland, R. (2002) *Encyclopedia of Global Environmental Change, chapter Sustainability: Human, Social, Economic and Environmental*. Wiley & Sons.

Gordieiev, O., Kharchenko, V. and Fusani, M. (2016) *Software Quality Standards and Models Evolution: Greenness and Reliability Issues, Information and Communication Technologies in Education, Research, and Industrial Applications. ICTERI 2015. Communications in Computer and Information Science*. doi: 10.1007/978-3-319-13206-8.

Groher, I. and Weinreich, R. (2017) 'An Interview Study on Sustainability Concerns in Software Development Projects', in *proceedings of 43rd Euromicro Conference on Software Engineering and Advanced Applications*, pp. 350–358. doi: 10.1109/SEAA.2017.70.

Guinée, J. (2016) ‘Life Cycle Sustainability Assessment: What Is It and What Are Its Challenges?’, *Springer International Publishing*, pp. 45–65. doi: 10.1007/978-3-319-20571-7.

Heinberg, R. (2010) ‘What Is Sustainability? [pdf] Available <http://www.postcarbon.org/publications/what-is-sustainability/> [Accessed on 29-04-2019]’, *Watershed Media*. Available at: http://www.postcarbon.org/publications/what-is-sustainability/.

Hevner, A. and Chatterjee, S. (2010) *Design Research in Information Systems*. Springer New York Dordrecht Heidelberg London. doi: 10.1007/978-1-4419-5653-8.

Hevner, A. R. *et al.* (2004) ‘Design Science in Information Systems Research’, *MIS Quarterly*, 28(1), pp. 75–105. doi: 10.2307/25148625.

IBM (2010) ‘Global CEO Study: The enterprise of the future’, *Available online: Https://www-935.ibm.com/services/uk/gbs/pdf/ibm_ceo_study_2008.pdf Accessed on 11-10-2017*, p. 76.

Idio, M. R. (2014) ‘Measuring Sustainability Impact of Software’, *International Journal of Computer Trends and Technology (IJCTT)*, 16(1), pp. 5–7.

Ilstedt, S., Eriksson, E. and Hesselgren, M. I. A. (2017) ‘Sustainable Lifestyles – How Values Affect Sustainable Practises’, 7(7).

Jannat, U. K. (2016) ‘Green Software Engineering Adaption In Requirement Elicitation Process’, *International Journal of Scientifc & Technology Research*, 5(08), pp. 94–98.

Johann, T. *et al.* (2012) ‘How to measure energy-efficiency of software: Metrics and measurement results’, in *Proceedings of 1st International Workshop on Green and Sustainable Software, GREENS*, pp. 51–54. doi: 10.1109/GREENS.2012.6224256.

Kern, E. *et al.* (2013) ‘Green Software and Green Software Engineering – Definitions , Measurements , and Quality Aspects’, pp. 87–94.

Kocak, S. A. (2013) ‘Green Software Development and Design for Environmental Sustainability’, in *ESEM-IDOISE Doctoral Symosium*.

Koçak, S. A., Alptekin, G. I. and Bener, A. B. (2014) ‘Evaluation of software product quality attributes and environmental attributes using ANP decision framework’, in *Proceedings of the Third International Workshop on Requirements Engineering for Sustainable Systems co-located with 22nd International Conference on Requirements Engineering (RE 2014)*, pp. 37–44. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84908292804&partnerID=40&md5=03d9e18dfe2eb69745df0031fc89be23.

Koziolek, H. (2011) ‘Sustainability evaluation of software architectures: A systematic review’, in *Proceedings of the joint ACM SIGSOFT conference on quality of software architectures. Boulder, Colorado, USA*, pp. 3–12. doi: 10.1145/2000259.2000263.

Kuhlman, T. and Farrington, J. (2010) ‘What is sustainability?’, *Sustainability*, 2(11), pp. 3436–3448. doi: 10.3390/su2113436.

Lago, P. *et al.* (2015) ‘Framing Sustainability as a Property of Software Quality’, *Communications of the ACM*, 55(11), pp. 56–64. doi: 10.1021/ac60289a702.

Lami, G. and Buglione, L. (2012) ‘Measuring software sustainability from a process-centric perspective’, in *Proceedings of the 2012 Joint Conf. of the 22nd Int. Workshop on Software Measurement and the 2012 7th Int. Conf. on Software Process and Product Measurement, IWSM-MENSURA 2012*, pp. 53–59. doi: 10.1109/IWSM-MENSURA.2012.16.

Macdonald, C. (2012) ‘Understanding Participatory Action Research: A Qualitative Research Methodology Option.’, *The Canadian Journal of Action Research*, 13(2), pp. 34–50.

Mahaux, M. and Canon, C. (2012) ‘Integrating the Complexity of Sustainability in Requirements Engineering’, *First International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)*.

Mahaux, M., Heymans, P. and Saval, G. (2011) ‘Discovering Sustainability Requirements: An Experience Report’, *Proceedings of Requirements Engineering: Foundation for Software Quality - 17th International Working Conference, REFSQ*, (January), pp. 247–261. doi: https://doi.org/10.1007/978-3-642-19858-8_3.

Mann, S. (2007) ‘Computing for Sustainability [online] Available at https://computingforsustainability.com/2007/08/27/biomimicry-in-software-engineering-a-super-system-metaphor/ [Accessed on 23-01-2019]’.

Microsoft (2015) ‘Microsoft 2015 Citizenship Report. [PDF] Available at: http://download.microsoft.com/download/7/3/6/736CED21-9D8B-4CBB-98E8-DCBAE7026251/Microsoft%202015%20Citizenship%20Report.pdf’.

Mohajan, H. K. (2018) ‘Qualitative Research Methodology in Social Sciences and Related Subjects’, *Journal of Economic Development, Environment and People*, 7(85654), pp. 23–48. Available at: https://mpra.ub.uni-muenchen.de/85654/1/MPRA_paper_85654.pdf.

Moir, S. and Carter, K. (2012) ‘Diagrammatic representations of sustainability - A review and synthesis’, *Association of Researchers in Construction Management, ARCOM 2012 - Proceedings of the 28th Annual Conference*, 2(September), pp. 1479–1489.

Musthaler, L. (2014) ‘Energy-aware software design can reduce energy consumption by 30% to 90% [online] Available <https://www.networkworld.com/article/2861005/energy-aware-software-design-can-reduce-energy-consumption-by-30-to-90.html > [Accessed on 15-04-2019]’, *Network World*. Available at: http://www.networkworld.com/article/2861005/green-it/energy-aware-software-design-can-reduce-energy-consumption-by-30-to-90.html.

Nations, U. (2005) ‘World Summit Outcome. General Assembly’, (October), pp. 1–38. Available at: https://generalassemb.ly/design.

Naumann, S. *et al.* (2011) ‘The GREENSOFT Model: A reference model for green and sustainable software and its engineering’, *Sustainable Computing: Informatics and Systems*, 1(4), pp. 294–304. doi: 10.1016/j.suscom.2011.06.004.

Nidumolu, R., Prahalad, C. . and Rangaswami, M. . (2013) ‘Why sustainability is now the key driver of innovation’, *IEEE Engineering Management Review*. doi: 10.1109/EMR.2013.6601104.

Noureddine, A. *et al.* (2012) ‘Runtime monitoring of software energy hotspots’, in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 160–169. doi: 10.1145/2351676.2351699.

Oyedeji, S. *et al.* (2019) ‘Validation Study of a Framework for Sustainable Software System Design and Development’, in *Proceedings of 6th International Conference on ICT for Sustainability, (ICT4S)*.

Oyedeji, S. and Penzenstadler, B. (2018) ‘Karlskrona Manifesto: Software requirement engineering good practices’, *Proceedings of the 7th International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy 2018) co-located with the 26th International Conference on Requirements Engineering (RE 2018)*, 2223, pp. 15–23.

Oyedeji, S. and Penzenstadler, B. (2019) ‘Experiences from Applying the Karlskrona Manifesto Principles for Sustainability in Software System Design’, in *Proceedings of the 8th International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy 2018) co-located with the 27th International Conference on Requirements Engineering (RE 2019)*.

Oyedeji, S., Seffah, A. and Penzenstadler, B. (2017) ‘Sustainability Quantification in Requirements Informing Design’, *6th International Workshop on Requirements Engineering for Sustainable Systems*, i. Available at: http://ceur-ws.org/Vol-1944/paper6.pdf.

Oyedeji, S., Seffah, A. and Penzenstadler, B. (2018a) ‘A catalogue supporting software sustainability design’, *Sustainability*, 10(7), pp. 1–30. doi: 10.3390/su10072296.

Oyedeji, S., Seffah, A. and Penzenstadler, B. (2018b) 'Classifying the Measures of Software Sustainability', *Proceedings of the 4th International Workshop on Measurement and Metrics for Green and Sustainable Software Systems co-located with 12th International Symposium on Empirical Software Engineering and Measurement (ESEM 2018)*.

Peffers, K. *et al.* (2007) 'A Design Science Research Methodology for Information Systems Research', *Journal of Management Information Systems*, 24(3), pp. 45–78. doi: 10.2307/40398896.

Penzenstadler, B. (2014) 'Infusing green: Requirements engineering for green in and through software systems', *3rd Intl. Workshop on Requirements Engineering for Sustainable Systems, 2014*, 1216(1), pp. 44–53.

Penzenstadler, B. (2015) 'Sustainability and Requirements: A Manifesto', *IEEE Software*, 32(5), pp. 90–92. doi: 10.1109/MS.2015.114.

Penzenstadler, B. (2016) 'Goal Model Requirements Engineering for Sustainability', *Lect. slides guest course Softw. Eng. Sustain. Lappeenranta Univ. Technol. Slides available online at http//birgit.penzenstadler.de/teach/LUT.html.*

Penzenstadler, B. *et al.* (2018) 'Software engineering for sustainability', in *Requirements Engineering Conference (RE) 2018 IEEE 26th International*, pp. 304–314. doi: 10.9774/gleaf.9781315465975_13.

Penzenstadler, B. and Femmer, H. (2013) 'A generic model for sustainability with process- and product-specific instances', in *Proceedings of the 2013 Workshop on Green in Software Engineering, Green by Software Engineering*, pp. 3–7. doi: 10.1145/2451605.2451609.

Penzenstadler, B., Mahaux, M. and Salinesi, C. (2014) 'RE4SuSy', in *3rd International Workshop on Requirements Engineering for Sustainable Systems*. Available at: http://ceur-ws.org/Vol-1944/preface.pdf.

Penzenstadler, B., Mahaux, M. and Salinesi, C. (2015) 'RE4SuSy', in *4th International Workshop on Requirements Engineering for Sustainable Systems, Part of the GREENS Alliance*, pp. 4–7. Available at: http://ceur-ws.org/Vol-1944/preface.pdf.

Penzenstadler, B., Salinesi, C. and Ruzanna, C. (2017) 'RE4SuSy', in *6th International Workshop on Requirements Engineering for Sustainable Systems*. Available at: http://ceur-ws.org/Vol-1944/preface.pdf.

Penzenstadler, B., Tomlinson, B. and Richardson, D. (2012) 'RE4ES: Support Environmental Sustainability by Requirements Engineering', *First International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)*. Available at:

http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:RE4ES+:+Support+Environmental+Sustainability+by+Requirements+Engineering#0.

Penzenstadler, B. and Venters, C. C. (2018) 'Software engineering for sustainability', *Digital Technology and Sustainability*, pp. 103–121. doi: 10.9774/gleaf.9781315465975_13.

Robillard, M. P. (2016) 'Sustainable Software Design', in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 920–923. doi: 10.1145/2950290.2983983.

Rondeau, E., Lepage, F. and Georges, J. (2015) 'Measurements and Sustainability, Green Information Tech- nology. A Sustainable Approach', *Elsevier*, p. pp.29-59.

Runeson, P. and Höst, M. (2009) 'Guidelines for conducting and reporting case study research in software engineering', *Empirical Software Engineering*, 14(2), pp. 131–164. doi: 10.1007/s10664-008-9102-8.

Seacord, R. *et al.* (2013) 'Measuring Software Sustainability', in *Proceedings International Conference on Software Maintenance ICSM 2003*. doi: 10.1017/CBO9781107415324.004.

Seyff, N., Betz, S., Groher, I., *et al.* (2018) 'Crowd-focused semi-automated requirements engineering for evolution towards sustainability', *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, pp. 370–375. doi: 10.1109/RE.2018.00-23.

Seyff, N., Betz, S., Duboc, L., *et al.* (2018) 'Tailoring requirements negotiation to sustainability', *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, pp. 304–314. doi: 10.1109/RE.2018.00038.

Shedroff, N. (2009) *Design is the Problem:The Future of Design Must be Sustainable*. Rosenfeld Media. Available at: www.rosenfeldmedia.com.

Spinellis, D. (2017) 'The Social Responsibility of Software Development', *IEEE Software*, 34(2), pp. 4–6.

Starman, A. B. (2013) 'The case study as a type of qualitative research.', *Journal of Contemporary Educational Studies*, 64(1), pp. 28–43.

Tainter, J. A. (2006) 'Social complexity and sustainability', *Ecological Complexity*, 3(2), pp. 91–103. doi: 10.1016/j.ecocom.2005.07.004.

Tate, K. (2005) *Sustainable Software Development: An Agile Perspective*, *Addison-Wesley Professional*.

Thwink.org (2019) ‘Sustainability Definition. [online] Available < http://thwink.org/sustain/glossary/Sustainability.htm> [Accessed on 06-02-2019]’, *Web Resource*.

Tilbury, E. D. *et al.* (2002) *Education and Sustainability: Responding to the Global Challenge, International Journal of Sustainability in Higher Education*. doi: 10.1108/ijshe.2003.24904bae.007.

UN General Assembly (1987) ‘Report of the World Commission on Environment and Development: Our Common Future’, *Report of the World Commission on Environment and Development: Our Common Future*.

United Nations (2013) *World Economic and Social Survey 2013. New York: Department for Economic and Social Affairs*. doi: 10.1016/j.compind.2010.10.001.

United Nations (2015) ‘Sustainable Development Goals [online] Available at: <https://www.un.org/sustainabledevelopment/sustainable-development-goals/> [Accessed on 28-12-2018]’, (September 2000), pp. 8–23.

Venters, C. C. *et al.* (2014) ‘Software sustainability: The modern tower of babel’, *3rd International Workshop on Requirements Engineering for Sustainable Systems.CEUR Workshop Proceedings*, 1216, pp. 7–12. doi: http://ceur-ws.org/Vol-1216/.

Venters, C. C. *et al.* (2017) ‘Characterising sustainability requirements: A new species red herring or just an odd fish?’, *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track, ICSE-SEIS 2017*, pp. 3–12. doi: 10.1109/ICSE-SEIS.2017.2.

Venters, C. C. *et al.* (2018) ‘Software sustainability: Research and practice from a software architecture viewpoint’, *Journal of Systems and Software*, 138, pp. 174–188. doi: 10.1016/j.jss.2017.12.026.

Wiersum, K. F. (1995) ‘200 years of sustainability in forestry: Lessons from history’, *Environmental Management*. Springer, 19(3), p. 321.

Wolfram, N., Lago, P. and Osborne, F. (2017) ‘Sustainability in Software Engineering’, *2017 Sustainable Internet and ICT for Sustainability (SustainIT)*. IFIP. doi: 10.1109/CSEET.2011.5876124.

Wu, J. *et al.* (2018) ‘Information and communications technologies for sustainable development goals: State-of-the-art, needs and perspectives’, *IEEE Communications Surveys and Tutorials*, 20(3), pp. 2389–2406. doi: 10.1109/COMST.2018.2812301.

Yin, R. K. (1984) *Case Study Research -Design and Methods*. Second Edi. Sage Publication. doi: 10.1201/b16851-12.

Zainal, Z. (2007) ‘Case Study as a Research Method’, *UTM Journal of Humanities*, 5, pp. 1–15. doi: 10.4135/9781473915480.n2.

Publication I

S. Oyedeji, A. Seffah, and B. Penzenstadler
Sustainability Quantification in Requirements Informing Design

In: *6th International Workshop on Requirement Engineering for Sustainable System co-located with the 25th International Conference on Requirements Engineering*

Reprinted with permission from
CEUR Workshop Proceedings
Vol. 1944, pp. 34-43, 2017

Sustainability Quantification in Requirements Informing Design

Shola Oyedeji
Department of Software Engineering
Lappeenranta University of Technology
Lappeenranta, Finland
shola.oyedeji@lut.fi

Ahmed Seffah
Department of Software Engineering
Lappeenranta University of Technology
Lappeenranta, Finland
ahmed.seffah@lut.fi

Birgit Penzenstadler
Department of Computer Engineering and Computer Science
California State University Long Beach
Long Beach, California, USA
birgit.penzenstadler@csulb.edu

***Abstract*— Sustainability has been defined with different perceptions and from diverse dimensions making it an ambiguous concept to objectively engineer and integrate into software development lifecycle. Although a large body of knowledge already exists on what sustainability is and isn't, little research has explored how to quantify sustainability. How can the definitions and perceptions of sustainability from software engineering and other fields be turned into requirements, effective measures that quantify sustainability and most importantly can inform a "sustainability by design" approach? What are the measures and measurement scale of sustainability? Our long-term research goal is to answer such questions and similar ones. In this position paper, we summarize our investigations and pave the road for a theoretical ground of sustainability quantification in software development and measurement. The goal is to foster research and standardization initiatives on sustainability as a quality attribute and sustainability by design.**

Keywords: *Software Sustainability, Sustainability Requirements, Software Measurement, Software Development, Sustainability Metrics, Software Design*

I. INTRODUCTION

In a broad sense, sustainability is "the capacity to endure" [1]. In software engineering, sustainability has been introduced from different dimensions with diverse perceptions and definitions. Sustainability can be differentiated into several dimensions including environmental, human, social, and economic. According to Becker et al. [2] sustainability dimensions are interdependent and cumulative - first, second and third order effects from each dimension will bleed into each other. Sustainability consideration as a non-functional requirement like security, usability, reliability can help reduce a software system's first order impacts which will also aid reduction of second and third-order impacts of software systems. By doing so, developers have the potential to considerably improve software systems sustainability from the requirement engineering stage onwards [3]. This also requires measures informing how well the development process produces sustainable software [4].

The fundamental question is how to quantify sustainability not only for software products, systems and services but also for the entire digital ecosystem created by the system of software systems? This research aims to serve different communities, though there is still need to conduct empirical studies to validate these benefits. Quantifying sustainability in software systems will encourage software engineering community to develop processes, tools and new metrics to assess sustainability of software system like the other quality attributes. It will help companies, organizations and managers to easily adopt and institutionalize sustainability in their mainstream software development and management processes, assess objectively the cost-benefit while creating a business model associated with sustainability of their software system.

Furthermore, it will guide standardization bodies like ISO and governmental agencies to enact standards and policies for software system sustainability. For example, what is the minimum sustainability level of a software system to get certain accreditation like we do with security today? It will also make the society and people more aware about the impact of software systems when developing and using it; one example is the categorization of a fridge based on its level of greenness (energy usage) such as A+, A++. Shall we adopt the same approach in software engineering?

Kocak et al. [5] stated that software development industry is now getting pressure from regulators to consider green certification. As an answer to this pressure, green attributes of software products should be defined as quality factor. Then, the biggest challenge facing companies is how to integrate sustainability into their engineering practices when knowing the lack of consensus on what sustainability means in software systems and how it can be quantified and measured.

Quantification of sustainability requires that it should be considered among the six divisions in ISO standards SQuaRE Model such as: Quality Management Division, Quality Model Division, Quality Measurement Division, Quality Requirements Division, Quality Evaluation Division, SQuaRE Extension Division [6]. By including sustainability in such standardization framework, sustainability may be considered more effectively in the industry. This is not really the case today. One starting point towards this, is to turn the current meanings, perceptions, and beliefs into requirements, factors, measurable criteria and tangible measures.

This paper presents the early results of an ongoing research that aims to build a theoretical ground for

sustainability requirement quantification in software development. Hopefully, the paper can stimulate a discussion as a means of getting feedbacks for further investigations.

The remainder of this paper is as follow. The next section provides various sustainability definitions for requirements. Section III traces the research trends and outcomes from requirement engineering domain. Section IV discusses sustainability in software measurement and propose an approach for it. Section V details the proposed approach with an example. Section VI contains the conclusion with remarks for future work.

II. SUSTAINABILITY DEFINITIONS FOR REQUIREMENT

The varying definitions of sustainability show there are diverse opinions about what is sustainability. This makes it harder to define especially when applied to software systems. Still, these definitions provide a basis to start grounding sustainability in software engineering research and practices. Some clarity is needed as to how to quantify sustainability in software systems in term of quantifiable variables in order to be able to access and evaluate sustainability of software systems.

Sustainable software has been viewed from three angles [7] as:

(1) Long lasting software which relates to how well a piece of software will be able to cope with changes;
(2) Lean software that require less hardware and reduces its own power consumption (energy efficient);
(3) Software for sustainable humans as software that induces sustainable human behavior.

This definition leads to three measurable concerns that we should consider during requirement: energy efficiency, longevity and user experiences.

Venters et al. [8] explore emerging definitions of software sustainability from different angles in the field of computational science and engineering in order to contribute to the question, what is software sustainability? They stated that in software engineering, longevity and maintenance are the two most important factors for understanding sustainability. Their perception is based on the Oxford English dictionary definition for sustainability 'the quality of being sustained', where sustained can be defined as 'capable of being endured' and 'capable of being 'maintained'.

This work highlighted the importance of longevity and maintenance for the requirement of sustainability.

Heiko Koziolek [9] define sustainability of software systems from the perspective of software architecture as long living system that should last for more than 15 years and can be cost-efficiently maintained and evolved over its entire life-cycle. This also supports the requirement of longevity and maintainability.

Tainter [10] introduces sustainability as an active condition of problem solving, not a passive consequence of consuming less resources. To define sustainability in specific context the questions should be to sustain what, for whom, how long and at what cost? Applying Tainter's definition to software systems, it will help frame definition of sustainability into context in order to understand what the boundaries are in a system.

Seacord et al. [11] defined software sustainability as the 'ability to modify a software system based on customer needs and deploy these modifications,' which means sustainability is the quality of conforming to user specification. Modifiability is the key requirement from this definition.

Harris and Goodwin [2] describe sustainability as system that must achieve fairness in distribution and opportunity, adequate provision of social services, including health and education, gender equity, and political accountability and participation. Their definition focus on social sustainability relating to how well a system can cater for different user needs irrespective of their condition. The definition highlights the requirement for accessibility.

Naumann et al. [12] defined sustainable software as 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. Base on this definition the main requirements for sustainability can be derived from the economic, environment, social and individual dimensions of sustainability.

Table 1 summarizes the most cited definitions and identifies the key requirements.

TABLE I. DEFINITION SUMMARY AND REQUIRMENTS

Author	Definition	Requirement
M. R. Idio [7]	Long lasting and Lean software, Software for sustainable humans	Energy efficiency, Longevity and User Experiences.
Venters et al. [8]	Sustainability is the quality of being sustained. Longevity and maintenance are the two most important factors for understanding sustainability	Longevity and Maintenance
Heiko Koziolek [9]	Long living system that should last for more than 15 years and can be cost-efficiently maintained and evolved over its entire life-cycle.	Longevity and Maintenance
Seacord et al. [11]	Ability to modify a software system based on customer needs and deploy these modifications	Modifiability
Harris and Goodwin [2]	Sustainability as system that must achieve fairness in distribution and opportunity, adequate provision of social services	Accessibility
Naumann et al. [12]	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	Economic, environment, social and individual

Author	Definition	Requirement
Tainter [10]	To define sustainability in specific context the questions should be to sustain what, for whom, how long and at what cost?	Sustainability is a requirement within a certain context. It requires the specification of the context

III. SUSTAINABILITY IN REQUIREMENT ENGINEERING

The following are some of the research work in the domain of requirement engineering for sustainability in software systems.

Raturi et al. [13] focused on how to develop sustainability as a non-functional requirement (NFR) using NFR framework informed by sustainability models and how it can be used to correctly obtain and describe sustainability related requirements of the software system to be developed. The sustainability model has five dimensions (Human, Social, Economic, Environmental and Technical sustainability).

Penzenstadler et al. [14] also support the consideration of sustainability as a nonfunctional requirement like safety and security that are considered as a system quality attribute.

Mahaux et al. [15] highlights the fact that requirements engineering has a major role to play for making software last long by reducing the impact of development and disposal phase.

Roher et al. [16] concerned with the lack of software engineering teams including environmental sustainability during software development proposed the use of sustainability requirement patterns (SRPs) as a guide for software engineers to elicit sustainability requirements.

Becker et al [3] explains the crucial role of requirements not only for software systems but also for how requirement for sustainability can also impact on the social-economic and natural environment. The two case studies presented by the authors' shows the importance of requirement in sustainability design.

Based on the above research, there are three major issues for quantifying sustainability during the requirement stage as seen in the summary in Table I and section III:

- First, different research suggests different definitions, so there is no consensus definition.
- Second, the proposed definitions are either too complex or focus mainly a particular dimension of sustainability.
- Third, there is no central framework that is pivotal to the quantification of sustainability.

This shows there is need for discussing and coming to a consensus by researchers interested in sustainability of software systems. This can enable development of a central formwork that would support the addition of sustainability into the SQuaRE Model [6]. We believe this will foster a focused research towards better quantification of sustainability for software system. It will also encourage research on how best to incorporate management goals and requirements in the adoption of sustainability for software system design and development.

IV. SUSTAINABILITY MEASURES

Sustainability is still not fully explored in the field of software measurement. These are the different works on quantifying sustainability that have been done so far and also attempts to measure sustainability.

Lami et al. [17] stated there are few studies on 'what' aspects of sustainability to measure and 'how' to do it. Calero et al. [18] highlighted that nowadays, sustainability is a key factor that should be considered in the software quality models, though there is less research channeled towards it. Seacord et al. [11] indicated that planning and management of software sustainment is impaired by a lack of consistently applied, practical measures. Without these measures, it is difficult to determine the effect of efforts to improve sustainment practices.

Johann et al. [19] presents a generic metric to measure software energy efficiency and a method to apply it in software engineering process using the formula "Useful Work Done/Used Energy."

Krisztina Erdélyi [20] studies the lifecycle activities of software development with focus on environmental protection by proposing a formula to calculate software waste to encourage the development of green software.

Albertao et al. [4] proposed software engineering metrics based on software quality like reusability, portability, supportability, performance as a way for measuring the sustainability performance of software projects.

Bozzelli et al. [21] paper focused on describing and classifying metrics related to software "greenness" present in the software engineering literature through systematic literature review in order to analyze the evolution of those metrics, in terms of type, context, and evaluation methods highlighting metric types like energy, performance, utilization, software energy consumption.

One of the most referenced model for developing and measuring sustainable software is the Greensoft Model by Naumann et al. [22]. It is a conceptual reference model for "Green Software." The Greensoft model has the objective to support software developers, administrators, and software users in creating, maintaining, and using software in a more sustainable way but lacks the clarity and practical examples of how this model can be implemented for software system development. The key to measuring sustainability of software system requires quantifiable variables that can be applied to all sustainability dimensions in relation to software system development.

Thus, a new proposed approach; Sustainable Business Goal Question Metric (S-BGQM) is introduced here. It encourages the incorporation of sustainability during the entire software system development engineering processes.

S-BGQM is influenced by work from [23] and [24]. It combines results from the software requirement engineering process [23] into the design and development process. It is formed by two major components; the Sustainable Business Assessment and the Goal Question Metric. Figure 1 portrays

S-BGQM. All artefacts in the sustainable business assessment component provides support for all activities in the Goal Question Metric component of S-BGQM.

In the Sustainable Business Assessment, analysis of information in the sustainable business canvas leads to creation of sustainability goals. These goals are categorized into business, usage and system goals with consideration of sustainability that serve as a requirement for measurement. Based on this categorization, a set of questions are generated to characterize all those goals.

System vision and sustainability analysis provide a quick overview of the software system first, second and third order impacts based on those goals. And it provides information useful for specifying the right metric to evaluate the software system.

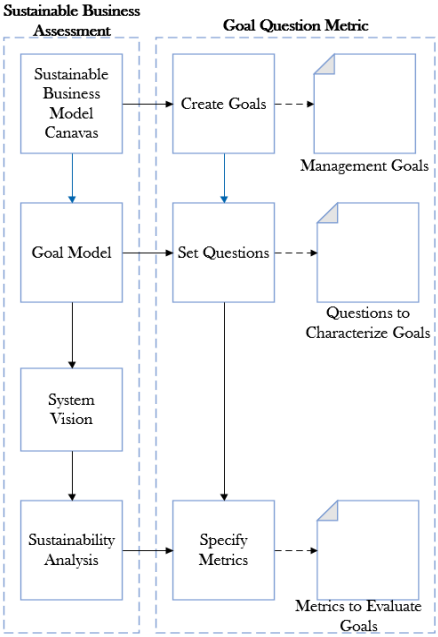


Figure 1 Sustainable Business Goal Metric Process Flow (S-BGQM)

The Sustainable Business Assessment component involves the following (See Figure 1):

- Sustainable Business Model Canvas: The Business Canvas incorporates sustainability considerations during business model design. It allows users to describe, design, challenge, invent, and pivot their business model with sustainability consideration [25] [26].
- Goal Model: It shows comprehensive and holistic goals of the organization or company in relation to the software under development from the economic, social and environmental perspective represented in business goal, usage goal and system goal [27].
- System Vision: It provides an overview of the whole system and how it interacts with different external components and its potential users based on the agreement of all stakeholders [28].
- Sustainability Analysis: Sustainability analysis describe the system from sustainability perspective by considering sustainability purpose of the system, impact the system has on environment as well as sustainability goal and constraint of the system [29].

The Goal Question Metric (GQM) component covers the following (See Figure 1):

- Tracing and measurement of system goals based on the result from the sustainable business assessment.
- Allows software engineers/ managers and company to define questions that can be used to evaluate their software system goals
- Choose appropriate metrics that can be used to measure their software system base the questions with consideration of sustainability.

These metrics are categorized according the sustainability dimensions as discussed by Raturi et al. [30] and Penzenstadler and Femmer [31] (Economic, Environmental, Social, Individual and Technical Sustainability).

Table II portrays metrics and their categorization according to the five dimensions of sustainability. The metrics samples presented in the GQM component give managers simple yardsticks to calibrate how well their company is doing in terms of resource consumption while extracting more value from their processes. The metrics support decision-making by providing a mechanism for benchmarking performance, tracking improvement over time, evaluating products and processes, and developing strategies for improvement.

TABLE II. METRIC CATEGORIZATION

Category	Metric	Description
Technical	BMI=Number of problems close/number of problems arrival *100	Backlog Management index (BMI) is a workload statement for software maintenance. It is related to both the rate of defect arrivals and the rate at which fixes for reported problems become available.
	Rework Metric	The total number of functions modified per commit related to adding a new feature/function. The "extensibility" of a system is generally the ability of the system to tolerate additional features or functionality with little or no required rework.
Economy	BMI=Number of problems close/number of problems arrival *100	Same as the above BMI
	Defect Density= Total defects/Size	The value of the total defects which are known to the size of the software product

Category	Metric	Description
		calculated.
	Net Cost	The Budgeted Capital - Total Capital Spent
Environment	BMI=Number of problems close/number of problems arrival *100	Same as the above BMI
	Defect Density= Total defects/Size	Same as the above Defect Density
	Energy efficiency	Useful work done/Used Energy
Social	Gateway metric (1=Task success and 0= Task failure)	The amount of successful task completed
	Defect Density= Total defects/Size	Same as the above Defect Density
	Net working hours	Budgeted hours - Total working hours
Individual	Gateway metric (1=Task success and 0= Task failure)	Same as the above Gateway metric
	Defect Density= Total defects/Size	Same as the above Defect Density

V. S-BGQM Preliminary Study Based on Information Researched Online

The preliminary study described here provides an example of how S-BGQM, as a way of quantifying sustainability works during requirements. A sample project where the project team proposed development of car sharing system called ShareVoyage for students in City of Lappeenranta is presented. It is an online web platform for group shopping and also to share unused foods.

The following seven steps process demonstrate how S-BGQM works while illustrating the different artifacts such as the Sustainable Business Canvas, Goal Model, System vision sustainability analysis of the system and metric worksheet.

1. Create Sustainable Busines Canvas. Figure 3 is an example of a canvas created in this study.

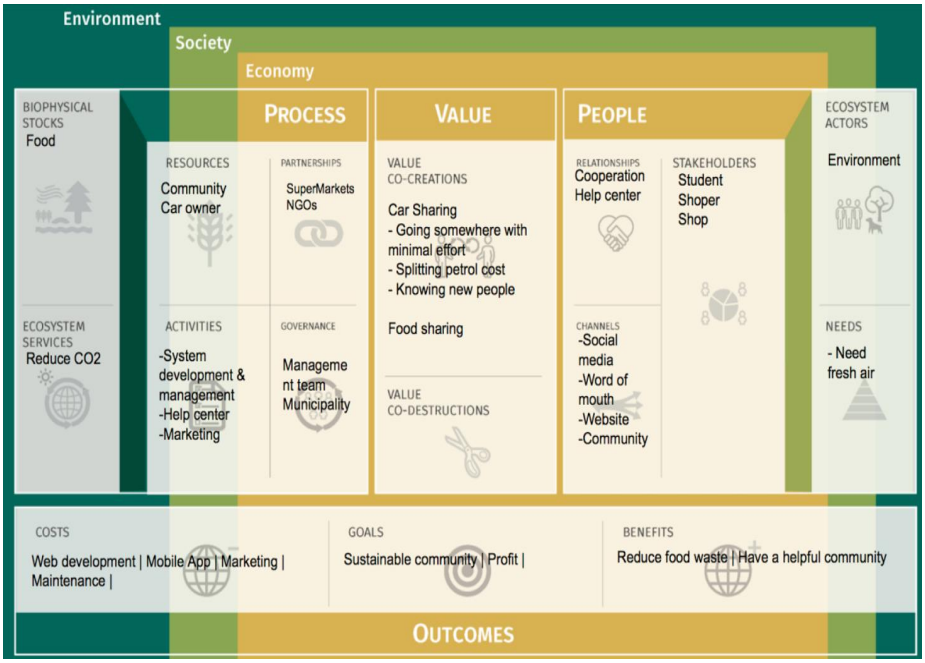


Figure 2. Sustainability Business Model Canvas (Sustainable Business Assessment)

2. Measurable management goals are created based on the information derived from the sustainable business canvas (see Figure 2). These are the goals derived based on the contents from the Canvas :
   - Reduce C02
   - Encourage car sharing
   - Reduce food waste by encouraging food sharing
   - Promote sustainable community

3. All the goals from step 2 are divided into three in the Goal Model phase show the business, usage and system goal of the software system. This division of goals serves as a means of proper classification for easier measurement after system development. Goal model is the basis for early conflict identification and resolution in the system development. Figure 3 shows the details of Goal Model.

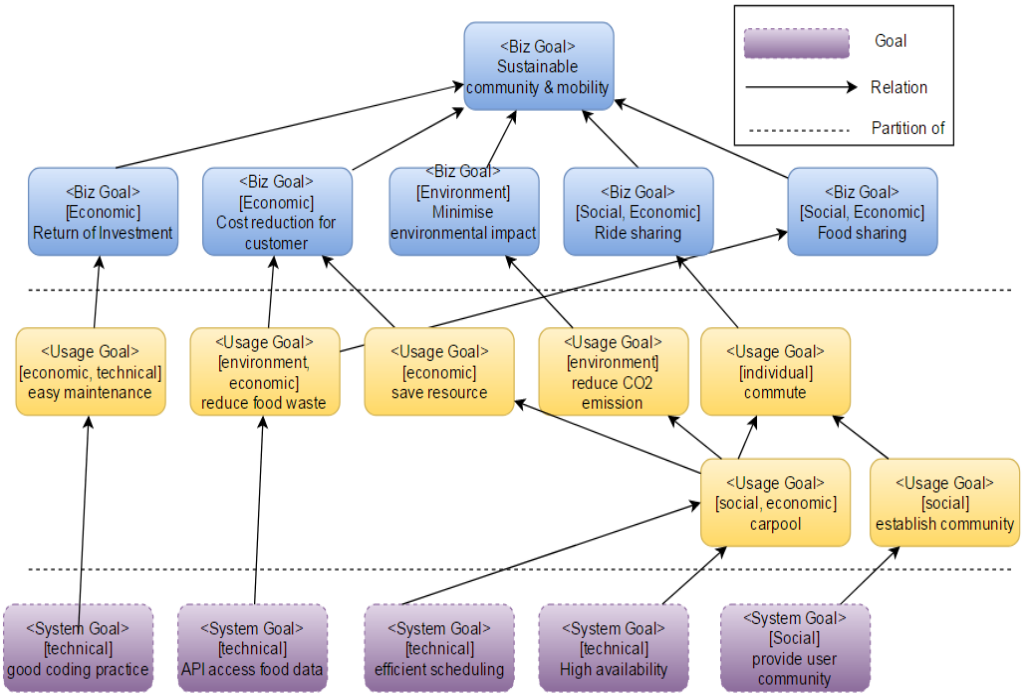


Figure 3. Goal Model (Sustainable Business Assessment)

4. The Biz Goal from Figure 3 represents the business goals that have direct impact on the system. The usage goals are those functional objectives of the system based on how it should behave. The system goals relates to the systems features. The color semantics in Figure 3 is only used to different each section. Based on the Goal Model (see Figure 3), a set of questions is created to characterize each goal. Table II details the questions associated with each goal.

TABLE III. SET QUESTIONS (GQM)

Goals	Questions
Reduce C02	Does the application reduce the amount of carbon emission in Lappeenranta?
Encourage car sharing	Is there an increase in car sharing among students?
Reduce food waste	What is the percentage of food waste after the application launch?
Promote sustainable community	Are students more aware of Sustainability?

5. System vision created to show the common understanding of all the stakeholders including users, management staffs, and developers. It is usually a pictorial overview of the system. It portrays how the system functions during operation.

6. Sustainability analysis shows the software system first, second and third order impact as shown in

Figure 4 with consideration for economic, sustainability dimensions. This analysis is based on the inputs from step 1 (sustainable business canvass) on contents of the environment, society, environment, social, individual and technical economy, process, value and people. It provides a holistic view of how different dimension of sustainability impact each other and their relation.

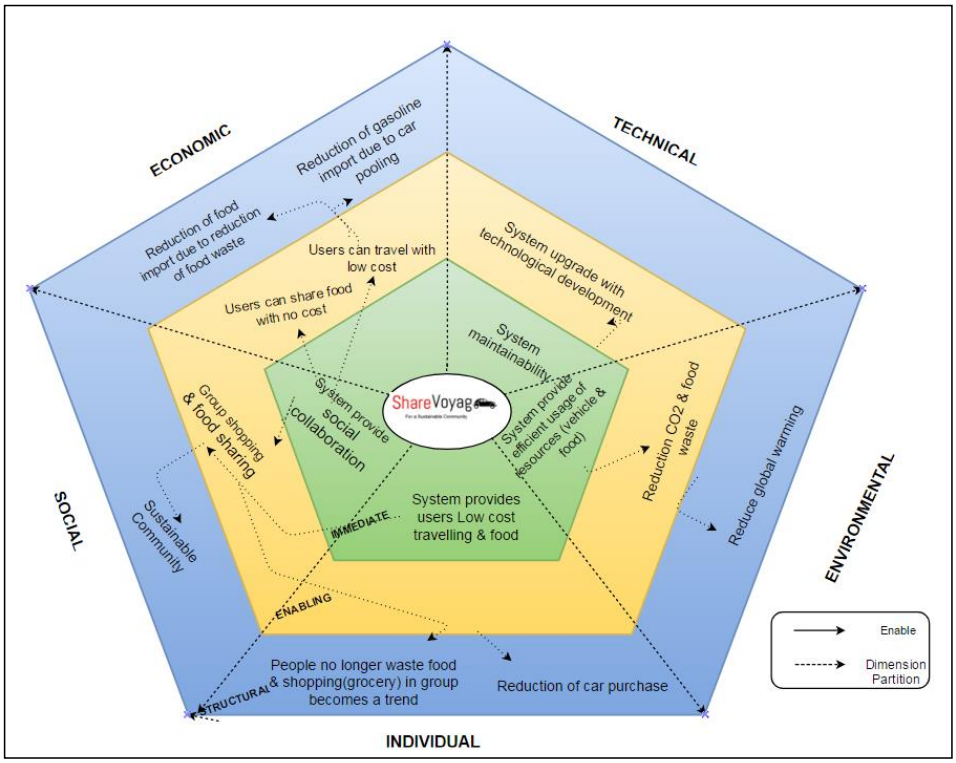


Figure 4. Sustainability Analysis (Sustainable Business Assessment)

7. Based on the system vision and sustainability analysis (see Figure 4) the software development team will be able to generate a metric worksheet (Table III) to evaluate the software system. To clarify, benchmark values are calculated based on the total software project modules and lines of codes.

TABLE IV. METRIC WORKSHEET (GQM)

Category	Question	Metric	Benchmark Value
Technical	What is the Backlog Management Index (BMI)?	BMI=Number of problems close/number of problems arrival *100	0 or 100
	What is the amount of rework?	Rework Metric (Total Number of function modified)	0

Category	Question	Metric	Benchmark Value
Economy	What is the BMI?	BMI=Number of problems close/number of problems arrival *100	0
	What is the software defect density?	Defect Density= Total defects/Size	≤ 10.46
	Does the actual project cost outweigh budgeted cost?	Net Cost	Positive Number
Environment	What is the BMI?	BMI=Number of problems close/number of problems arrival *100	0 or 100
	What is the defect density?	Defect Density= Total defects/Size	≤ 10.46
	How much energy does the software consume?	Energy efficiency = Useful work done/Used Energy	
	What is the percentage of car sharing?	Total amount of rides /100	
	What is the percentage of food shared?	Total amount of food share /100	
Social	Can users successfully complete task?	Gateway metric (1=Task success and 0= Task failure)	7
	What is the software defect density?	Defect Density= Total defects/Size	≤ 10.46
	Are the project teams happy?	Net working hours = Budgeted hours - Total working hours	Positive number
	Are the people more aware of sustainability?	Percentage of food shared	Positive number
Individual	Can users successfully complete task?	Gateway metric (1=Task success and 0= Task failure)	7
	What is the software defect density?	Defect Density= Total defects/Size	≤ 10.46

The result from Table IV provides a quantifiable result of the system measurement from the five sustainability dimensions. It allows for all-inclusive overview of the system with traces back the questions that are used to characterize each goals during the initial requirement stage.

The procedures and steps in S-BGQM encourage major stakeholders to consider sustainability during the software system development. It can be applied to software development life cycle using the enhancement model for sustainable software engineering proposed by Dick et al. [32]. This model covers sustainability review and preview, sustainability journal, process assessment and sustainability retrospect.

S-BGQM does not cover all aspects of sustainability. There is still need to improve the methodology used in deriving requirements goals from the business assessment component. The lack of intermediate stages to transform sustainability metrics has hinder the ability of S-BGQM to provide a better metric categorization.

VI. CONCLUSION

As highlighted in this paper, researchers have concentrated their efforts on the definitions and meanings of sustainability. Sometimes, definitions are somehow similar and often they are contradictory or conflicting. There is not yet a general consensus or a common ground on what sustainability and software sustainability means and how it can be quantified objectively. There is an urgent need for the entire software engineering community including practitioners and standardization bodies to have a standardized definition of sustainability, similar to other software quality factors. This will help to ground it in software measurement theories and practices.

We identified a set of sustainability requirements from the most cited definitions. This motivated our research on quantifying sustainability using those requirements. Quantification of sustainability means using variables that are measures of sustainability. We noticed that the biggest issue is that building a model or framework for sustainability quantification or/and defining its measurement scale and measures is already a difficult endeavor. The interpretation of such measures and their validation is a real challenge that requires a long-term research investigations and industry experiments.

Without a standard for software sustainability requirements, it becomes difficult to identify sustainability boundaries. A standard will lead to a unifying consensus that can foster sustainability quantification in software system.

S-BGQM is a modest contribution. We do not claim in this paper that S-BGQM is by itself a completed validated approach or framework. It is a kind of foundation that would be understood as "showing the map or road about what is need to be done to quantify and measure sustainability". It's not by itself the right and the unique road but it's just a possible one.

Our ambition was also to open the doors, or ground the efforts in a research agenda on how to measure sustainability. However we found that these concerns are necessary to overcome the obstacles on this long road for building a model for sustainability. The model should be based on a consensus and it can or should be part of ISO standards. There is a need for software engineering community to create cross-disciplinary research platform, for example building a kind of forum for discussing the definitions, perceptions and understanding of sustainability quantification. That forum can take the form of a new workshop or it can be part of an existing workshop of RE

like RE4SuSy or ICSE like the GREENS or it can be a joint book that bring people together to discuss it. This paper also calls for a forum that brings together all the different workshops like GREENS, RE4SuSy, GIBSE, and GinSENG to create a wider consensus.

Based on all these investigation, our intention is to bring this to the workshop discussion community with the hope that it can raise interest among researchers for further research on sustainability requirements, quantification and measurement. The following are some of the issues awaiting for further investigations:

- How to methodically specify sustainability requirements, meaning to quantify it?
- How can the sustainability requirements be measured? What are the measurement scales or measures for those requirements?
- How to categorize the current sustainability metrics and how they related to the five sustainability dimension?

Answers to these questions are a major milestone towards a model of sustainability as a quality attribute. One next stage in our research is a survey to explore sustainability perceptions and practices in industry.

Our future work includes carrying out large-scale industrial case studies to identify the practices of sustainability in software design and also to test the approach proposed in this paper. The goal is also to understand the ways to integrate and measure software sustainability. Another work is to study the process of issuing sustainability and green certification to companies. What are the activities that can be used to improve sustainability practices in the industry? One of such certification is the Albert Sustainable Production Certification [33] and Green Business certification. [34][35].

ACKNOWLEDGMENT

This work is fully supported and funded by DIGI-USER - Smart Services for Digitalization platform in Lappeenranta University of Technology (LUT).

REFERENCES

[1] B. Penzenstadler, “What does Sustainability mean in and for Software Engineering ?,” *1st Int. Conf. ICT Sustain.*, 2013.

[2] C. Becker *et al.*, “Sustainability Design and Software: The Karlskrona Manifesto,” *Proc. - Int. Conf. Softw. Eng.*, vol. 2, pp. 467–476, 2015.

[3] B. Christoph *et al.*, “Requirements: The key to sustainability,” *IEEE Softw.*, vol. 33, no. 1, pp. 56–65, 2016.

[4] F. Albertao, J. Xiao, C. Tian, Y. Lu, K. Q. Zhang, and C. Liu, “Measuring the Sustainability Performance of Software Projects,” *2010 IEEE 7th Int. Conf. E-bus. Eng.*, pp. 369–373, 2010.

[5] S. A. . Koçak, G. I. . Alptekin, and A. B. . Bener, “Evaluation of software product quality attributes and environmental attributes using ANP decision framework,” *CEUR Workshop Proc.*, vol. 1216, pp. 37–44, 2014.

[6] ISO, “ISO/IEC 25010,” *https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en*, 2001.

[7] M. R. Idio, “Measuring Sustainability Impact of Software,” vol. 16, no. 1, pp. 5–7, 2014.

[8] C. C. Venters *et al.*, “Software sustainability: The modern tower of babel,” *3rd Int. Work. Requir. Eng. Sustain. Syst. Work. Proc.*, vol. 1216, pp. 7–12, 2014.

[9] H. Koziolek, “Sustainability Evaluation of Software Architectures : A Systematic Review,” *Architecture*, pp. 3–12, 2011.

[10] J. A. Tainter, “Social complexity and sustainability,” *Ecol. Complex.*, vol. 3, no. 2, pp. 91–103, 2006.

[11] R. Seacord *et al.*, “Measuring Software Sustainability,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013.

[12] S. Naumann, M. Dick, E. Kern, and T. Johann, “The GREENSOFT Model: A reference model for green and sustainable software and its engineering,” *Sustain. Comput. Informatics Syst.*, vol. 1, no. 4, pp. 294–304, 2011.

[13] A. Raturi, B. Penzenstadler, B. Tomlinson, and D. Richardson, “Developing a sustainability non-functional requirements framework,” *Proc. 3rd Int. Work. Green Sustain. Softw. - GREENS 2014*, pp. 1–8, 2014.

[14] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, “Safety, security, now sustainability: The nonfunctional requirement for the 21st century,” *IEEE Softw.*, vol. 31, no. 3, pp. 40–47, 2014.

[15] M. Mahaux and C. Canon, “Integrating the Complexity of Sustainability in Requirements Engineering,” *First Int. Work. Requir. Eng. Sustain. Syst.*, 2012.

[16] K. Roher and D. Richardson, “Sustainability requirement patterns,” *2013 3rd Int. Work. Requir. Patterns, RePa 2013 - Proc.*, pp. 8–11, 2013.

[17] G. Lami, F. Fabbrini, and L. Buglione, “An ISO / IEC 33000-compliant Measurement Framework for Software Process Sustainability Assessment,” pp. 50–59, 2014.

[18] C. Calero, M. F. Bertoa, and M. Angeles Moraga, “A systematic literature review for software sustainability measures,” *Green Sustain. Softw. ({GREENS}), 2013 2nd Int. Work.*, pp. 46–53, 2013.

[19] T. Johann, M. Dick, S. Naumann, and E. Kern, “How to measure energy-efficiency of software: Metrics and measurement results,” *2012 1st Int. Work. Green Sustain. Software, GREENS 2012 - Proc.*, pp. 51–54, 2012.

[20] K. Erdélyi, “Special factors of development of green software supporting eco sustainability,” *SISY 2013 - IEEE 11th Int. Symp. Intell. Syst. Informatics, Proc.*, pp. 337–340, 2013.

[21] P. Bozzelli, Q. Gu, and P. Lago, “A systematic literature review on green software metrics,” *Sis.Uta.Fi*, 2013.

[22] S. Naumann, M. Dick, E. Kern, and T. Johann, “The GREENSOFT Model: A reference model for green and sustainable software and its engineering,” *Sustain. Comput. Informatics Syst.*, vol. 1, no. 4, pp. 294–304, 2011.

[23] B. Penzenstadler, “Infusing green: Requirements engineering for green in and through software systems,” *3rd Intl. Work. Requir. Eng. Sustain. Syst. 2014*, vol. 1216, no. 1, pp. 44–53, 2014.

[24] V. Basili, G. Caldiera, and H. D. Rombach, “Goal Question Metric Paradigm,” *Encyclopedia of Software Engineering*. pp. 528–534, 1994.

[25] “3 Minute Introduction to Strongly Sustainable Business Model Canvas, https://prezi.com/k5x2civcaw7y/3-minute-introduction-to-strongly-sustainable-business-model-canvas/ , Accessed on 8-06-2017,” no. November, p. 2014, 2014.

[26] “Strongly Sustainable Business Model Canvas Http://www.ssbmg.com/ Accessed on 8-06-2017.”

[27] B. Penzenstadler, “Goal Model Requirements Engineering for Sustainability,” *Lect. slides guest course Softw. Eng. Sustain. Lappeenranta Univ. Technol. 2016. Slides available online http//birgit.penzenstadler.de/teach/LUT.html.*

[28] B. Penzenstadler, “System Vision Requirements Engineering for Sustainability,” *Lect. slides guest course Softw. Eng. Sustain. Lappeenranta Univ. Technol. 2016. Slides available online http//birgit.penzenstadler.de/teach/LUT.html.*

[29] B. Penzenstadler, “Requirements Engineering for Sustainability -

Sustainability Analysis," *Lect. slides guest course Softw. Eng. Sustain. Lappeenranta Univ. Technol. 2016. Slides available online http//birgit.penzenstadler.de/teach/LUT.html*.

[30] Z. Durdik, B. Klatt, H. Koziolek, K. Krogmann, J. Stammel, and R. Weiss, "Sustainability guidelines for long-living software systems," *IEEE Int. Conf. Softw. Maintenance, ICSM*, pp. 517–526, 2012.

[31] B. Penzenstadler and H. Femmer, "A generic model for sustainability with process- and product-specific instances," *GIBSE 2013 - Proc. 2013 Work. Green Softw. Eng. Green by Softw. Eng.*, no. June 2015, pp. 3–7, 2013.

[32] M. Dick and S. Naumann, "Enhancing Software Engineering Processes towards Sustainable Software Product Design," vol. 2010, pp. 706–715, 2010.

[33] Bbc, "albert certification . Available at: http://www.bbc.co.uk/responsibility/environment/albert-plus."

[34] C. G. B. Network, "Green Business Certification, Available at: http://www.greenbusinessca.org/."

[35] G. B. Bureau, "Green Business Available at : https://greenbusinessbureau.com/how-gbb-certification-works/," no. June, pp. 1–6, 2013.

Publication II

S. Oyedeji, A. Seffah, and B. Penzenstadler.
A catalogue supporting software sustainability design

Reprinted with permission from
Sustainability
Vol. 10, pp. 1-30, 2018

sustainability

MDPI

Article

A Catalogue Supporting Software Sustainability Design

Shola Oyedeji [1,*], Ahmed Seffah [1] and Birgit Penzenstadler [2]

[1] LUT School of Engineering Science (LENS), Lappeenranta University of Technology, 53850 Lappeenranta, Finland; ahmed.seffah@lut.fi

[2] Department of Computer Engineering and Computer Science, California State University Long Beach, Long Beach, CA 90840, USA; birgit.penzenstadler@csulb.edu

* Correspondence: shola.oyedeji@lut.fi; Tel.: +358-4-1369-7708

Received: 30 May 2018; Accepted: 27 June 2018; Published: 3 July 2018

Abstract: Like other communities, sustainability in and for software design is a grand research and development challenge. Current research focuses on eliciting the meanings of sustainability and on building approaches for its engineering and integration into the mainstream software development lifecycle. However, few concrete guidelines that software designers can apply effectively are available. A guideline aims to streamline the design processes according to a set of well-known research routines or sound industry practices. Such guidelines can help software developers in the elicitation of sustainability requirements and testing software against these requirements. This paper introduces a sustainability design catalogue (SSDC) comprising a series of guidelines. It aims to assist software developers and managers in eliciting sustainability requirements, and then in measuring and testing software sustainability. The catalogue is based on reviews of the current and past research on sustainability in software engineering, which are the grounds for the development of the catalogue. Four different case studies were analyzed using the Karlskrona manifesto principles on sustainability design. A pilot framework is also proposed that includes a set of sustainability goals, concepts and methods. It exemplifies how to apply and quantify sustainability.

Keywords: sustainability; software sustainability; information and communication technology; software design; sustainability requirement; software sustainability analysis; software sustainability guidelines; Karlskrona manifesto

1. Introduction

Software sustainability and software engineering for sustainability are now recognized as timely important concerns not only for researchers, but also for the entire software industry and standardization bodies. A Microsoft report as well as an IBM global chief executive officer (CEO) study on sustainability showed an increasing growth in the percentage of organizations redesigning their entire business models to incorporate sustainability [1–3]. The Sustainability and Innovation Global Executive Study indicates that 48% of respondents out of the 4000 executives and managers interviewed worldwide agree that sustainability urged them to modify their business models [4,5]. Sustainable development is also driving software innovations for creating new opportunities to cut costs, adding value and gaining competitive advantage [6]. As software is the catalyst for economic and social changes today [7] and the pillar for all industries, there is a huge pressure from regulators and civil society to develop more green software that uses less energy [8].

As a matter of fact, the Ericsson sustainability report shows that information and communications technology (ICT) could help reduce global greenhouse gas (GHG) emissions by up to 15%. It forecasts that by 2021, 28 billion devices will be connected to each other [9] which will increase energy

consumption drastically. Another energy and carbon report from Ericsson forecasts 90% of the world's population to have mobile coverage, and 60% will have the ability to access high-speed long-term evolution (LTE) data networks [10]. Such reports are clear indicators of the huge sustainability impact of ICT. Overall, the ICT sector contributes around 2% of the global CO_2 emissions. It is also accountable for approximately 8% of the European Union's (EU) electricity consumption and 2% of the carbon emissions from ICT devices and services [6].

It is therefore important to look how to reduce the impact of ICT on the environment and how sustainability can be incorporated better into the software development lifecycle. However, current software development practices do not provide sufficient support to all sustainability concerns. This should not be limited to energy consumption, but it should include also all the other aspects of sustainability.

For example, [6] provides a more broader perception of sustainability in software engineering. Sustainability may or should refer also to, for example, electronic waste management and the ecological impacts of recycling the drastically increasing amount of computing gear. Moreover, it is worth mentioning that the impact of the cryptocurrency market on energy consumption is also a very serious problem from a sustainability perspective [11]. The energy consumption of blockchain technologies has raised environmental concerns. The resulting energy consumption per year is estimated at 12.76 TWh [12]. The digital infrastructure supporting the wide diversity of interactive devices and services available on the cloud accounts for up to 85% of total environmental impact [13].

Therefore, the meanings and integration of sustainability should cover the five dimensions of software sustainability [14–16]: economic, social, individual, environmental and technical [17]. The economic, technical and business dimensions are now core aspects of fundamental values in companies embracing sustainable development [18]. An area that has received less attention is the social dimension. It entails the well-being of the software users community and developers [14], and is about changing the human mindset and designing their perceptions and experiences of sustainability.

Practices and processes that are widely used in an industry setting such as agile methodologies and model-driven approaches lack aspects addressing sustainability challenges [19]. Practitioners are not prepared for integrating sustainability efficiently and effectively. Where should sustainability ingredients be considered? Indeed, the different sustainability dimensions have no reference framework that can assist software developers. Researchers also highlighted the vital need to define measures of sustainability and search for avenues for their integration in the wider engineering processes [6,20].

Our research focuses mainly on the integration of sustainability during the software design stage and into the design practices. Design is a key milestone where supporters and pioneers largely recognize the importance of sustainability. Varying perspectives have been discussed such as the design of sustainability and sustainability by design [21]. Sustainability by design is one way to achieve sustainability and for integrating sustainability perceptions in software engineering [22].

The research discussed in this paper is twofold. First, based on the analysis of the literature and built on the Sustainability Design Manifesto principles, the paper introduces an original sustainability design catalogue (SSDC). Then, it describes a pilot framework that exemplifies how the SSDC can be applied and how the underlying sustainability design principles can be incorporated into the design practices for all software development life-cycle (SDLC) phases. The SSDC is a set of practical concrete guidelines and indicators supporting sustainability by design practices. The Karlskrona manifesto principles are viewed in this research as a set of high-level abstract principles and perceptions for sustainability design in and for software systems [23,24].

Overall our research addresses the following specific questions and provides the following contributions:

1. How do the principles detailed in the Karlskrona manifesto relate to the software development life-cycle phases (SDLC) in general and the design stage especially? SSDC suggests concrete guidelines to apply the high levels and abstracts principles of sustainability.

2. How do the SSDC and the underlying perceptions of sustainability relate to the first-, second- and third-order impacts of software sustainability as well as the five dimensions of sustainability? SSDC tried to bridge the current gaps between the principles and the indicators of sustainability.
3. How can these principles be applied while ensuring that sustainability is achieved during design? The SSDC can be viewed as a tool supporting the sustainability by design approach.
4. How should these principles be applied for the wide diversity of software systems that exists today and those in the future that should consider sustainability as a quality in the same way we engineer the other quality attributes today of such security and usability? A pilot framework is described portraying the applicability of the SSDC for diverse software systems.

The remainder of this paper is as follows. Section 2 covers the background and related research work on sustainability in software design. Section 3 presents the foundation of the Karlskrona manifesto principles and how these relate to the software development life-cycle phases. Section 4 details the structure and components of the Software Sustainability Design Catalogue (SSDC). Section 5 discusses how the SSDC was derived and can be applied. Section 6 provided a practical example for the usage of the SSDC and the pilot framework. The benefits of the SSDC and pilot framework are summarized in Section 7. The conclusion summarizes with comments and future research work.

2. Sustainability in Software Design: Background and Related Research

Sustainability is one of the grand challenges of our civilization because of its pervasiveness. The way we design, and consequently use, software-intensive systems has a significant impact and can influence human perceptions of sustainability greatly [25]. Although design is a central phase of any software development process [22], there has been limited research work on software sustainability design. The most relevant related works are listed and described in the following.

Currently there is no single point of reference for researchers or practitioners where the sustainability measures are gathered and exemplified [26]. The perception of professionals about sustainability affects the way sustainability has been applied in software development [27]. However, the pathway to a sustainable society is unclear since sustainability means different things to different people [28]. People's different lifestyles, values and practices also affect how sustainability is treated [29]. Furthermore, one of the major problems for software designers is that even with a systems approach, there are few existing tools that wrap core principles of sustainability together. Instead, designers must learn to patch together a series of disparate sustainability understandings, and frameworks in order to address the different dimensions of sustainability [30]. An alternative design solution is based on the sustainable design practices that use the least energy over ICT's life cycle [31]. The global Sustainable Development Goals formally adopted by the United Nations (UN) in 2015 can serve as an inspiration. They have the potential to guide software practitioners, especially human–computer interaction (HCI) specialists [32]. In the area of cloud computing, there is not enough awareness about the value benefits of sustainability especially when selecting and deploying cloud computing software among organizations [33].

Software design as a key factor can help reduce energy consumption by 30% to 90% because software provides the real energy saving that tells hardware what to do and how to function [34]. A catalogue of sustainability guidelines has been proposed in [35]. It incorporates all phases of the system development life cycle while providing specific support to project managers, software architects, and developers during the entire system design, development, operation, and maintenance. However, it is not as detailed as the SSDC proposed in this article as it does not cover all the different sustainability dimensions (economic, environmental, social, individual, and technical), nor the first, second and third order of impacts of software systems and metrics/indicators to evaluate the effectiveness of guidelines in the catalogue.

The International Organization for Standardization (ISO) and Institute of Electrical and Electronics Engineers (IEEE) series of software engineering standards provide little guidance on sustainability. While there has been some increase in literature about the environmental and social dimension of

sustainability for software systems, there is less attention on sustainability in software development and use [36]. The effects of software systems are getting less attention and work to formalize sustainability as part of software engineering process is still not considered in the official standards and models of software systems [37]. The concept of sustainability for software design, and its integration into the existing catalogue of design quality attributes is needed to achieve sustainable software [38] and sustainability should also be considered as a quality of software systems like security and usability [39]. Sustainability in software design requires a multidimensional and interdisciplinary approach [40–43]. Kern et al. [44] developed a model for green software based on sustainability criteria, although there are still several software design quality models that include attributes like flexibility and reusability, with no attribute that captures how cost-efficient are the set of design decisions over time.

A life-cycle model that uses a cradle-to-cradle approach to analyse impacts of each software product life-cycle phase can help to develop green and sustainable software products [45]. Another proposed generic model for sustainability was proposed with instances for companies and projects based on different cases studies. The proposed process helps requirements engineers to properly analyse projects during software design based on different sustainability dimensions [17]. An experience report from a case of applying standard requirements engineering methods to analyse sustainability aspects shows how requirements can impact software design [46]. Another generic model for improving the general software development process for sustainable software product design is the process enhancement model, which includes activities and artefacts such as sustainability reviews and previews, ongoing process assessments, a sustainability retrospective, and a sustainability journal. Although the model does not currently cover sustainability benchmarks, it provides a sound basis for future integration [47].

A description of how to support different aspects of sustainability in software development processes, software system analysis for production, and usage phases of the life cycle can also provide an understanding of what sustainability means in software engineering [19]. In the same vein, a study of the life-cycle activities of software development with a focus on environmental protection provided a guide through a formula to calculate software waste to encourage the development of green software [48]. The author highlights key activities during software design and development with key factors at each stage of software design and how each of these factors relates to a green aspect in software development. Also highlighted is the fact that thoroughly designed and implemented software uses energy efficiently through computational and data efficiency [48].

Researchers from different disciplines tried to tackle the issue of sustainability through collaborative work via organizing interdisciplinary conferences and workshops [24]. One common focus is sustainability in requirements. Sustainability requirements were treated as first-class quality requirements, and as such systematically elicited, analysed and documented with the goal of showing that small and easy steps during requirement can lead to the design and development of more sustainable systems [46]. This is corroborated by another research work [49] stating the need to characterize software sustainability as a quality factor in requirements elicitation. In addition, a sustainability requirements checklist and guide approach demonstrate how to include the objective of environmental sustainability from the very early steps of software development [50]. It also shows how green requirements engineering may be conducted within the scope of general purpose requirements engineering and accommodate the new objective of improving environmental sustainability [50].

The use of sustainability requirement patterns (SRPs) is another approach that provides software engineers with guidance on how to write specific types of sustainability requirements. The aim is to overcome the barriers of incorporating environmental sustainability into the requirements engineering process [51]. Sustainability requirements can also be a non-functional requirement (NFR) using an NFR framework informed by sustainability models and how it can be used to correctly obtain and describe sustainability related requirements of the software system to be developed.

The impact of ICT on the world's CO_2 emissions can be reduced through improved software-energy efficiency on multi-core systems [52] although there are few studies and suggestions about 'what' aspects of sustainability to measure and 'how' to do it with regards to ICT [53].

Sustainability should be considered in software quality models, although there has been less research channelled towards it [54]. Planning and management of software sustainability as a quality attribute is impaired by a lack of consistently applied, practical measures [55]. Without these measures, it is impossible to determine the effect of efforts to improve sustainment practices.

The following are the main conclusions from the background and related work:

1. There is no single reference point where measures of software sustainability are gathered and exemplified.
2. Design is key to achieve software sustainability, thus the need to show how software designers can incorporate sustainability during software design to improve ICT energy usage and CO_2 emission.
3. The need for a framework or model to assist and guide developers during software design to incorporate sustainability requirements.

These conclusions are the reasons for initiating the creation of a SSDC that can be used by researchers and developers to create new frameworks, tools, guidelines and practices for software design and development. An example of such framework is the proposed pilot framework in section five of this article as guide for both experienced and infant software designers during software design and development.

One last clarification that needs to be made here is the fact that the concepts of sustainable and green are often used interchangeably, in many communities including software engineering. This article considers that "green software" and "sustainable software" is not the same. Green is usually defined as "products, systems and services that have limited negative impact on human health and environment".

As defined in the article at hand, sustainability includes green and it goes beyond green. It is represented by five pillars for environmental, social, economic, human and technical sustainability.

3. The Foundations of Sustainability by Design: The Karlskrona Manifesto

The Karlskrona Manifesto for Sustainability Design (KMSD) has its roots in the Third International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy) [56], held at RE'14 in Karlskrona, Sweden. Christoph Becker's paper [57] about the relationship between the concerns of sustainability and longevity provided one of the motives for the creation of the manifesto.

The key goal was to blend the diverse aspects of sustainability to clarify its scope, objectives and challenges of the perceptions of sustainability leading to an interdisciplinary platform for researching sustainability [57]. The manifesto brings together input from researchers of various disciplines in the field of software engineering with sustainability research interests as the creators of the design manifesto [23,58].

The Karlskrona Manifesto for Sustainability Design includes nine principles of sustainability design [23]. Those principles provide the basis for creating a reference point that can be applied during software design by different stakeholders (Table 1). The manifesto is accessible via the web [58], where those interested in supporting the manifesto can sign it.

Table 1. Description of the Karlskrona manifesto principles, adapted from [23].

Principle Number	Principle	Description
P1	Sustainability is systemic	Sustainability is never an isolated property. It requires transdisciplinary common ground of sustainability as well as a global picture of sustainability within other properties.
P2	Sustainability has multiple dimensions.	We have to include those dimensions into our analysis if we are to understand the nature of sustainability in any given situation.
P3	Sustainability transcends multiple disciplines.	Working in sustainability means working with people from across many disciplines, addressing the challenges from multiple perspectives.
P4	Sustainability is a concern independent of the purpose of the system.	Sustainability has to be considered even if the primary focus of the system under design is not sustainability.
P5	Sustainability applies to both a system and its wider contexts.	There are at least two spheres to consider in system design: the sustainability of the system itself and how it affects the sustainability of the wider system of which it will be part.
P6	System visibility is a necessary precondition and enabler for sustainability design.	Strive to make the status of the system and its context visible at different levels of abstraction and perspectives to enable participation and informed responsible choice.
P7	Sustainability requires action on multiple levels.	Seek interventions that have the most leverage on a system and consider the opportunity costs: whenever you are taking action towards sustainability, consider whether this is the most effective way of intervening in comparison to alternative actions (leverage points).
P8	Sustainability requires meeting the needs of future generations without compromising the prosperity of the current generation	Innovation in sustainability can play out as decoupling present and future needs. By moving away from the language of conflict and the trade-off mindset, we can identify and enact choices that benefit both present and future.
P9	Sustainability requires long-term thinking.	Multiple timescales, including longer-term indicators in assessment and decisions, should be considered.

The Karlskrona manifesto principles aim to be a practical guide to the entire community like the Agile manifesto [59], the Business Rules manifesto [60], the Service-oriented architecture (SOA) manifesto [61,62], and the Recomputation manifesto [63]. It supports stakeholders in industry and academia (companies, standardization organization, software practitioners, researchers and students) for promoting and developing sustainability design and practices in software development [23,57]. The Karlskrona manifesto also serves as a facilitator for thinking about the broad effects of software on society and the need to embody longer-term thinking, ethical responsibility, and an understanding of how to integrate sustainability into the design of software systems [24].

Table 2 shows how these Karlskrona principles can be related to software development phases [64]. Relating these principles to the software development phases will provide an avenue for using these principles especially for different software systems.

Table 2. Karlskrona manifesto principles in relation to software development life-cycle (SDLC) phases.

SDLC Phases	Karlskrona Manifesto Principles
Phase 1. Project Definition	**P1-** This ensures that the project initiation considers sustainability in the overall project definition from the beginning. **P2-** Software sustainability has different dimensions that have to be considered from the beginning for better project management with different stakeholders. **P3-** Software project usually involves stakeholders from different domains, incorporating their sustainability concerns provides better management of those concerns from multiple perspectives which can help the incorporation of sustainability for the software.
Phase 2. User Requirements Definition	**P2-** It is important to take note of user requirements in relation to each of the sustainability dimensions in order to have better sustainability analysis during the analysis and design phase
Phase 3. System Requirements Definition	**P4-** During elicitation of system requirements, requirement engineers should consider sustainability concerns for the system during the requirements definition even when it is not a core part of the user requirements. **P5-** Cross evaluate the consequential impacts of the system sustainability requirements and the environment in which the system will function.
Phase 4. Analysis and Design	**P2-** Applying this principle provides a blueprint for system evaluation from all sustainability dimensions (economic, environment, social, individual and technical). **P4-** At this phase, this principle helps to encourage analysis of system design based on sustainability in order to facilitate better sustainable system. **P6-** Application of this principle enables better visual and visible overview of the system from different levels of abstraction. **P8-** This will provide better understanding during analysis to make better choices that will help the potential users of the system in present and in future when the system evolves.
Phase 5. Development	**P2-** This will encourage developers during this phase to consider different sustainability dimensions, especially technical, social and individual dimensions. **P4-** Encourage the search for better avenues to make the system sustainable from the development perspective (developers) and also the functions of the system to aid longevity.
Phase 6. Integration and Testing	**P2-** Provides integration and for test team to have a sustainability template that can be used to test the system for all sustainability dimensions based on the sustainability requirement output from phases 2, 3 and 4. **P4-** Application of this principle will aid consideration of sustainability in this phase even if the primary focus of system is not about sustainability.
Phase 7. Implementation	**P5-** Provides beforehand reasoning for the development team to consider the sustainability of the system, its production environment and when pushing it live for use. **P7-** Based on principle 5 (P5), this principle will aid consideration of seeking the involvement of different stakeholders to make the actualization of the system sustainability possible in the production environment and when pushed live.
Phase 8. Sustainment/Maintenance	**P9-** This principle at this stage help to create the conscious awareness so that when the system is in a live environment, there will be continuous evaluation to assess the system sustainability and think of ways for optimizing and improving the sustainability of the system from the different dimensions.

Table 2 highlights some avenues for putting the Karlskrona manifesto principles into practices. Relating these principles to the software development phases will provide an avenue for better understanding of how these principles relates to software development. However, the Karlskrona manifesto focused on high-level principles, not techniques [24], which means there is a need to exemplify the principles to show their practical usage with techniques. The following are the limitations of the manifesto that motivate the development of the SSDC:

1. The principles are abstract and generic to serve all the possible stakeholders interested in sustainability in all the stages of the software development and management phases.

2. The principles are at a high level of abstraction, missing many details for their practical usage.
3. The principles are closely related, making a trade-off among them difficult, especially for a novice in the field of sustainability.
4. The principles are not connected to tangible measures but serve as a guide to create measures.

4. Structure of the Proposed Software Sustainability Design Catalogue (SSDC)

The SSDC serves as a tool that can facilitate the integration of sustainability into design practices as well as lead to a better understanding of sustainability by practitioners. The Software Sustainability Design Catalogue (SSDC) is a set of criteria derived from the nine Karlskrona manifesto principles based on cross analysis of different systems. For each criteria, indicators of sustainability are also derived. The structure of the software sustainability design catalogue is detailed in Figure 1.

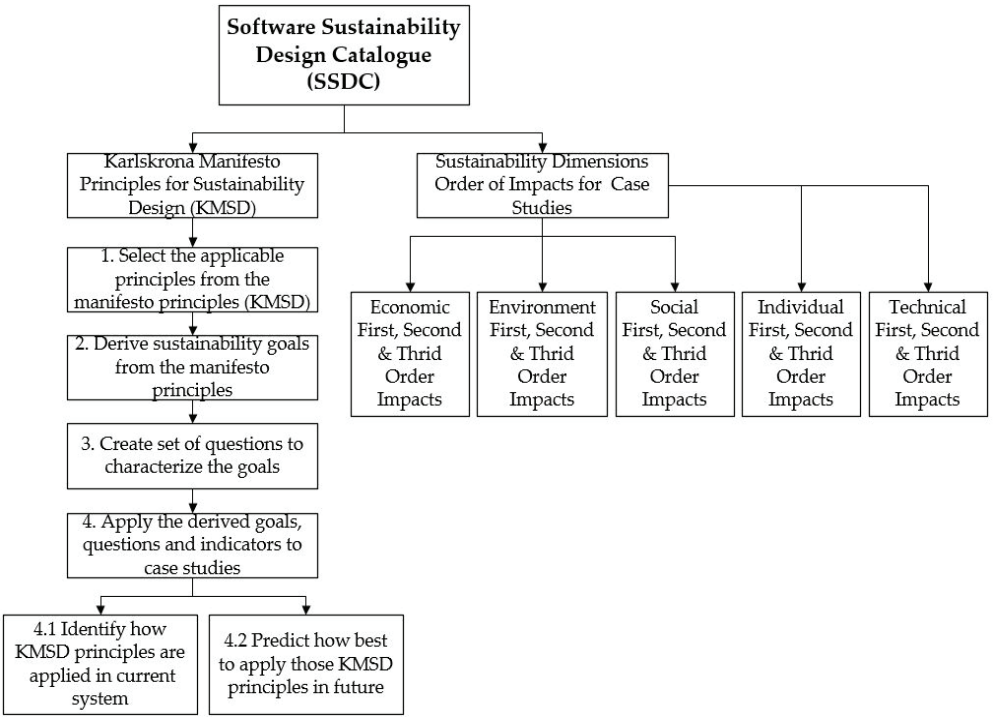


Figure 1. Flow of the derivation of Software Sustainability Design Catalogue (SSDC).

The SSDC distinguishes mainly two components. The first is the sustainability analysis of systems, which is mainly a set of criteria for evaluating the sustainability of software systems. Each criterion is characterized by the following core elements used for evaluating software systems:

1. One or more Karlskrona principle (the 9 principles in Table 1) is used in the evaluation of each system category. Not all principles necessarily can be applied to all systems. The principles are identified using a tag of **P1** to **P9** (Principle 1 to 9).
2. Goal/requirement: this highlights the desired end result for each system category based on sustainability consideration.
3. Stakeholders: those responsible for implementing the goals/requirement.
4. Questions characterizing each goal. From each goal, a set of questions are derived that will determine if each goal is being met.

5. Indicators are used to answer the questions as a way to evaluate if the goals were achieved.
6. Current principle usage in software: this covers the current application of the principle in existing system design and development, even if it is not explicitly stated in current system documentation.
7. Future principle usage in software: based on the evaluation of the current principle application in existing system design and development, a potential usage of the principle in future system enhancement and design is suggested.

The second component are the indicators of sustainability associated with each criteria. These indicators are related to the sustainability dimensions and their order of impacts.

The orders of impact [65,66], cover all the positive and negative effects of software on the environment which are decomposed into three orders of magnitude. The first order impacts (Immediate effects) are about the direct effects of the development and use of software system. The second order impacts (enabling effects) are about the indirect impacts related to the effects of using the software system in its application domain. The third order impacts (structural effects) are the cumulative long-term effects resulting from accumulating first and second order impacts over time.

The sustainability dimensions include [24]:

- The **individual dimension** covers individual freedom and agency (the ability to act in an environment), human dignity, and fulfilment. It includes individuals' ability to thrive, exercise their rights, and develop freely.
- The **social dimension** covers relationships between individuals and groups. For example, it covers the structures of mutual trust and communication in a social system and the balance between conflicting interests.
- The **economic dimension** covers financial aspects and business value. It includes capital growth and liquidity, investment questions, and financial operations.
- The **technical dimension** covers the ability to maintain and evolve artificial systems (such as software) over time. It refers to maintenance and evolution, resilience, and the ease of system transitions.
- The **environmental dimension** covers the use and stewardship of natural resources. It includes questions ranging from immediate waste production and energy consumption to the balance of local ecosystems and climate change concerns.

5. How the SSDC Have Been Derived and Can Be Used

The SSDC was developed using four case studies (see Table 3). Data were gathered for analysing the four different case studies using also the Karlskrona manifesto principles and the orders of impacts. The second and third authors then cross-validated the data collected. Based on the aggregated data, a first draft of catalogue was developed. Then, the proposed software sustainability design catalogue was refined using other types of systems.

Table 3. System categories and types used in case studies.

System Category	System Type
Cyber physical system	Smart home system
Embedded system	Washing machine
Gaming	Angry bird
Desktop application	Microsoft office

The types of systems used in the case studies are summarized in eight tables, Tables 4 and 5 (Cyber physical systems—Smart Home) and Tables A1–A6 detailed in the Appendix cover the following types of systems:

1. Embedded systems that are composed of electrical and mechanical components completely encapsulated by the device they control. The sample case study used in this category is a "Washing Machine" [67–72] (see Tables A1 and A2 in Appendix).
2. Mobile games as an application design that runs on mobile devices. The game case study used in this category is "Angry Bird Game" [73–77] (see Tables A3 and A4 in Appendix).
3. Desktop applications that run on standalone computers. The sample application used in this category is "Microsoft Office" [78–81] (see Tables A5 and A6 in Appendix).

To illustrate how the SSDC works and what the guidelines look like, Tables 4 and 5 present the guidelines for cyber physical system (CPS). CPS are defined here as the integrations of computation, networking, and physical processes that are tightly connected with its users. The sample system used in the catalogue is "Smart Home". The presented guidelines can assist companies and software developers identify key areas that relate to sustainability and recognize strategic avenues on how current and future smart home solutions should be designed in a more sustainable manner. This enables them to make good sustainability decisions during and after the design of smart home solutions.

Table 4 highlights one important issue that standardization authorities in this domain can work on, which is the cross-platform compatibility for smart home devices. Smart home appliances should be compatible with other devices from different manufacturers based on standards to avoid increase in energy usage. Smart home solutions should provide meaningful graphical information that can educate users, thereby encouraging users to behave more sustainably.

Table 5 of the SSDC for cyber physical system (smart home) provides different insights on the direct, indirect and structural impact of home automation design and deployment from the different dimension of sustainability. From Table 3, companies and stakeholders will be able to incorporate the following sustainability goals for the design and development of home automation solutions:

1. **Environment**: reduce household energy consumption.
2. **Economic**: reduce household cost on energy.
3. **Individual**: provide user friendly solution for home users with easy to use user interface and information to induce sustainable behaviour among users.
4. **Technical**: provide good security for user personal data and avoid technical glitch that could lead high energy usage.
5. **Social**: encourage users to form communities to share data as a way of encouraging each other to be energy conscious and environmentally aware of the consequences of their actions and inaction while using smart home solutions.

The application of these principles from the catalogue offers explicit goals and opportunities for sustainability integration in system design through multiple perspectives for systems with sustainability as their core goal and those system without sustainability as their main goal. The below are detailed descriptions of the principles used in providing information on how best to engineer and think of sustainability for smart home solution (see Tables 4 and 5) from the catalogue.

Table 4. Sustainability analysis of cyber physical system (smart home) based on Karlskrona principles [82–85].

Karlskrona Principle and Goal	Principle Usage	Stakeholders	Question	Indicator
(P2) Cross platform compatibility	**Current**: Smart home appliances are compatible with only few other manufacturers' devices in the market. **Future**: Smart home appliances should be compatible with other devices from different manufacturers based on standard interface to avoid increases in energy usage. This can be achieved by enforcing device standards that can be used for cross-platform compatibility. Home automation appliances should be economic and at same time environmental friendly.	Business analyst	Can device function with other device from different manufacturer?	Device cross-platform compatibility
(P4) Educate Users	**Current:** Smart home solutions provides graphical information about energy usage but not necessarily educate users on how to be energy conscious based on their daily habit over a period of time. **Future**: User data from smart home solution should be used for educating users thereby encouraging users to behave more sustainably by presenting energy usage in an informative and educative manner (for example, relate energy usage to the amount of killed trees). User data could also be used for prediction aimed at optimizing the use resources such as water and energy within a household or company. The solutions could help interconnect other systems that can help save resources like water and electricity in a household or company.	Software developers Business analyst	Are users aware of their actions relating to electrical appliances in the house or company?	Usage data over time to detect changes in user habits
(P9) Reduce production and solution cost	**Current**: There are currently few cost-effective solutions that will encourage users to adopt home automation solutions in the long term. **Future:** Use cheap, environmentally friendly resources in the production of home automation device (hardware) that can reduce production cost. If the cost of production reduces, the overall cost of smart home solution will also reduce which will increase its affordability among users. There can also be low-cost solutions for poor countries to assist in the use of water and energy judiciously (reduce waste).	Business Analyst	Did we manage to reduce costs compared to previous years (before solution was a smart home)?	Net cost of smart home solution

Table 5. Sustainability dimensions order of impacts for cyber physical system (smart home) [82,86–89].

Order of Impacts	Environment	Economic	Technical	Social	Individual
1st	Increase in the use of natural resources in the production of hardware for smart home devices and pollution of the environment from toxic material used in production.	Creates new business opportunities for those in this sector (setup and installation of devices at home).	Pave way for improving existing technologies and development of new tools to meet new market demands for sustainable usage of these technologies.	Breeds new communities of users and suppliers.	Users rely on devices to control some aspects of their lives at home and in offices.
2nd	Reduce household energy consumption.	Reduce household bill for energy consumption.	High demand for security of user personal data (privacy).	Increase comfort, safety, flexibility, and security of user.	Demand for sustainable user-friendly solution for home users.
3rd	Increase in the use of toxic material for production of hardware. Less energy consumption over a long period of time.	Decrease cost of energy through optimized solution over time.	Efficient provision of sound technical solutions to avoid technical glitch that could lead to high energy usage. Encourage innovation on how to create cost-effective technologies and devices to reduce household and company energy usage	Encourage users to form communities to share data as a way of encouraging each other to be energy conscious	Induces sustainable behaviour among users.

Sustainability has multiple dimensions (P2): the application of this principle provides an overview of the fundamental issues and positive opportunities that could encourage stakeholders in the smart home domain to cross reference in the design and development of smart home solutions, especially during solution requirements from users and choosing appropriate boundaries.

Smart home design and deployment in this domain requires getting inputs for the effect of design solutions on the environment from natural resources used in building hardware devices, energy consumption of the devices, social behaviour and interaction between people in a family (household), company and other places where these solutions will be deployed. This means all sustainability dimensions (environment, economic, social, individual, technical) will be analysed for better design output.

Sustainability is a concern independent of the purpose of the system (P4): the goal of most smart home solutions is to provide comfort and reduce energy consumption for its users, but it is important to consider an encompassing view of sustainability. This is to be able to get even more benefits such as reducing pollution through the use of environmental friendly materials in producing hardware devices used for smart home solutions. The smart home solution can be used to educate and inform users about the negative consequential effect of their behaviour and habits. This can help induce sustainable behaviour among users. For a smart home solution design to be effective and meet user needs, it will require the expertise of a psychologist or at least an adequately educated interaction designer to help provide information according to the level of comfort and technical expertise of those in manufacturing, transportation, electrical, business and ICT discipline.

Sustainability requires long-term thinking (P9): it is important to think of how the smart home solution provided today will evolve to meet the requirements of current users and be adaptive enough to satisfy future user needs. This will require looking at measures to capture user behaviours over time through computational intelligence to predict future actions of users through data generated from time to time.

Based on the SSDC, a pilot framework to guide stakeholders involved in the design and development of a software system is proposed. Figure 2 provides a detailed flow of the pilot framework.

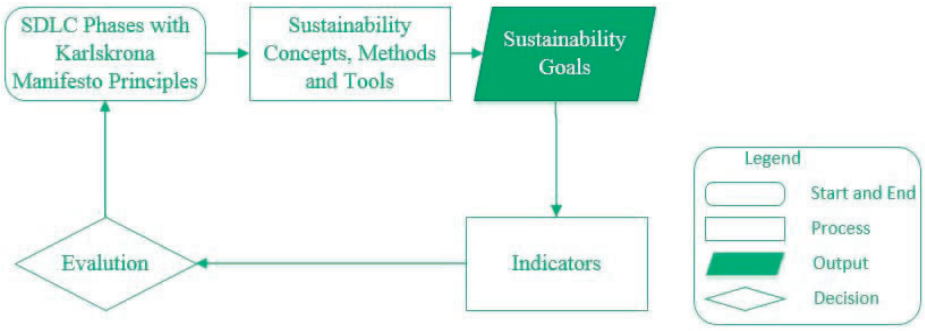


Figure 2. Pilot framework for sustainability of software system design based on SSDC.

The pilot framework is the first derivative from the SSDC to assist developers incorporate sustainability during system design and development covering the software development life-cycle (SDLC) phases. For a better understanding, the pilot framework is presented below in a tabular form to show contents that are involved in the framework. Table 6 contains all contents of the framework. It is important to highlight that the indicators used in the framework (Table 6) are influenced by the nine Karlskrona manifesto principles mapped to each of the software development life-cycle phases (see Table 2) and the work of Kem-Laurin Kramer [90].

Table 6. Contents of pilot framework for sustainability of software system design.

SDLC Phases and Karlskrona Manifesto Principles	Sustainability Goals	Sustainability Concepts, Methods and Tools	Indicators
Phase 1. Project Definition, P1, P2 and P3	Transmaterialization, design for sustainable efficiency, reusability.	Cradle to cradle, biomimicry, sustainable business canvas.	Carbon footprint, material footprint, end of life footprint.
Phase 2. User Requirements Definition, P2	Increase sustainability awareness among users.	Helix of sustainability.	Total number of sustainability requirements, priority assign to sustainability requirements.
Phase 3. System Requirements Definition, P4, and P5	Design for efficiency, sustainability awareness and interoperability.	Biomimicry, cradle to cradle, goal model.	Total number of system goals relating to sustainability dimensions.
Phase 4. Analysis and Design, P2, P4, P6 and P8	Design for reuse and efficiency, localization, interoperability	Biomimicry, helix of sustainability, Life-cycle sustainability assessment, social return on investment, sustainability analysis radar chart.	Number of first-, second- and third-order impacts of system identified.
Phase 5. Development, P2 and P4	Design for reuse, design for module replicability, design for efficiency, design for sustainability awareness, design for efficiency, design for easy service and maintenance.	Biomimicry, cradle to cradle.	Number of coding choices influenced by sustainability, number of features (functions) added to systems to inform users about sustainability through functions like eco feedback.
Phase 6. Integration and Testing, P2 and P4	Design for easy assembly and disassembly, design for durability,	Cradle to cradle, sustainability analysis radar chart, life-cycle sustainability assessment.	How much information from sustainability analysis chart was used during integration and testing such as the number of systems functions tested against sustainability concerns such as the first-order (immediate) impact and possible second-order (enabling) impacts of the system.
Phase 7. Implementation, P5 and P7	Design for easy use, design to induce conscious sustainability awareness, design to educate users about sustainability, design for easy recycle.	Biomimicry, cradle to cradle.	The priority assign to sustainability by developers and the system owners/users during after implementation
Phase 8. Sustainment/Maintenance, P9	Proper design for serviceability, design for easy replacement of code modules, design for continuous user engagement through sustainability awareness.	Life-cycle sustainability assessment, sustainability analysis radar chart, cradle to cradle.	Number of improvements to system based on sustainability requirements either from users' feedback or developers.

6. Application of the SSDC and the Pilot Framework

In order to exemplify the application of SSDC and the pilot framework, an excerpt from a cyber physical system (smart home) is used here. We consider the following scenario:

> "*A software engineer called Henry has the task of eliciting and documenting the requirements for a new smart home system. Being aware of his responsibility for the software system sustainability he creates, and its impacts, he takes the template of the sustainability analysis of the five dimensions and the three orders of effects from the design catalogue with him to the customer during their first meeting. The customer is curious about these additional analysis ideas, and Henry explains to his client what they mean and gives his client a couple of examples. Then, together with the customer, he fills out the template applying the concepts from the design catalogue (SSDC) to find out what those dimensions and orders of impact mean for the smart home system the customer wants for his house. The information from the activity goes into the requirements analysis that is subsequently conducted and used as a measurement yardstick during the smart home system development and deployment*."

To showcase the use of the framework in the above scenario, the following explanation breakdown how the pilot framework for software sustainability design was used in creating the smart home system from the planning to requirement phase and finally delivery of system.

Phase 1 (project definition) with Karlskrona principles 1, 2 and 3: Henry uses the sustainable business canvas to show value that can be generated through sustainability consideration and how it can help improve the product. Henry was able to pinpoint two sustainability goals from this phase, which is design for sustainable efficiency and to create sustainability awareness through the smart home system by facilitating a community of users willing to share their energy usage to motivate each other.

Phase 2 (user requirements definition) with Karlskrona principle 2: from the information gathered in phase one and a discussion with the client, Henry was able to identify the goal of increasing the sustainability awareness among users of the system once it is created based on the sustainability helix concept. These were the indicators from this phase: percentage of reduced energy usage of the household, amount of feedback on the environmental impact of energy used (CO_2) by the family through eco feedback, number of suggestions provided on how to improve household energy usage based on usage patterns.

Phase 3 (system requirements definition) with Karlskrona principles 4 and 5: the goal in the phase of system requirements is to design for efficiency and sustainability awareness based on the overall system goal from phase 1. He uses the goal model to showcase how the system goals were broken into smaller piece based on the system requirements in order to identify requirement conflicts that might occur. Some of the smaller goals based on the overall goal in this phase include: reduce energy consumption, reduce CO_2 emissions, establish community of users sharing energy usage data, ensure high availability of system, and provide eco-feedback.

Phase 4 (analysis and design) with Karlskrona principles 2, 4, 6 and 8: in this phase, the main goals are design for easy usage, efficiency and sustainability awareness. Using the sustainability analysis diagram according to [24], Figure 3 portrays the sustainability analysis of the smart home system design for the first, second and third (immediate, enabling, and structural) impacts of smart home solutions from the different sustainability dimensions. Information from the analysis provides avenue for evaluating while guiding different stakeholders (managers, developers and users) on the benefits to aspire for sustainability in smart home solutions.

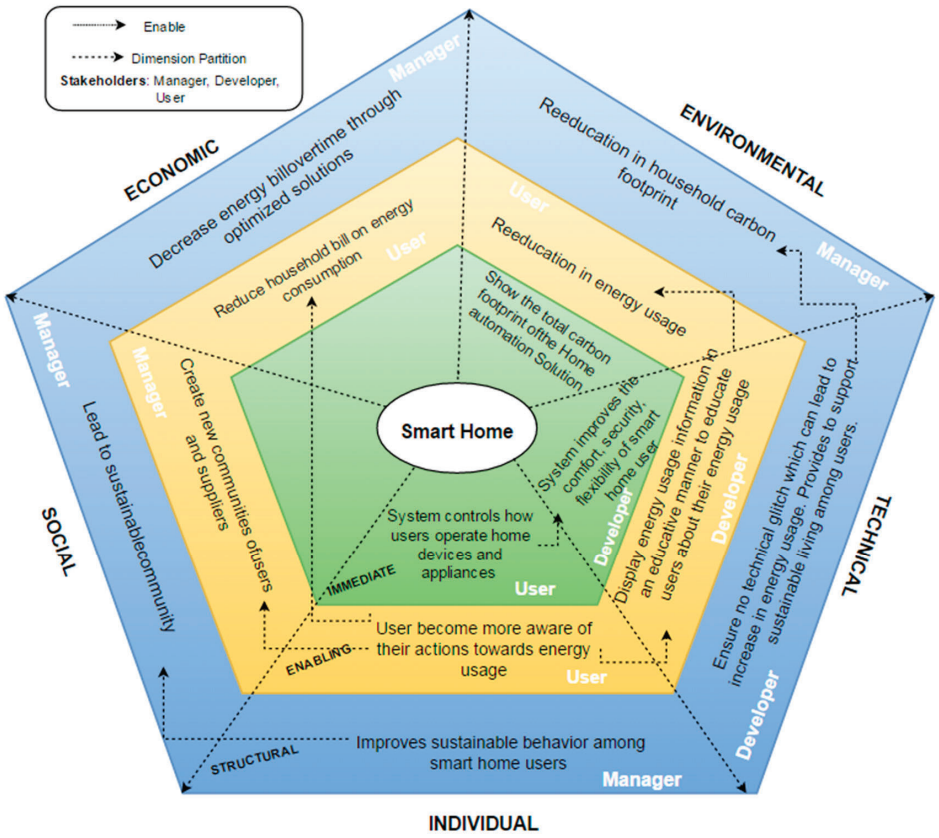


Figure 3. Immediate, enabling and structural effects of smart home solution in sustainability dimension.

The impacts described in Figure 3 are then taken into account during the requirements refinement phase in such a way that they can be implemented. The indicators from this phase are the immediate, enabling and structural impacts of the system identified in the sustainability analysis diagram.

Phase 5 (development) using Karlskrona principles 2 and 4). The goal of this phase is to implement a smart home system that induces sustainability awareness among users. The sustainability sub-attribute that influences developers during this stage is biomimicry. This concept encourage developers to rethink how to create functionality of the system that can reduce energy consumption, while providing eco-feedback that improves users' sustainability awareness and as an enabler for reducing household energy consumption. The biomimicry is made visible using an energy user interface (UI) dashboard for the smart home system where a fully grown tree is used to mimic energy consumption. As the energy consumption increases, the tree leaf starts to change colour to brown (indicating that is dead), when energy consumption decreases, the tree becomes greener (indicating the tree is back to life).

The following are functions added to the system which are used as indicators in this phase: percentage of energy saved during system usage, the accuracy in the eco-feedback based on the energy usage pattern of users, total number of times users shared their energy usage percentage with friends on social media, the level of user comprehension and understanding of the displayed information on CO_2 emission and indicator for amount of saved trees based on reduce energy usage over time.

Phase 6 (integration and testing) with Karlskrona principles 2 and 4: at this stage, the goal is to assembly and disassembly using the sustainability analysis chart (see Figure 3) including the indicators from the other phases as guide for integrating and testing the smart home system.

Phase 7 (implementation) with Karlskrona principles 5 and 7: the goals of this phase are design for easy use, induce conscious sustainability awareness and educate users about sustainability, which is influenced by the sustainability concept of biomimicry. These are used as indicators: the change in developers coding practice based on the effectiveness of functions added due to sustainability requirements such as percentage of energy saved during system usage, accuracy in the eco-feedback based on the energy usage pattern of users, understandability of presented information on CO_2 emission as well as amount of saved trees based on reduced energy usage over time.

Phase 8 (sustainment/maintenance) with Karlskrona principle 9): this phase covers the long-term goal of the smart home system such as serviceability and continuous user engagement. The indicators used in this phase are the efficiency of data generated in optimizing the smart home system, the effectiveness of the eco-feedback to improve user behaviour, as well as the total percentage of energy saved over time and the backlog management index (BMI).

One important question here is what the effects of the principles detailed in the SSDC for the smart home solution (see Tables 4 and 5) mean for a process engineer. For improving the software development process, the principles considered relevant for the example at hand were principles 2 (multi-dimension), 4 (independent of purpose), and 9 (long-term thinking). Principles 2, 4 and 9 are further explained here as a way to show how the Karlskrona manifesto principles influence decisions made during the development of the smart home example.

Principle 2 from the SSDC highlights the need for cross platform compatibility, which should be considered during the project definition of the software development life cycle (SDLC). It is also relevant during the user requirements elicitation phase when collecting their perceptions of sustainability.

Principle 4 emphasizes the need to educate smart home users about how their actions and reactions affect the environment when using smart home devices. Even if the users are not interested in sustainability, as software developers, designers and engineers it is their responsibility to inform users about the benefits of sustainability during the requirement gathering [91]. Efforts to educate users are also addressed during the documentation processes. The developer needs to create user documentation that include information about energy usage, as well as ways of saving other natural resources such as waste paper when printing.

Principle 9, which is about reducing production and solution costs, plays a vital role. It is important to identify choices that benefit both the current and future users, as well as how the solution can be cost beneficial when encouraging a wider population of users. The business analyst takes charge of this before moving on to the user requirements stage and this issue is monitored throughout the whole project development.

To illustrate how these principles are used, we consider the following second scenario:

> *A company named Energy Life, based on the SSDC analysis, provides a game like menu to control smart home devices for a family named Miralles. The family only wants to reduce household energy cost. Mark, the deployment manager, after reading the SSDC for smart homes (see Tables 4 and 5), pilot framework (see Figure 2), Table 6 (framework description) and the sustainability analysis for smart home solution (see Figure 3), realizes that the best way to reduce the energy cost of the Miralles is to implement a game-like menu for the Miralles to control their smart home devices. This provides information about the energy consumption of each home device. Within a few days the Miralles were able to see the flow of their energy usage and how their daily habits impact the unnecessarily high energy use within their household with impacts on the environment. The game also provides the family with tips on how they can save energy and the amount of CO_2 emissions. Members of the family are able to identify the amount of energy consumed by their washing machine. They decided not to run a half empty washing machine again as they can see that this happens almost every day,*

and even with the half-load mode it would make more sense to run a full one every two days instead. The Miralles also started a new habit of reminding each other to switch off the computer, TV and other household appliances when they are not in use.

Later on, when the company wants to update their system, developers revisit those orders of effect and the different dimensions. Developers also thought about what may and may not have changed in order to improve their system to be more efficient and effectively with regards to sustainability.

Using the definition of a smart home proposed by Nicholl et al. [92] "dwellings that use integrated communication systems to monitor and manage the performance of the home, and to support the lifestyle choices of the occupants", this scenario illustrates the following features to support sustainability in a smart home solution:

1. Automatic analysis of users' data to educate them and induce sustainable behaviour among users.
2. Alerting users through notification when electrical devices or appliances are running without being used.
3. Automatic scheduling of task such as washing cloths and dishes when energy rate is low during the day.
4. Planning when to turn on/off heating and lighting based on season and user behaviour (prediction).

7. Discussion

The SSDC provides a comprehensive overview of sustainability design considerations and requirements for systems and applications in different domains. SSDC guidelines are based on the analysis of information around the impact of different kinds of system on its application for sustainability dimensions-environment, society, individual, economy, technical and an overview of potential long-term consequences as seen in Tables 4 and 5, Tables A1 and A6 in the Appendix.

Table 4 for sustainability analysis for a cyber physical system (smart home) indicates areas where sustainability improvement can applied in smart home system design such as cross-platform compatibility and design for user awareness about sustainability. Indicators to evaluate these changes are also provided for stakeholders in this domain. Table 5, which presents all the sustainability dimensions order of impacts for cyber physical system (smart home), gives insight into the holistic overview of sustainability effects for smart home systems design and development. Tables A1–A6 in the Appendix includes other system types in the SSDC with information for how to better design those systems and evaluate their sustainability impacts. The SSDC as a catalogue that can inspire the development of tools and framework as shown in this article encourages the development of the pilot framework.

The pilot framework for sustainability of software system design exemplifies the use of the SSDC. The example for smart home in Section 6 shows how each of the development phases mapped with the Karlskrona manifesto and the sustainability goal inspired by different sustainability concepts such as biomimicry, sustainability helix provides better understanding of how sustainability can be centre of software design and development. The indicators from each phase of the SDLC while applying the pilot framework provide a way to evaluate the process and derivatives from each of the SDLC phase influenced by different sustainability concepts. The application of sustainability methods and tools used illustratively, such as sustainability business canvass, goal model and sustainability analysis diagram, provides software developers and requirements engineers with a way to structurally elicit and manage sustainability requirements and monitor system impacts (immediate, enabling and structural).

Specifically, in Section 6 we use the template for sustainability analysis by [24], where Figure 3 depicts an instance of such a sustainability analysis diagram for the smart home solution. During the analysis phase of SDLC, it provides a variety of information for different stakeholders for the direct, indirect and structural effects of sustainability in smart home design and deployment. This information

can then be used to create or enhance processes, methods and tools that can automate the incorporation of sustainability into the design of smart home solutions.

The software sustainability design catalogue and the underlying pilot framework can be beneficial for the following stakeholders interested in sustainability, its engineering and its integration in/for software systems design and development:

1. For companies and software developers, it serves as guide on how sustainability can be incorporated into software design and development. It can also enable them to identify the effects of their project on technical, economic, social, individual, and environmental sustainability. Furthermore, we support the current revision of the Association for Computing Machinery (ACM) code of ethics and propose to incorporate sustainability principles and explicitly acknowledge the need to consider sustainability as part of professional practice [23].
2. Standardization organizations can benefit from it to create future standards for software and organizational sustainability. SSDC shows areas where software applications can impact the environment and humans, and this information can help create standards that would encourage companies and stakeholders to improve existing and new applications and policies to promote sustainability.
3. Public authorities will be able to use the information from the catalogue to enact new laws persuading industry practitioners to design software systems, applications and devices in a more sustainable manner.
4. Academic institutions can identify avenues to advance research on sustainability by design, sustainability design patterns and tools to support, among others.

8. Conclusions

Effective sustainability engineering and integration requires clarifying the current perceptions of sustainability and defining a concrete framework for its engineering and measurement. As a first milestone, this paper presented a catalogue that quantifies sustainability via a series of guidelines that can be used for incorporating sustainability into the design loop.

By analysing how the principles defined in the Karlskrona Manifesto for Sustainability Design can be applied for some specific systems, we were able to identify a series of guidelines and develop the foundations for a "sustainability by design" approach. First, we reviewed the current perceptions of sustainability for various types of systems. Furthermore, based on how sustainability has been perceived in different software engineering disciplines, the SSDC has been defined. Each guideline is defined as a set of principles, dimensions of sustainability, orders of impact, and indicators. The usage and applicability of the catalogue have been demonstrated for four types of systems.

Future research includes examining other types of systems and the application of the guidelines in an industry setting. This will give better insights for the development of the guidelines for different types of systems and its usage by diverse stakeholders in the software development life cycle. An important aspect of its validity is that the catalogue was created based on the expertise of the wide set of researchers involved in the sustainability design manifesto. Considering the fact that sustainability in software engineering is still evolving, the SSDC provides common ground for further research.

An important limitation of the SDLC is its validity in industry. Consequently, the theoretical validation of the methodological aspects underlying the proposed guidelines will be considered beyond the industry evaluation to be conducted in future. The Appraisal of Guidelines for Research and Evaluation (AGREE) instrument [93], which is the appropriate tool that assesses the methodological rigor and transparency in which the guidelines is being developed, will used for this purpose.

The SSDC also has automation potential in the future. The design catalogue can become the basis for a recommender system. This would help developers to identify and apply effectively the sustainability guidelines. However, this requires more case studies for building a knowledge base required by a recommender system. The automation will provide a practical guide to enable developers

during each stage of design and development to understand and incorporate the Karlskrona manifesto principles, sustainability goals, concepts, tools and methods with indicators that can help in the evaluation of a software system.

Finally, this paper provides a foundation (via the SSDC and pilot framework) for the software engineering community to design and engineer sustainability into their systems.

Author Contributions: S.O. conceived the idea of the paper; S.O. and B.P. designed the research methodology; investigation and data collection: S.O.; data validation: B.P. and A.S.; S.O. wrote the paper and all authors contributed to reviewing all sections; review and editing supervision: B.P. and A.S.

Funding: This research received no external funding.

Acknowledgments: The authors are grateful to the researchers who initiated the Karlskrona manifesto principles which were used in this article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Tables A1 and A2 provide information on sustainability analysis of an embedded system (washing machine) using the Karlskrona principles and sustainability dimensions based on usage of order of impacts. Based on output from Tables A1 and A2, water and energy efficiency are key objectives of sustainability for a washing machine.

Tables A3 and A4 covers information about sustainability analysis for a mobile game (Angry Bird) centred on the Karlskrona principles and sustainability dimensions based on order of impacts. Table A3 provides points of energy efficiency, reduction of wear and tear of hardware and creating a sense of belong to community among users as key goals of sustainability for stakeholders. In addition, details from Table A4 prompt the need to aspire for these goals from all sustainability dimensions:

- **Environment**: optimize energy and computing resource consumption during game development and when users are playing game.
- **Economic**: provide continuous innovation on the game features to encourage current users to keep playing the game and attract new users and ensure game is maintainable (longevity).
- **Technical**: ensure that game does not encourage quick hardware wear and tear and at same time has the ability to evolve with new demands of the market.
- **Individual**: good user experience while interacting with the game and serves as a medium of inducing sustainable behaviour.
- **Social**: create good community sense among angry bird users and educate them about sustainability.

Tables A5 and A6 presents information on sustainability analysis of desktop application (Microsoft Office) using the Karlskrona principles and sustainability dimensions based on order of impacts.

Table A1. Sustainability analysis of embedded system (washing machine) based on Karlskrona principles.

Karlskrona Principles and Goal	Current Principle Usage	Future Principle Usage	Stakeholders	Question	Indicator
(P 9) Efficient water usage	Current washing machine has some design features to help reduce water wastage during washing circle.	Good mechanism within the washing machine to aid efficient use of water during washing cycle and also display the amount of water saved to users. This will serve as means of educating users about water wastage. This will aid positive impact on the amount of water usage in a household.	Software developer	Does the washing machine reduce water usage?	Washing cycle/total amount of water used.
(P 6, 8) Energy Efficiency	Some current sets of washing machine have eco-friendly features to reduce energy usage.	One good feature to reduce energy usage of washing is to turn off or hibernate automatically after washing cycle if idle for 2 min. It will help reduce energy cost (P 6) and also reduce resource usage in the long term (P 8). This will help reduce energy consumption when machine is idle. Incorporate the use of scheduler to the washing machine as a way to time when the washing cycle should start during the period of the day when energy cost is less.	Software developer	Does the machine use too much energy for a single washing cycle?	Energy efficiency (washing cycle/total amount of energy used).
(P 8, 9) Water efficiency	Allows collection of grey water from washing machine.	Encourage reuse of greywater (with biodegradable laundry detergent) in garden watering.	Business analyst	How much of the greywater gets reused afterwards?	Percentage (%) of reused water.

Table A2. Sustainability dimensions order of impacts for embedded system (washing machine).

Order of Impacts	Environment	Economic	Technical	Social	Individual
1st	Increase in the use of natural resources in the production of washing machine components such as iron, copper, medium-density fibreboard, ethylene propylene diene monomer (EPDM) rubber.	High demand for natural resources boosts the economies of countries with those natural resources.	Increase demand for new technologies, tools and equipment for extracting raw materials.	New job opportunities for people.	Increase the risk of having skin diseases due to toxic material exposure.
2nd	Increase in water and energy usage when using a washing machine as opposed to manual washing.	High energy cost for household as a result of increase in energy usage due to ease of washing cloths (convenience factor and little manual labour).	Demand for energy and water saving mechanism in washing machine.	More job opportunities for technicians with knowledge of washing machine technologies.	Increase the ease of washing for users.
3rd	Over a long time it lead to increase water usage and a culture of washing a lot (e.g., California). In turn, that leads to a higher wear and tear of the hardware.	Increase in profit for washing machine production companies with high demand for washing machine from users.	Over time there will be pressure to build a more energy and water efficient washing machine from manufacturers to have a competitive edge over other competitors.	Increase job creation over time both from industry for skilled workers and in household for technicians to fix minor issues of washing machine.	Over a long time it leads to increased water usage and a culture of washing a lot (e.g., California). In turn, that leads to a higher wear and tear of the hardware

Table A3. Sustainability analysis of mobile games (Angry Bird) based on Karlskrona principles.

Karlskrona Principles and Goal	Current Principle Usage	Future Principle Usage	Stakeholders	Question	Indicator
(P 1) Energy Efficient	Currently game development focus more on usability and fun factor aspect than sustainability aspect.	Create the mobile game architecture with sustainability consideration. Since sustainability is systemic, it should be core of the application structure. Consider energy efficiency during game development. Incorporate green patterns to game application development.	Software developer	Is the mobile energy Efficient?	(Energy efficiency) useful-work done/energy used.
(P 1) Reduce wear and tear of hardware	Although sustainability is not the core of current game development practices, although game developer tries to ensure optimal use of hardware resources to make game run faster and better on hardware (phone, computer, tablets).	Ensure that the mobile game during operation uses hardware resources (memory, CPU etc.) in efficient and sustainable manner to reduce wear and tear.	Software developer	What is the impact of the game on hardware components like CPU and RAM?	Does the game use too much hardware resources?
(P 2) Sense of belonging to a community and connectedness to other people	Current game development provide sort of community for it users in form of forums and groups online.	There is a digital community surrounding most games and the question is how it compares to face to face communities for gamers. Create a community that make gamers feel connected to both the digital and real worlds. Make gamers feel like they are part of something (people always want to belong to a tribe).	Business analysts, software developer	Is there a sense of community amongst players?	Connectedness in community? Number of 'friends' in a particular gaming community?
(P9) Good user experience	Current game development incorporates user experience into their production to gain more user base and profit.	Game should use reasonable lighting effect for the game display which can help reduce energy usage and also incorporate sustainability concept in the total overall design of the game. It can also add features that will educate users about sustainability.	User experience (UX) engineer and software developer	Can users complete their task easily?	Gateway metrics.

Table A4. Sustainability dimensions order of impacts for mobile games (Angry Bird).

Order of Impacts	Environment	Economic	Technical	Social	Individual
1st	Increase in energy usage from computers and mobile devices used for developing and testing game application.	Cost of production for game development companies.	Increase in demand for sophisticated hardware and software for game development.	Open job opportunities for game developers.	Provides avenues for leisure activities for users.
2nd	Increase in energy usage because users are using mobile phone, laptops, iPad to play game. There will also be need for charging of these devices coupled with the energy consumption when playing the game.	Company make profit from game purchase. Increase demand and user cost for hardware (computers, phones and tablets).	Demand for better graphics and quick game response from users.	Create community sense among Angry Bird users.	Demand for good user experience while interacting with the game.
3rd	Over time leads to hardware wear and tear because of continuous game time from user side on computer/phones and continuous game development from game production company side.	Increase in profit from huge user and fan base for the game company.	Demand for newer features and innovation from customers	Increase the sense of belonging in form of community among users.	Lead to game addiction and lack of social interaction with outside world.

Table A5. Sustainability analysis of desktop application (Microsoft Office) based on Karlskrona principles.

Karlskrona Principles and Goal	Current Principle Usage	Future Principle Usage	Stakeholders	Question	Indicator
(P1) Incorporate sustainability into development process.	Current development framework allow the use of scrum and version control.	Provide a development framework that support sustainability focusing on the software and those developing the software itself.	(Development) process engineer	Are guidelines available?	Boolean (Yes or No).
(P8) Offer reasonable amount of features.	Currently all office application comes with all the features which sometimes are rarely used by users.	Provide all basic features for office application and allow users to add other features when needed.	Business analyst and software developer	What is the number of changes to be made to add new features?	Rework metric.
(P9) Add green print (and let user know how many pages they save over time).	Users can print any document as their need requires.	Incorporate green print to office application that inform users whenever they want to print a document for the second time that they can skip few pages because changes were not made on those pages.	Software developer	Do people print less by using this green print button?	Number of pages printed compared to if there was no green button.

Table A6. Sustainability dimensions order of impacts for desktop application (Microsoft Office).

Order of Impacts	Environment	Economic	Technical	Social	Individual
1st	Reduce the amount of manual writing on papers which in turn reduces the amount of paper used by people for writing.	Open new potential market for company to explore.	Ensure the ability of Word application to meet new market demands from the technical implementation aspect.	Creates new job opportunities for those that are expert with most of the office applications.	Provide efficient way of doing daily task such as documentation, project management and design.
2nd	Reduce paper resource wastage.	Increase in profit for Microsoft and other partner companies through sales of office application.	Increase in use of computers for development of add-ons and plugins for office applications.	Provides a community feeling among users.	Guarantee that user will easily finish their task while using the application for their day-to-day work.
3rd	It will increase energy usage over time due to ease of doing daily task on computers, phones and tablets.	Rise in company's profit through product innovation as demands change from generation to generation.	Demand for new features to meet new demands.	Improves connectedness among users through collaboration online.	Improves the overall ability of users completing their task (documentation, project management and design).

References

1. Microsoft. Microsoft 2015 Citizenship Report Letter from Our CEO. 2015. Available online: http://download.microsoft.com/download/7/3/6/736CED21-9D8B-4CBB-98E8-DCBAE7026251/Microsoft%202015%20Citizenship%20Report.pdf (accessed on 20 March 2018).
2. IBM. Global CEO Study: The Enterprise of the Future. Available online: https://www-935.ibm.com/services/uk/gbs/pdf/ibm_ceo_study_2008.pdf (accessed on 11 October 2017).
3. Nidumolu, R.; Prahalad, C.K.; Rangaswami, M.R. Why sustainability is now the key driver of innovation. *IEEE Eng. Manag. Rev.* **2013**, *41*, 30–37. [CrossRef]
4. Firstcarbon Solutions. Constructing Sustainable Business Models. Available online: http://www.firstcarbonsolutions.com/resources/newsletters/january-2014-constructing-sustainable-business-models/constructing-sustainable-business-models/ (accessed on 18 June 2018).
5. Knut, H.; Martin, R.; Ingridvon, S.; Michael, A.; David, K.; Nina, K. Sustainability Nears a Tipping Point Sustainability Nears a Tipping Point. *MIT Sloan Manag. Rev.* **2012**, *53*, 69–74.
6. Calero, C.; Piattini, M. Introduction to Green in software engineering. *Green Softw. Eng.* **2015**, 3–27. [CrossRef]
7. Becker, C.; Betz, S.; Chitchyan, R.; Duboc, L.; Easterbrook, S.M.; Penzenstadler, B.; Ssyff, N.; Venters, C.C. Requirements: The key to sustainability. *IEEE Softw.* **2016**, *33*, 56–65. [CrossRef]
8. Koçak, S.A.; Alptekin, G.I.; Bener, A.B. Evaluation of software product quality attributes and environmental attributes using ANP decision framework. *CEUR Workshop Proc.* **2014**, *1216*, 37–44.
9. Ericsson. Technology for Good. Available online: https://www.ericsson.com/assets/local/about-ericsson/sustainability-and-corporate-responsibility/documents/2015-corporate-responsibility-and-sustainability-report.pdf (accessed on 30 November 2014).
10. Ericsson. Energy and Carbon Report. 2013. Available online: http://www.ericsson.com/res/docs/2013/ericsson-energy-and-carbon-report.pdf (accessed on 11 November 2017).
11. Digiconomist. Bitcoin Energy Consumption Index. Available online: https://digiconomist.net/bitcoin-energy-consumption (accessed on 18 June 2018).
12. Elbahrawy, A.; Alessandretti, L.; Kandler, A.; Pastor-satorras, R. Evolutionary dynamics of the cryptocurrency market. *R. Soc. Open Sci.* **2017**, 1–16. [CrossRef] [PubMed]
13. Chris, P.; Dan, S.; Eli, B. Understanding and Mitigating the Effects of Device and Cloud Service Design Decisions on the Environmental Footprint of Digital Infrastructure. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16), San Jose, CA, USA, 7–12 May 2016; ACM/IEEE: New York, NY, USA, 2016; pp. 1324–1337.
14. Al Hinai, M.; Chitchyan, R. Engineering Requirements for Social Sustainability. 2016. Available online: http://www.cs.le.ac.uk/people/rc256/ict4s2016_hinai_chitchyan.pdf (accessed on 20 June 2018).
15. Penzenstadler, B. Software Engineering for Sustainability. Available online: https://www.ics.uci.edu/~djr/DebraJRichardson/SE4S.html (accessed on 20 June 2018).
16. Blevis, E.; Preist, C.; Schien, D.; Ho, P. Further Connecting Sustainable Interaction Design with Sustainable Digital Infrastructure Design. In Proceedings of the 2017 Workshop on Computing Within Limits (LIMITS'17), Santa Barbara, CA, USA, 22–24 June 2017; ACM/IEEE: New York, NY, USA, 2017; pp. 71–83.
17. Penzenstadler, B.; Femmer, H. A Generic Model for Sustainability with Process-and Product-Specific Instances. In Proceedings of the 2013 workshop on Green in/by software engineering (GIBSE'13), Fukuoka, Japan, 26 March 2013; ACM: New York, NY, USA, 2013; pp. 3–7.
18. Jeanette Schwarz, E.B.; Beloff, B. Use Sustainability Metrics to Guide Decision Making. *CEP* **2002**, *98*, 58–63.
19. Penzenstadler, B. What does Sustainability mean in and for Software Engineering? In Proceedings of the 1st International Conference on ICT for Sustainability (ICT4S), Zurich, Switzerland, 14–16 February 2013.
20. Venters, C.C.; Jay, C.; Lau, L.M.S.; Griffiths, M.K.; Holmes, V.; Ward, R.R.; Austin, J.; Dibsdale, C.E.; Xu, J. Software sustainability: The modern tower of babel. In Proceedings of the Third International Workshop on Requirements Engineering for Sustainable Systems Co-Located with 22nd International Conference on Requirements Engineering (RE 2014), Karlskrona, Sweden, 26 August 2014.
21. Ehrenfeld, J.R. *Sustainability by Design: A Subversive Strategy for Transforming Our Consumer Culture*; Yale University Press: New Haven, Ct, USA; London, UK, 2008.
22. Freeman, P. The Central Role of Design in Software Engineering: Implications for Research. *Softw. Eng. Res. Dir.* **1980**, 121–132. [CrossRef]

23. Becker, C.; Chitchyan, R.; Duboc, L.; Easterbrook, S.; Mahaux, M.; Penzenstadler, B.; Rodriguez-Navas, G.; Salinesi, C.; Seyff, N.; Venters, C.; et al. The Karlskrona manifesto for sustainability design. *Softw. Eng.* **2014**, *20*, 2014.
24. Becker, C.; Chitchyan, R.; Duboc, L.; Easterbrook, S.; Penzenstadler, B.; Seyff, N.; Venters, C.C. Sustainability Design and Software: The Karlskrona Manifesto. In Proceedings of the 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, Italy, 16–24 May 2015.
25. Saval, G.; Mahaux, M.; Heymans, P. Discovering Sustainability Requirements: An Experience Report. In Proceedings of the 20th International Working Conference on Requirements Engineering: Foundation for Software Quality, Essen, Germany, 7–10 April 2014.
26. Chitchyan, R.; Betz, S.; Duboc, L.; Penzenstadler, B.; Ponsard, C.; Venters, C.C. Evidencing Sustainability Design Through Examples. 2015. Available online: http://eprints.hud.ac.uk/id/eprint/25699/1/Session2Paper3.pdf (accessed on 28 June 2018).
27. Groher, I.; Weinreich, R. An Interview Study on Sustainability Concerns in Software Development Projects. In Proceedings of the 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, Austria, 30 August–1 September 2017.
28. Pargman, D.; Eriksson, E.; Höjer, M.; Östling, U.G.; Borges, L.A. The (Un)sustainability of Imagined Future Information Societies. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17), Denver, CO, USA, 6–11 May 2017; ACM: New York, NY, USA, 2017; pp. 773–785.
29. Ilstedt, S.; Eriksson, E.; Hesselgren, M.I.A. Sustainable Lifestyles–How Values Affect Sustainable Practises. 2017. Available online: http://www.nordes.org/nordes2017/assets/full_papers/nordes17a-sub1039-cam-i26_ILSTEDT_v2.pdf (accessed on 28 June 2018).
30. Shedroff, N. Design is the Problem: The Future of Design Must be Sustainable. 2009, p. 582. Available online: https://designethosandaction.files.wordpress.com/2015/01/design-is-the-problem.pdf (accessed on 28 June 2018).
31. Bibri, M. Sustaining ICT for Sustainability. 2009. Available online: https://www.diva-portal.org/smash/get/diva2:833352/FULLTEXT01.pdf (accessed on 28 June 2018).
32. Eriksson, E.; Pargman, D.; Bates, O.; Normark, M.; Gulliksen, J.; Anneroth, M.; Berndtsson, J. HCI and UN's Sustainable Development Goals: Responsibilities, Barriers and Opportunities. In Proceedings of the 9th Nordic Conference on Human-Computer Interaction (NordiCHI'16), Gothenburg, Sweden, 23–27 October 2016; ACM: New York, NY, USA, 2016; pp. 140:1–140:2.
33. Vister, K.K.; Evans, R.D. Identifying Contributing Factors to Sustainability Awareness in the Norwegian Software Industry. In Proceedings of the 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran, 22 December 2017.
34. Musthaler, L. Energy-Aware Software Design Can Reduce Energy Consumption by 30% to 90%. Network World. 2014. Available online: https://www.networkworld.com/article/2861005/green-it/energy-aware-software-design-can-reduce-energy-consumption-by-30-to-90.html (accessed on 28 June 2018).
35. Durdik, Z.; Klatt, B.; Koziolek, H.; Krogmann, K.; Stammel, J.; Weiss, R. Sustainability Guidelines for Long-Living Software Systems. In Proceedings of the 2012 28th IEEE International Conference on Software Maintenance (ICSM), Trento, Italy, 23–28 September 2012; pp. 517–526.
36. Malik, M.N.; Khan, H.H. Investigating Software Standards: A Lens of Sustainability for Software Crowdsourcing. *IEEE Access* **2018**, *6*, 5139–5150. [CrossRef]
37. Nina, W.; Patricia, L.; Francesco, O. Sustainability in Software Engineering. 2017. Available online: http://dl.ifip.org/db/conf/ifip6-3/sustainit2017/08_P08_S21_SustainIT2017.pdf (accessed on 28 June 2018).
38. Robillard, M.P. Sustainable Software Design. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Seattle, WA, USA, 13–18 November 2016; ACM: New York, NY, USA; pp. 920–923.
39. Lago, P.; Koçak, S.A.; Crnkovic, I.; Penzenstadler, B. Framing Sustainability as a Property of Software Quality. *Commun. ACM* **2012**, *55*, 56–64. [CrossRef]
40. Chitchyan, R.; Duboc, L.; Becker, C.; Betz, S.; Penzenstadler, B.; Venters, C.C. Sustainability Design in Requirements Engineering: State of Practice. In Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), Austin, TX, USA, 14–22 May 2016.

41. Calienes, G.G. Requirements Prioritization Framework for developing Green and Sustainable Software Using ANP—Based Decision Making. 2013. Available online: http://enviroinfo.eu/sites/default/files/pdfs/vol7995/0327.pdf (accessed on 28 June 2018).
42. Penzenstadler, B. RE4ES: Support Environmental Sustainability by Requirements Engineering. 2012. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.382.9172&rep=rep1&type=pdf (accessed on 28 June 2018).
43. Bozzelli, P.; Gu, Q.; Lago, P. A Systematic Literature Review on Green Software Metrics. 2013. Available online: http://dare.ubvu.vu.nl/bitstream/handle/1871/52417/SLR?sequence=1 (accessed on 28 June 2018).
44. Kern, E.; Dick, M.; Naumann, S.; Guldner, A.; Johann, T. Green Software and Green Software Engineering—Definitions, Measurements, and Quality Aspects. In Proceedings of the First International Conference on Information and Communication for Sustainability, Zurich, France, 14–16 February 2013; pp. 87–94.
45. Johann, T.; Dick, M.; Kern, E.; Naumann, S. Sustainable Development, Sustainable Software, and Sustainable Software Engineering: An Integrated Approach. In Proceedings of the 2011 International Symposium on Humanities, Science and Engineering Research, Kuala Lumpur, Malaysia, 6–7 June 2011; pp. 34–39.
46. Sawyer, P.; Paech, B.; Heymans, P. (Eds.) Requirements Engineering: Foundation for Software Quality. In Proceedings of the 13th International Working Conference (REFSQ 2007), Trondheim, Norway, 11–12 June 2007; pp. 247–261.
47. Dick, M.; Naumann, S. Enhancing Software Engineering Processes Towards Sustainable Software Product Design. 2010. Available online: https://pdfs.semanticscholar.org/98d0/4de0318b252f273e3cf36cc385253542b924.pdf (accessed on 28 June 2018).
48. Erdélyi, K. Special Factors of Development of Green Software Supporting Eco Sustainability. In Proceedings of the 2013 IEEE 11th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 26–28 September 2013.
49. Oyedeji, S.; Seffah, A.; Penzenstadler, B. Sustainability Quantification in Requirements Informing Design. 2017. Available online: http://ceur-ws.org/Vol-1944/paper6.pdf (accessed on 28 June 2018).
50. Penzenstadler, B. Infusing Green: Requirements Engineering for Green in and through Software Systems. 2014. Available online: http://ceur-ws.org/Vol-1216/paper8.pdf (accessed on 28 June 2018).
51. Roher, K.; Richardson, D. Sustainability Requirement Patterns. In Proceedings of the 2013 3rd International Workshop on Requirements Patterns (RePa), Rio de Janeiro, Brazil, 15–19 July 2013.
52. Colmant, M.; Rouvoy, R.; Seinturier, L. Improving the Energy Efficiency of Software Systems for Multi-Core Architectures. In Proceedings of the 11th Middleware Doctoral Symposium, Bordeaux, France, 8–9 December 2014.
53. Lami, G.; Buglione, L. Measuring Software Sustainability from a Process-Centric Perspective. In Proceedings of the 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement, Assisi, Italy, 17–19 October 2012.
54. Calero, C.; Bertoa, M.F.; Moraga, M.A. A Systematic Literature Review for Software Sustainability Measures. In Proceedings of the 2013 2nd International Workshop on Green and Sustainable Software (GREENS), San Francisco, CA, USA, 20 May 2013.
55. Seacord, R.; Elm, J.; Goethert, W.; Lewis, G.A.; Plakosh, D.; Robert, J.; Wrage, L.; Lindvall, M. Measuring Software Sustainability. In Proceedings of the International Conference on Software Maintenance, Amsterdam, The Netherlands, 22–26 September 2003.
56. Penzenstadler, B.; Martin, M.; Camille, S. RE4SuSy: Requirements Engineering for Sustainable Systems. *J. Syst. Softw.* **2013**, *995*. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.415.8601&rep=rep1&type=pdf (accessed on 28 June 2018).
57. Christoph, B. Sustainability and longevity: Two sides of the same quality? *CEUR Workshop Proc.* **2014**, *1216*, 1–6.
58. Becker, C. Website for The Karlskrona Manifesto for Sustainability Design. 2014. Available online: http://sustainabilitydesign.org/karlskrona-manifesto/ (accessed on 10 October 2017).
59. Fowler, M.; Highsmith, J. The agile manifesto. *Softw. Dev.* **2001**, *9*, 28–35.
60. Bussiness Rules Group. The Business Rules Manifesto. Available online: http://www.businessrulesgroup.org/brmanifesto.htm (accessed on 12 November 2017).

61. Arsanjani, A.; Booch, G.; Boubez, T.; Brown, P.; Chappell, D.; de Vadoss, J.; Loesgen, B. The SOA Manifesto. *SOA Manifesto* **2009**, *35*, 82–88.
62. Felikson, L. The SOA Manifesto: Establishing a Common Understanding about SOA and Service-Orientation History of The SOA Manifesto. 2010. Available online: http://2010.secrus.org/wp-content/uploads/download/Felikson.pdf (accessed on 28 June 2018).
63. Gent, I. The Recomputation Manifesto. Available online: https://www.software.ac.uk/blog/2016-10-05-recomputation-manifesto (accessed on 12 November 2016).
64. Langer, A.M. Guide to Software Development—Designing and Managing the Life Cycle. 2016, p. 402. Available online: https://www.springer.com/la/book/9781447167976 (accessed on 28 June 2018).
65. Berkhout, F.; Hertin, J. Impacts of Information and Communication Technologies on Environmental Sustainability: Speculations and Evidence. Available online: http://www.oecd.org/sti/inno/1897156.pdf (accessed on 28 June 2018).
66. Hilty, L.M.; Aebischer, B. ICT for Sustainability: An Emerging Research Field. In *ICT Innovations for Sustainability*; Advances in Intelligent Systems and Computing; Hilty, L., Aebischer, B., Eds.; Springer: Cham, Switzerland, 2015; Volume 310.
67. ETH Sustainability Summer School 2011. Washing Machine. 2011. Available online: http://webarchiv.ethz.ch/sustainability-v2/lehre/Sommerakademien/so2011/washies_report.pdf (accessed on 10 December 2017).
68. Consumer Reports. Washing Machines That Save Water and Money. 2017. Available online: https://www.consumerreports.org/cro/news/2015/04/washing-machines-that-save-water-and-money/index.htm (accessed on 28 June 2018).
69. National Geographic. Washing Machines Buying Guide. 2017. Available online: http://environment.nationalgeographic.com/environment/green?guide/washing?machine?buying?guide/environmental?impact/ (accessed on 5 May 2017).
70. Jack, T. The dirt on clothes: Why washing less is more sustainable. *J. Home Econ. Inst. Aust.* **2013**, *20*, 44.
71. Ross, C. The Damage I Cause When I Wash My Clothes. 2015. Available online: https://theswatchbook.offsetwarehouse.com/2015/07/16/environmental-impact-of-the-washing-machine/ (accessed on 28 June 2018).
72. Markham, D. How to Reuse Grey Water in the Home and Yard. 2017. Available online: https://www.treehugger.com/green-home/how-reuse-grey-water-home-and-yard.html (accessed on 28 June 2018).
73. R Entertainment. UN Honours 'Angry Birds, Happy Planet' Campaign. 2017. Available online: http://www.goodnewsfinland.com/un-honours-angry-birds-happy-planet-campaign/ (accessed on 28 June 2018).
74. Deloittte, G. 2016 Mobile Industry Impact Report: Sustainable Development Goals. 2016. Available online: https://www.gsma.com/betterfuture/wp-content/uploads/2016/09/_UN_SDG_Report_FULL_R1_WEB_Singles_LOW.pdf (accessed on 28 June 2018).
75. Arndt, H.; Dziubaczyk, B.; Mokosch, M. *Information Technology in Environmental Engineering*; Springer International Publishing: Cham, Switzerland, 2014; pp. 13–24.
76. Sony Pictures. Make the Angry Birds Happy! Take Climate Actio Send a Tweet, Record a Vine, Take a Photo That Shows Raising Your Voice for Good Cause. Available online: http://www.angrybirdshappyplanet.net/ (accessed on 13 October 2017).
77. Sony Pictures. The Angry Birds Movie Themed Angry Birds for a Happy Planet. Available online: http://www.sonypicturesgreenerworld.com/articles/angry-birds-for-a-happy-planet (accessed on 14 October 2017).
78. Accenture. Microsoft, Accenture and WSP Environment & Energy Study Shows Significant Energy and Carbon Emissions Reduction Potential from Cloud Computing. Available online: https://newsroom.accenture.com/article_display.cfm?article_id=5089 (accessed on 20 November 2017).
79. Microsoft. Microsoft Products Our Products are Empowering People and Organizations to Achieve More While Improving Efficiency and Reducing Carbon Emissions. Cloud Services. Available online: https://www.microsoft.com/about/csr/environment/solutions/cloud/ (accessed on 11 November 2017).
80. Ahmad, I.; Ranka, S. (Eds.) *Handbook of Energy-Aware and Green Computing–Two Volume Set*; Chapman and Hall/CRC: New York, NY, USA, 2012.
81. Accenture. Cloud Computing and Sustainability: The Environmental Benefits of Moving to the Cloud. Available online: http//download.microsoft.com/download/A/F/F/AFFEB671-FA27-45CF-9373-0655247751CF/CloudComput.Sustain.-Whitepaper-Nov2010.pdf (accessed on 22 August 2017).

82. Queensland Department of Public Works. Smart and Sustainable Homes Design Objectives. 2016. Available online: http://www.hpw.qld.gov.au/construction/sustainability/smartsustainablehomes/Pages/Default.aspx (accessed on 28 June 2018).
83. Wilson, C.; Hargreaves, T.; Hauxwell-Baldwin, R. Benefits and risks of smart home technologies. *Energy Policy* **2017**, *103*, 72–83. [CrossRef]
84. Bhati, A.; Hansen, M.; Chan, C.M. Energy conservation through smart homes in a smart city: A lesson for Singapore households. *Energy Policy* **2017**, *104*, 230–239. [CrossRef]
85. Monaghan, A. Seven Things You Need to Know about the UK Economy. Available online: https://www.theguardian.com/sustainable-business/2015/apr/17/things-need-know-sustainable-smart-technology (accessed on 18 December 2017).
86. Solanki, V.K.; Muthusamy, V.; Katiyar, S. Think Home: A Smart Home as Digital Ecosystem. *Circuits Syst.* **2016**, *7*. [CrossRef]
87. Li, R.Y.M. The Usage of Automation System in Smart Home to Provide a Sus-Tainable Indoor Environment: A Content Analysis in Web 1.0. *Int. J. Smart Home* **2013**, *7*, 47–60.
88. Barker, S.; Mishra, A.; Irwin, D.; Cecchet, E.; Shenoy, P.; Albrecht, J. Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Home. 2012. Available online: http://www.ecs.umass.edu/~deirwin/sustkdd12.pdf (accessed on 28 June 2018).
89. Makonin, S.; Bartram, L.; Popowich, F. A Smarter Smart Home: Case Studies of Ambient Intelligence. *IEEE Pervasive Comput.* **2013**, *12*, 58–66. [CrossRef]
90. Kramer, K.-L. *User Experience in the Age of Sustainability*; Morgan Kaufmann: Burlington, MA, USA, 2012.
91. Spinellis, D. The Social Responsibility of Software Development. *IEEE Softw.* **2017**, *34*, 4–6. [CrossRef]
92. Nicholl, A.; Perry, M. *Smart Home Systems and the Code for Sustainable Homes: A BRE Guide*; IHS BRE Press: Bracknell, UK, 2009; ISBN 9781848061125.
93. Brouwers, M.C.; Kho, M.E.; Browman, G.P.; Burgers, J.S.; Cluzeau, F.; Feder, G.; Fervers, B.; Graham, I.D.; Grimshaw, J.; Hanna, S.; et al. AGREE II: Advancing guideline development, reporting and evaluation in healthcare. *Can. Med. Assoc. J.* **2010**, *182*, E839–E842. [CrossRef] [PubMed]

Publication III

S. Oyedeji, A. Seffah, and B. Penzenstadler
Classifying the Measures of Software Sustainability

In: *4th International Workshop on Measurement and Metrics for Green and Sustainable Software Systems co-located with 12th International Symposium on Empirical Software Engineering and Measurement (ESEM 2018)*

Reprinted with permission from
CEUR Workshop Proceedings
Vol. 2286, pp. 19-25, 2018

Classifying the Measures of Software Sustainability

Shola Oyedeji
LUT School of Engineering (LENS)
Lappeenranta University of Technology
Lappeenranta, Finland
shola.oyedeji@lut.fi

Ahmed Seffah
LUT School of Engineering (LENS)
Lappeenranta University of Technology
Lappeenranta, Finland
ahmed.seffah@lut.fi

Birgit Penzenstadler
Department of Computer Engineering and Computer Science California State University Long Beach (CSULB) Long Beach, California, USA
birgit.penzenstadler@csulb.edu

***Abstract*— Energy efficiency is one of the very few measures widely used for evaluating green and sustainable software systems. This paper investigates the current measures of software sustainability from the four different software sustainability perceptions: Sustainability in Software Development, Green Software Systems, Software for Sustainability, Sustainability of the Software Eco System and Software Sustainability Dimensions (Economic, Social, Individual, Technical and Environment). While exploring the literature on green and sustainable software systems, measures of green software and software sustainability were identified, compiled and classified according to the four sustainability perceptions.**

Keywords*— *green software, sustainable software, measures, sustainability, sustainability perceptions, green measures, software measurement.

I. INTRODUCTION

Sustainability is now one of the world major challenge [1][2]. The United Nations Sustainable development Goals (SDGs) shows the importance of sustainability in all facet of human lives and development. Today's economy rely on information and communications technology (ICT) in which software is a key factor and catalyst for all economic activities and a major driver linking all sectors. As stated in an Ericsson report that ICT can help reduce the global greenhouse gas (GHG) emissions by 15% [3]. Currently ICT itself contributes an estimated 2% to the global CO_2 emissions and accountable for approximately 8% of the European Union (EU) electricity consumption [4]. This shows ICT has a huge potential to help support sustainability and Green [5] but at same time it is important to explore avenues to make ICT domain more green and sustainable because of its huge impact on sustainability. Finding ways to properly evaluate software in regards to green and sustainability will provide avenues to reduce the current negative impacts of ICT.

This research explores the ongoing perceptions in the software engineering domain with the goal to identify the current and future measures used in the evaluation of green and sustainable software. Via triangulation of data from diverse sources, the measures are clustered into the four perceptions of sustainability in software engineering and sustainability dimensions. The long term goal of this research is to answer the following challenging questions: what are the current measures used in evaluating green and sustainability aspects of software systems and how can these measures be grounded in the software sustainability measurement theory.

II. BACKGROUND

A. Sustainability in Software Development

As a measurable attribute, software sustainability is more than the perceptions of capacity to endure [6]. Sustainable software measures should include the direct and indirect negative impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software [7]. It is also beyond the current focus of sustainability in requirements engineering where sustainability is considered as a nonfunctional requirement (NFR) by some [8][9][10]. In [2], the authors reported on a software project in which sustainability requirements were treated as quality requirements, and systematically elicited and documented. Another work also proposed an approach to tackle sustainability during software systems development and maintenance that decomposes sustainability into four aspect in software development life cycle such as the development process, maintenance process, system production and system usage [11]. This approach is useful for a process engineer who instantiates this approach for a software development company or requirements engineer who instantiates it for a specific system under development.

The Software Sustainability Design Catalogue (SSDC) that quantifies sustainability via a series of guidelines used for incorporating sustainability into the design loop for software system. The SSDC is created to promote effective sustainability engineering and integration in phases of software development life cycle. Design according to the authors Oyedeji et al. [12] is a good way to achieve sustainability in software development.

Furthermore a checklist and guide approach that demonstrates how to include the objective of environmental sustainability from the very early steps of software development can assist in identifying key stakeholders. This will facilitate the ability to accommodate new objectives of improving the environmental sustainability of software systems [13]. Roher et al. [14] suggests the use of sustainability requirement patterns (SRPs), which will provide software engineers with guidance on how to write specific types of sustainability requirements with the goal to overcome the barriers of incorporating environmental sustainability into the requirements engineering process.

B. Green software system

Green software is an environmentally friendly software that consumes less energy, provides less impacts on environment and support carbon management [15]. It is also software that fulfils high level requirements, ensuring the software engineering process, maintenance, and disposal saves and/or reduces resource waste [16] [17]. Green software is divided into four parts: software that is energy efficient during

execution, software that are embedded to execute and support smart operations in green manner, software to produce environment viable products and policies [18]. The goal of green software engineering is to provide supports for efficient consumption of natural resources while continuously monitoring, evaluating and optimizing the aftermath effects caused during the software system life cycle [19].

Erdélyi [20] paper provides an overview of different activities and advice on what to do in order to develop green software which uses energy efficiently and produce less waste. The paper highlights three ways software engineering can be green such as: produce green software, produce software to support environmentally consciousness (green by software) and produce less waste during development.

Dick et al. [21] provides basis for the right way to engineer green software systems using development process that ensures that the positive and negative effects of the software is continuously monitored and evaluated in order to optimize the software over its life cycle to be more green (environmental friendly).

Colmant et al. [22] presented researches on to improve the software-energy efficiency on multi-core systems. Colmant et al. [28] motivations were driven by the huge impact of the ICT on the world CO2 emissions which represents 2%. Calero et al. [4] highlights some of the meanings of green software notably a software that consumes less energy to run and produces as little waste as possible during its development and operation. Largely, research on green software has focused more on energy consumption and environmentally friendly software systems.

C. Software for Sustainability

There has been some interest in various domains such as manufacturing, energy sector, transportation and for different application in recycling, product packaging, data center setup, gas emissions. Some of the good examples are in grid computing, in Human Computer Interaction (HCI) to change the habit of people.

In [23], authors presented a software system that support sustainable lifestyles with an example of a domestic plant guild to show how sustainable human systems can effectively support a sustainable lifestyle, which can reduce the cost of living as well as the ecological footprint. Penzenstadler et al. [24] highlights vision for systems that will be supporting sustainability in the future (2029) with a set of fictional abstracts around the concepts of sustainability, complexity, collapse, and resilience of ICT systems.

Software can also provide support for sustainability in different domains such as:

- The use of software systems for tracking gas emissions
- Software for climate and disaster prediction
- Smart infrastructural management software
- Enterprise carbon and energy management software
- Smart transportation software to reduce CO^2 emissions.
- Sustainability Knowledge and Learning Management software
- Software for environmental awareness on wildlife and plants

D. Sustainability of the software ecosystem

Today software systems are the pillars of the economy, the software eco system is probably the biggest system in the world we human created. Software eco system has been defined according to Jansen et al. [25] as a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and they operate through the exchange of information, resources and artifacts.

Thus, the sustainability of software ecosystem involves the sustainment of the global system of software systems and services covering aspect of how different sub systems form a huge interconnected system and all the interactions. It covers all different components such as hardware, software and network that is used to resolve complex relationships among companies/organizations in all the different sectors and industries [26].

Sustainability of the software ecosystem entails how can the system of software systems endure with the evolving user requirements and usage overtime with less negative impact on the environments, social, technical and humans. This means the ability of software ecosystem to continue to function and evolve irrespective of any glitch is some part of the ecosystem and should continuously fulfil users' needs.

III. Perceptions of Sustainability in/for Software Systems

In this research, we defined sustainability as a quality construct in the same ways other factors are defined (see, for example, the ISO 25 000 family of standards). In our perception sustainability aims to create balance in the way humans live, produce, and use products and services (resources) with the objective to have less negative impact on the environment and promote the wellbeing of all living species. This means the capacity of software systems to endure in certain ecosystems under current and future conditions while satisfying the needs of users today and tomorrow with minimum negative impact on the environment; at the same time supporting business growth and societal values.

Currently, the dimensions of software sustainability are known and classified into five: economic, environment, social, individual and technical [27] but there is currently no clear categorisation for the perceptions of sustainability in/for software engineering. This section explains the categorization of software sustainability perceptions based on the literature review from the background section. Software sustainability evolution today can be perceive from one of the following perception (see Figure 1); Sustainability in Software Development, Software for Sustainability, Green Software Systems, Sustainability of Software Ecosystems.

- Sustainability in software development (Development): this refers to the processes involve in the development of software (software development life cycle).

- Software for sustainability (Usage): how software are used to support sustainability, an example is a software in fridge to minimize energy wastage (embedded software).
- Green software systems (Focused impact): software systems that uses less energy resource and promotes policies that supports green awareness.
- Sustainability of software ecosystems (Net effect): This is the total impact of the entire software ecosystem (systems of system)

The advancement of software sustainability from the four perceptions has received different level of research attention and contributions. Sustainability in software development, Green Software system we observed has the most important advancement in research compared to software for sustainability and sustainability of software ecosystem that were not fully explored.

Figure 1 portrays the categorization of software sustainability perceptions.

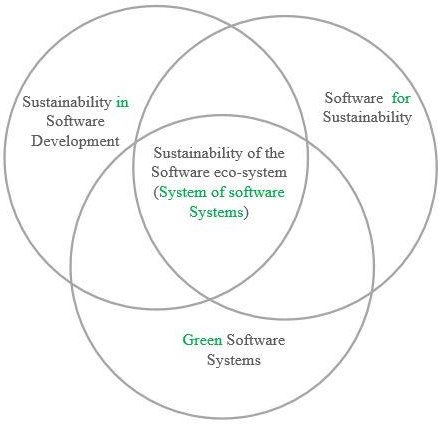


Figure 1. Sustainability Perceptions in/for Software Engineering

IV. Measures and Measurement of Green and Sustainability in/for Software Systems

This section presents different research work relating to green and sustainable software system measures and measurement. According to Britannica [28], measurement is the science of assigning of a quantity, either quantitative or qualitative, to a characteristic of an object or event, while making it comparable to other objects or events. Here object is the software and event is the development process. Sustainability measurement is still a new idea [29] [30] [31] [32]. Indeed, Lami et al. [31] stated that there are few studies about 'what' aspects of sustainability to measure and 'how' to do it. Calero et al. [33] highlighted that nowadays, sustainability is a key factor that should be considered in the software quality models, though there has less research channelled towards sustainability measurement. Seacord et al. [29] stated that planning and management of software sustainment is impaired by a lack of consistently applied, practical measures, and there is no central theoretical framework on measurement of software sustainability.

One of the most referenced sustainability measurement model for software system is the GREENSOFT Model [7]. It is a conceptual reference model for "Green and Sustainable Software", which has the objective to support software developers, administrators, and software users in creating, maintaining, and using software in a more sustainable way [34]. Another software sustainability measurement approach is the Sustainable Business Goal Question Metric (S-BGQM) [35]. It encourages the incorporation and measurement of sustainability during the entire software system development processes. Kramer [36] also wrote about sustainability measurement by proposing some set of questions that should be addressed by any sustainability framework.

A study for monitoring software energy hotspot proposed power model for software energy cost formula as Esoftware = Ecomp +Ecom +Einfra, where Ecomp is the computational cost (i.e., CPU process- ing, memory access, I/O operations), Ecom is the cost of exchanging data over the network, and Einfra is the addi- tional cost incurred by the OS and runtime platform (e.g., Java VM) [37]. The study focused on energy consumption of CPU and network demanding software at different levels of granularity. Also, the formula proposed for software energy efficiency (EF) is UsefulWorkDone/UserdEnergy [38]. This generic measure provide a way for evaluating the energy consumption of different software parts and modules using white box testing to measure which parts are consuming more energy and to see which parts can be optimized for efficient energy usage.

In order to facilitate research on energy usage attribution, software energy footprint lab was setup to provide insight on energy footprint measurements with results interpreting hardware dissipation profiles for various servers under different kinds of software stress [39]. This shows the relations between different hardware resource and the amount of resource required by the running software in relation to the power consumption.

Furthermore, a support tool is presented to analyze legacy systems in order to estimate the energy consumption and detect parts of the system with higher energy consumption. Using the profiling technique, the tool instrument legacy Java systems in order to keep track of its execution. This information, together with the energy consumption, enables the engineer to analyze legacy system consumption detecting energy peaks in the system [40].

Additionally, a modular Energy-Aware Computing Framework (EACOF) is proposed as a way to allow access to energy consumption information of software through API calls. The EACOF is separated into two task for collection and utilization of dynamic energy consumption data which reduce development and maintenance overhead required for the successful completion of each task[41]. Another approach is also proposed for monitoring power consumption of software in order to assist software designers and developer to reduce software power consumption and have better energy efficiency [42]. This approach currently monitors power consumption at source code level, this approach will provide better insights on software energy consumption if extended to the hardware running the software.

As summarized in Table 1 and the research work detailed in [43] [44] [45] and [46], other measures of green and sustainable software have been on software and hardware

energy consumption with less research for measures covering software sustainability dimensions such as individual, social, economic and software sustainability perceptions (Software for sustainability and Software ecosystem).

The measures detailed in Table 1 are structure based on categorization of software sustainability and green measures for software sustainability dimensions and the four sustainability perceptions. Each column after the main title has a "YES or No" to indicate if the proposed measure in the research paper cover any of the categories listed in Table 1. Most of the measures descriptions does not explicitly indicate that the authors considered sustainability dimensions. Base on the descriptions and explanations of the authors for all measures, we have categorized those measures according to the right sustainability dimension (Economic, Social, Individual, Technical and Environment) to show how it relates to the four sustainability perceptions.

TABLE I. MEASURES FOR GREEN AND SUSTAINABLE SOFTWARE LINKED TO SUSTAINABILITY DIMENSIONS AND PERCEPTIONS

Name	**Definition**	**Formula**	**Software Development Lifecycle**	**Green Software**	**Software for sustainability**	**Software ecosystem**	**Sustainability Dimensions**
[37] Software energy cost	The computational cost of performing task involving CPU processing, memory access, I/O operations and exchanging data over the network.	Esoftware = Ecomp +Ecom +Einfra where Ecomp is the computational cost (i.e., CPU process- ing, memory access, I/O operations), Ecom is the cost of exchanging data over the network, and Einfra is the addi- tional cost incurred by the OS and runtime platform (e.g., Java VM)	Yes	Yes	No	No	Environment
[39] Software energy footprint	Not stated	Experimental lab setup details can be found in [39]	No	Yes	No	No	Environment
Energy Efficiency (EF) [47]	Not stated	Energy Efficiency = UsefulWorkDone /UsedEnergy	No	Yes	No	No	Environment, Technical
Performance Efficiency (PE) [48]	Not stated, sub-characteristics measure listed as Time behavior, Resource utilization, capacity		Yes	Yes	No	No	Environment
Power Usage Effectiveness (PUE) [49]	The ratio of facilities energy (supply side) to IT equipment energy (demand size)	PUE= Total Facility Energy/IT equipment Energy	No	Yes	No	No	Environment, Technical
Performance [50]	Not stated	Not available	No	Yes	No	No	Environment, Technical
Efficiency [50]	Not stated, third level indicators provided as: Time Behaviour, Resource Utilization	Not available	Yes	Yes	No	No	Environment, Technical
Resource usage [50]	Not stated, third level indicators provided as: CPU Usage, I/O Usage, Memory Usage, Storage Usage	Not available	Yes	Yes	No	No	Technical
Energy impact [50]	Not stated, third level indicators provided as: Energy Consumption, CO2 Emission, Green Energy Usage	Not available	Yes	Yes	No	No	Environment
Energy efficiency (Speedup Greenup, Powerup, and) [51]	Speedup is defined as the ratio of serial code runtime over parallel code runtime. Greenup is the ratio of the total energy consumption of the non-optimized code (Eφ) over the total energy consumption of the optimized code (Eo). Powerup implies the power effects of an optimization. A less than	Speedup=Tφ/To where Tφ is the total execution time of non-optimized code, and To is the total execution time of the optimized code. Greenup = Eφ/Eo Assuming, Pφ is the average power consumed by the non-optimized code and Po is the average power consumed by the	No	Yes	No	No	Environment, Technical

	1 Powerup implies power savings while a greater than 1 Powerup indicates that the optimized code consumes more power in average.	optimized code Powerup =Po /Pφ= Speedup /Greenup					
Software Project's Footprint [30]	Natural resources and environ- mental impact used during software development.	Transportation from/to the office, and Long-haul trips. Example used in the article: Work-From-Home Days: 2 days out of 165 total team- days (33 project days * 5 team members)=1.21% Long-Haul Roundtrips: By airplane: 6; By train: 0.	Yes	No	No	No	Environment
Functional Suitability (FS) [48]	Functional Completeness, Functional correctness, Functional appropriateness	Not available	Yes	Yes	No	No	Technical
Compatibility [48]	Not stated, sub-characteristics measure listed as Co-existence, Interoperability	Not available	Yes	Yes	No	No	Technical
Usability [48]	Not stated, sub-characteristics measure listed as Appropriateness recognizability, Learnability, Operability, User error protection, User interface eesthetics	Not available	Yes	Yes	No	No	Technical, Individual
Reliability [48]	Not stated, sub-characteristics measure listed as Maturity, Availability, Fault tolerance, Recoverability	Not available	Yes	Yes	No	No	Technical
Portability [48]	Not stated, sub-characteristics measure listed as Adaptability. Installability, Replaceability	Not available	Yes	Yes	No	No	Technical

V. Discussion

Table 1 provides details of measures attributed to green and sustainable software. From Table 1, it can be identified that most measures focused on energy efficiency or power consumptions. With most focus on green software, there is a limitation on having a holistic approach towards software sustainability measurement. The measures of software sustainability should consider the following:

- Human (End users) system interaction: involves the measures of the system sustainability based on how it impacts on users and their level of awareness about sustainability and green. It entails the well-being of the software users' community and the changing of the human mindset.
- Software system developers: evaluate the sustainability of the processes and practices for the development and integration of sustainability in software systems.

One of the key question/concern that should be clearly answered by a sustainability measurement framework is the difference between the different scales of software measurement and the interpretation of these scales of measurement for sustainability. The problem of software sustainability measurement is not only in measuring but rather giving meaningful interpretation of what the measurement means. For example today, fridges are categorized using A+, A++ and A+++ for quantifying and measuring its energy efficiency. Normally A+ consumes less energy, A++ has better energy efficiency than A+ and A++ has the best energy efficiency in today market. According to the EU Directive 92/75/EC which established an energy consumption labelling scheme [52], there are different descriptions of the measures that quantify why Fridge is labelled A+, A++ or A+++ based on its energy consumption. In the same line, there is need for a foundation or framework to ground the different measures for software sustainability measures and measurement with clear interpretation.

Currently, there is not enough firm scientific basis for important choices on how sustainability related factors should be defined and measured, the varying purposes for which the measures are used. This makes it difficult to effectively and efficiently evaluate software sustainability using the right measures.

VI. Conclusion

In this position paper, we summarized the research results on the categorization of software sustainability perceptions. Using the identified four perceptions of software sustainability, we referenced the current measures to each of

the four perceptions. The major focus of all identified green and sustainable software measures are on green software. Energy efficiency has received the most attention. Research work is needed to identify and assess the validity of other measures related to the other perceptions. Research on measures of sustainability has to be grounded in the tradition and theory of software measurement. This requires considering software sustainability as a quality attribute and define it in the same way other attributes are defined.

REFERENCES

[1] United Nations, *World Economic and Social Survey 2013*. 2013.

[2] G. saval Martin, mahaux, patrick heymans, "Requirements Engineering: Foundation for Software Quality," *Requir. Eng. Found. Softw. Qual.*, vol. 4542, no. January, pp. 247–261, 2007.

[3] Ericsson, "Technology for Good," *Available online: https://www.ericsson.com/assets/local/about-ericsson/sustainability-and-corporate-responsibility/documents/2015-corporate-responsibility-and-sustainability-report.pdf Accessed on 30-11-2017*, 2014.

[4] C. Calero and M. Piattini, "Introduction to Green in software engineering," *Green Softw. Eng.*, pp. 1–327, 2015.

[5] N. Condori-Fernandez, G. Procaccianti, and N. Ali, "Metrics for green and sustainable software: MeGSuS 2014," in *Proceedings - 2014 Joint Conference of the International Workshop on Software Measurement, IWSM 2014 and the International Conference on Software Process and Product Measurement, Mensura 2014*, 2014, pp. 62–63.

[6] B. Penzenstadler and A. Fleischmann, "Teach Sustainability in Software Engineering?," in *24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*, 2011.

[7] S. Naumann, M. Dick, E. Kern, and T. Johann, "The GREENSOFT Model: A reference model for green and sustainable software and its engineering," *Sustain. Comput. Informatics Syst.*, vol. 1, no. 4, pp. 294–304, 2011.

[8] A. Raturi, B. Penzenstadler, B. Tomlinson, and D. Richardson, "Developing a sustainability non-functional requirements framework," *Proc. 3rd Int. Work. Green Sustain. Softw. - GREENS 2014*, pp. 1–8, 2014.

[9] C. C. Venters *et al.*, "The Blind Men and the Elephant Towards an Empirical Evaluation Framework for Software Sustainability," vol. 2, no. 1, pp. 1–6, 2014.

[10] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, "Safety, security, now sustainability: The nonfunctional requirement for the 21st century," *IEEE Softw.*, vol. 31, no. 3, pp. 40–47, 2014.

[11] B. Penzenstadler, "Supporting Sustainability Aspects in Software Engineering," *3rd Int. Conf. Comput. Sustain.*, pp. 1–4, 2013.

[12] S. Oyedeji, A. Seffah, and B. Penzenstadler, "A catalogue supporting software sustainability design," *Sustainability*, vol. 10, no. 7, pp. 1–30, 2018.

[13] B. Penzenstadler, "Infusing green: Requirements engineering for green in and through software systems," *3rd Intl. Work. Requir. Eng. Sustain. Syst. 2014*, vol. 1216, no. 1, pp. 44–53, 2014.

[14] K. Roher and D. Richardson, "Sustainability requirement patterns," *2013 3rd Int. Work. Requir. Patterns, RePa 2013 - Proc.*, pp. 8–11, 2013.

[15] S. Murugesan and G. Gangadharan, *Harnessing Green IT : Principles and Practices*, no. September. John Wiley & Sons, Ltd, 2012.

[16] T. Juha, "Good, bad, and beautiful software. In search of green software quality factors," *CEPIS Upgrad. XII 422–27*, no. July, 2011.

[17] L. Erdmann Hilty, L., Goodman, J., Arnfalk, P. and L. Erdmann Hilty, L., Goodman, J., Arnfalk, P., "The Future Impact of ICTs on Environmental Sustainability," *IPTS Publ.*, no. August, p. 68, 2004.

[18] M. N. Malik and H. H. Khan, "Investigating Software Standards: A Lens of Sustainability for Software Crowdsourcing," *IEEE Access*, vol. 6, pp. 5139–5150, 2018.

[19] J. T. Kern Eva, Markus Dick, Naumann Stefan, Guldner Achim, "Green software and green software engineering - definitions, measurements, and quality aspects," *Proc. First Int. Conf. Inf. Commun. Technol. Sustain. ETH Zurich, Febr. 14-16, 2013*, no. January, pp. 175–182, 2013.

[20] K. Erdélyi, "Special factors of development of green software supporting eco sustainability," *SISY 2013 - IEEE 11th Int. Symp. Intell. Syst. Informatics, Proc.*, pp. 337–340, 2013.

[21] M. Dick and S. Naumann, "Enhancing software engineering processes towards sustainable software product design," *24th Int. Conf. Informatics Environ. Prot. (EnviroInfo 2010)*, vol. 2010, pp. 706–715, 2010.

[22] M. Colmant, R. Rouvoy, and L. Seinturier, "Improving the energy efficiency of software systems for multi-core architectures," *Proc. 11th Middlew. Dr. Symp. MDS 2014 - co-located with ACM/IFIP/USENIX 15th Int. Middlew. Conf.*, pp. 2–5, 2014.

[23] J. Norton, A. J. Stringfellow, J. J. L. Jr, B. Penzenstadler, and B. Tomlinson, "Domestic Plant Guilds : A Software System for Sustainability," vol. i, 2013.

[24] B. Penzenstadler *et al.*, "ICT4S 2029 : What will be the Systems Supporting Sustainability in 15 Years ?," 2014.

[25] S. Jansen and M. Cusumano, "Defining software ecosystems: a survey of software platforms and

business network governance : Software Ecosystems Analyzing and Managing Business Networks in the Software Industry," *Softw. Ecosyst. Anal. Manag. Bus. Networks Softw. Ind.*, pp. 13–28, 2013.

[26] J. V. Joshua, D. O. Alao, S. O. Okolie, and O. Awodele, "Software Ecosystem: Features, Benefits and Challenges," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 8, pp. 1–6, 2013.

[27] B. Penzenstadler and H. Femmer, "A generic model for sustainability with process- and product-specific instances," *GIBSE 2013 - Proc. 2013 Work. Green Softw. Eng. Green by Softw. Eng.*, no. June 2015, pp. 3–7, 2013.

[28] E. Britannica, "Measurement instruments and systems. Accessed on 7-12-2017 from: https://www.britannica.com/print/article/371701," pp. 1–2, 2017.

[29] R. Seacord *et al.*, "Measuring Software Sustainability," *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013.

[30] F. Albertao, J. Xiao, C. Tian, Y. Lu, K. Q. Zhang, and C. Liu, "Measuring the Sustainability Performance of Software Projects," *2010 IEEE 7th Int. Conf. E-bus. Eng.*, pp. 369–373, 2010.

[31] G. Lami and L. Buglione, "Measuring software sustainability from a process-centric perspective," *Proc. 2012 Jt. Conf. 22nd Int. Work. Softw. Meas. 2012 7th Int. Conf. Softw. Process Prod. Meas. IWSM-MENSURA 2012*, pp. 53–59, 2012.

[32] M. R. Idio, "Measuring Sustainability Impact of Software," vol. 16, no. 1, pp. 5–7, 2014.

[33] C. Calero, M. F. Bertoa, and M. Angeles Moraga, "A systematic literature review for software sustainability measures," *Green Sustain. Softw. ({GREENS)}, 2013 2nd Int. Work.*, pp. 46–53, 2013.

[34] S. Naumann, M. Dick, E. Kern, and T. Johann, "The GREENSOFT Model: A reference model for green and sustainable software and its engineering," *Sustain. Comput. Informatics Syst.*, vol. 1, no. 4, pp. 294–304, 2011.

[35] S. Oyedeji, A. Seffah, and B. Penzenstadler, "Sustainability Quantification in Requirements Informing Design," *6th Int. Work. Requir. Eng. Sustain. Syst.*, vol. i, 2017.

[36] K.-L. Kramer, *User Experience in the Age of Sustainability*. 2012.

[37] a Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "Runtime monitoring of software energy hotspots," in *Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on*, 2012, pp. 160–169.

[38] T. Johann *et al.*, "How to measure energy-efficiency of software : Metrics and measurement results," no. April 2015, pp. 51–54, 2012.

[39] M. A. Ferreira, E. Hoekstra, B. Merkus, B. Visser, and J. Visser, "Seflab: A lab for measuring software energy footprints," in *2013 2nd International Workshop on Green and Sustainable Software, GREENS 2013 - Proceedings*, 2013, pp. 30–37.

[40] V. Cordero, I. G. R. De Guzmán, and M. Piattini, "A first approach on legacy system energy consumption measurement," in *Proceedings - 2015 IEEE 10th International Conference on Global Software Engineering Workshops, ICGSEW 2015*, 2015, pp. 35–43.

[41] H. Field, G. Anderson, and K. Eder, "EACOF: A Framework for Providing Energy Transparency to enable Energy-Aware Software Development," pp. 1194–1199, 2014.

[42] S. Cagri, C. Furkan, C. James, K. Fouad, P. Lori, and W. Kristina, "Towards Power Reduction Through Improved Software Design," pp. 1–8, 2007.

[43] P. Bozzelli, Q. Gu, and P. Lago, "A systematic literature review on green software metrics," *Sis.Uta.Fi*, 2013.

[44] E. Kern, M. Dick, S. Naumann, A. Guldner, and T. Johann, "Green Software and Green Software Engineering – Definitions , Measurements , and Quality Aspects," pp. 87–94, 2013.

[45] T. Debbarma and K. Chandrasekaran, "Green measurement metrics towards a sustainable software: A systematic literature review," *2016 Int. Conf. Recent Adv. Innov. Eng. ICRAIE 2016*, 2017.

[46] R. V. O. Connor and A. D. Eds, *Software Process Improvement and Capability Determination*, vol. 155, no. June. 2011.

[47] T. Johann, M. Dick, S. Naumann, and E. Kern, "How to measure energy-efficiency of software: Metrics and measurement results," *2012 1st Int. Work. Green Sustain. Software, GREENS 2012 - Proc.*, pp. 51–54, 2012.

[48] G. Oleksandr, K. Vyacheslav, and F. Mario, "Software Quality Standards and Models Evolution: Greenness and Reliability Issues," vol. 469, pp. 277–299, 2016.

[49] E. Rondeau, F. Lepage, J. Georges, E. Rondeau, F. Lepage, and J. Georges, "Measurements and Sustainability," 2015.

[50] S. A. . Koçak, G. I. . Alptekin, and A. B. . Bener, "Evaluation of software product quality attributes and environmental attributes using ANP decision framework," *CEUR Workshop Proc.*, vol. 1216, pp. 37–44, 2014.

[51] S. Abdulsalam, Z. Zong, Q. Gu, and M. Qiu, "Using the Greenup, Powerup, and Speedup metrics to evaluate software energy efficiency," *2015 6th Int. Green Sustain. Comput. Conf.*, 2016.

[52] A. Michel, S. Attali, and E. Bush, "Energy efficiency of White Goods in Europe : monitoring the market with sales data," no. June, pp. 1–53, 2015.

Publication IV

S. Oyedeji, A. Seffah, and B. Penzenstadler
Karlskrona Manifesto: Software requirement engineering good practices

In: *6th International Workshop on Requirement Engineering for Sustainable System co-located with the 25th International Conference on Requirements Engineering*

Reprinted with permission from
CEUR Workshop Proceedings
Vol. 2223, pp. 15-23, 2018

Karlskrona Manifesto: Software Requirement Engineering Good Practices

Shola Oyedeji
School of Engineering Science (LENS)
Lappeenranta University of Technology (LUT)
Lappeenranta, Finland
shola.oyedeji@lut.fi

Birgit Penzenstadler
Department of Computer Engineering and Computer Science
California State University Long Beach (CSULB)
Long Beach, California, USA
birgit.penzenstadler@csulb.edu

***Abstract*—Manifestos in the history of computer science and software engineering have framed guiding principles upon which processes, methods and tools were developed. The Karlskrona Manifesto for Sustainability Design serves this same purpose as a guide for designing and developing sustainable software systems. The goal of this paper is to explore the derivation of good practices by applying the Karlskrona principles in sustainability requirements elicitation. How can the Karlskrona manifesto be translated into methods, processes and tools in the software requirements engineering domain? The result is a proposed list of best practices for software sustainability requirements elicitation. This will facilitate the application of the Karlskrona manifesto for sustainability requirements elicitation and engineering.**

***Index Terms*—Karlskrona Manifesto, requirements engineering, sustainable software, sustainability, best practice, best practice documentation, software requirements, sustainability requirement elicitation, sustainability design**

I. Introduction

Sustainability has become one of the major issues of society today because of the impact of human activity on our planet – this includes interactions in between individual persons, within communities, and between companies and users. Bonini et al. [1] report that sustainability is an important element in the program of many companies, but their environmental, social and governance activities are disconnected from their core strategy. The challenge for most companies is that there is little understanding of how sustainability can be understood by software and requirements engineering professionals to facilitate sustainability design as an established part of the software development process and, specifically, the requirements engineering process [2][3][4].

Users these days are willing to pay more for sustainable software products and services because of the increased awareness from different worldwide initiatives. One central initiative and set of guiding goals are the United Nations Sustainable Development Goals (SDGs), which state initiatives to tackle different crucial sustainability problems humanity faces [5]. Nielsen's global online study [6] shows the percentage of consumers willing to pay extra for products and services from companies dedicated to positive environmental and social impact increased from 55% in 2014 to 72% in 2015.

Software is a core of all human activities today and a major facilitator in the way humans produce and use products and services [7]. The way the software is designed, the requirements to ensure sustainability are factors in software design and how software can support sustainability are still areas that are evolving with different challenges on how best to elicit sustainability requirements for software systems [8].

Consequently, requirements engineering has a major role to play in ensuring the sustainability of software in its broadest understanding. The challenge is that, compared to other types of software requirements like usability and security requirements, which have a well-defined systematic structure and principles on how to elicit system requirements [9], there is still less support on how sustainability requirements can be derived systematically.

One known guiding framework for software sustainability design is the Karlskrona Manifesto for Sustainability Design (KMSD). Following in the footsteps of other successful manifestos such as the Agile manifesto [10], the Business Rules manifesto [11] and the Recomputation manifesto [12], the Karlskrona Manifesto proposes principles that aim to serve as a guide – in this case on how to think of sustainability when it comes to software systems design.

Manifestos like the Agile manifesto are one example that has transitioned into processes, methodologies and tools to help practitioners using Agile in software development. Dick et al. [13] showed how the Agile method was used in software engineering processes to develop "greener" software systems supported by Agile software project management. Agile has different frameworks and approaches such as Scrum, Kanban, and Lean. Agile also has some best practices such as test-driven development (TDD), refactoring, continuous integration, and Pair programing [14].

Relating Agile to requirement engineering, Paetsch et al. [15] have studied the similarities and difference between traditional requirements engineering and agile approaches in order to complement agile with some methods from requirements engineering. Up to now, the Karlskrona Manifesto for Sustainability Design has put forward only limited research on transforming these principles into processes, methods and tools that can support software designers and developers during software systems development.

This paper explores a starting point for such a transition of the principles into processes and methods that can educate and encourage software system designers and developers in eliciting software sustainability requirements.

Section 2 covers the background on the Karlskrona Manifesto and best practice documentation. Section 3 presents the research design for the paper. Section 4 sketches the relation between the Karlskrona Manifesto and software development life cycle phases. Section 5 highlights the proposed method for documenting requirements engineering best practices. Section 6 covers the discussion and Section 7 provides concluding thoughts and future work.

II. BACKGROUND

A. The Karlskrona Manifesto

The Karlskrona Manifesto for Sustainability Design (KMSD) was initiated through an initiative to create a common ground and a point of reference for the global community of research and practice in software and sustainability to effectively communicate major issues, goals, values and principles of sustainability for the design and development of software systems [16]. KMSD has its roots in the Third International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy) [17]. The motive for creating the KMSD was as a result of Christoph Becker's contribution [18] on the relationship between the concerns of sustainability and longevity.

The first stakeholders that contributed, drafted and signed the manifesto were a number of researchers from various areas in the field of software engineering with sustainability research interests as described in [19] [20].

The Karlskrona Manifesto was conceived based on the following guidance [16]:

- **Principles** not techniques as a guide for building, developing and improving new/old techniques and tools to support sustainability design.
- Provide a broader **scope** to be all-inclusive and encompassing all aspects of sustainability.
- Bottom up approach to cover all **emerging structure** from contributions of all participants involved in the initiation of the manifesto.
- Discussion through **participation and transparency** to encourage broader engagement of different experts of sustainability and interested participants.
- **Conversation over consensus** to enable dialogue among the community of stakeholders and all interested participants.
- **Minimal and adaptive process** focussed on emergent content and structure.
- **Synchronous collaboration**. Contents of the manifesto were written through synchronous collaboration.
- **Iterative evolution**. A common vision was formulated to guide the incremental evolution of the manifesto.

Table 1 covers all the Karlskrona Manifesto principles and description for each principles.

TABLE 1. KARLSKRONA MANIFESTO

Principles	Description
P1. Sustainability is systemic	Sustainability is never an isolated property. It requires transdisciplinary common ground of sustainability as well as a global picture of sustainability within other properties.
P2 Sustainability has multiple dimensions	All the different dimensions of sustainability has to be included into our analysis if we are to understand the nature of sustainability in any given situation.
P3 Sustainability transcends multiple disciplines	Working in sustainability means working with people from across many disciplines, addressing the challenges from multiple perspectives.
P4. Sustainability is a concern independent of the purpose of the system.	Sustainability has to be considered even if the primary focus of the system under design is not sustainability.
P5. Sustainability applies to both a system and its wider contexts	There are at least two spheres to consider in system design: the sustainability of the system itself and how it affects the sustainability of the wider system of which it will be part of.
P6. System visibility is a necessary precondition and enabler for sustainability design	Strive to make the status of the system and its context visible at different levels of abstraction and perspectives to enable participation and informed responsible choice.
P7. Sustainability requires action on multiple levels	Seek interventions that have the most leverage on a system and consider the opportunity costs: Whenever you are taking action towards sustainability, consider whether this is the most effective way of intervening in comparison to alternative actions (leverage points).
P8. Sustainability requires to meet the needs of future generations without compromising the prosperity of the current generation	Innovation in sustainability can play out as decoupling present and future needs. By moving away from the language of conflict and the trade-off mind-set, we can identify and enact choices that benefit both present and future.
P9. Sustainability requires long-term thinking	Multiple timescales, including longer-term indicators in assessment and decisions should be considered.

The Karlskrona Manifesto as a guide has helped in increasing sustainability awareness amongst those interested in software systems design and development. However, the core challenge is how to exemplify these principles through practical application in software development. Requirements engineering as a starting point in any software development has a cru-

cial role to play in exemplifying the use of the manifesto principles in software systems requirements elicitation and engineering. There have already been research strides on sustainability in requirements engineering stating the need for sustainability requirements in software systems such as the following research in chronological order:

Mahaux et al. [23] present an experience report about projects that treated sustainability as a first class quality requirements. The authors assessed the current techniques used in systematically eliciting, analyzing and documenting sustainability requirements and pointed at the need for a sustainability toolbox to support requirements engineers to better elicit sustainability requirements.

Roher et al. [21] are concerned with the lack of software engineering teams including environmental sustainability during software development proposed the use of sustainability requirements patterns (SRPs).

Penzenstadler et al. [9] support the consideration of sustainability as a nonfunctional requirement like safety and security that are considered as a system quality attribute.

Raturi et al. [22] focused on how to develop sustainability as a non-functional requirement (NFR) using NFR framework informed by sustainability models.

Becker et al. [24] explain the crucial role of requirements not only for software systems but also for how requirements for sustainability can impact the social-economic and natural environment.

Hinai et al. [25] proposed the use of requirements engineering methodology using social values to elicit social sustainability requirements for software systems.

As highlighted by Becker et al. [16], there are different concerns and dimensions of sustainability, software engineers focusing on the concerns of software qualities, business stakeholders looking at how to make profit and keep business afloat. Furthermore, there is the aspect of social wellbeing of people to ensure better living standards. This, at times, makes the global concern of sustainability difficult to elicit and engineer. Also, quoting Becker et al. [16] offering a way forward: "Rather than asking whether it is appropriate to balance these concerns, we should instead be asking *What methods and tools are needed to explore inter-dependencies between these concerns, and to foster more integrated and long-term thinking?*"

Oyedeji et al. [7] support this further, stating that without a standard for software sustainability requirements, it becomes difficult to identify the boundaries of the sustainability of software systems. A standard will lead to a unifying consensus that can foster sustainability quantification in software systems. Software sustainability has also gained attention as a quality attribute in which there is a proposal to extend the ISO/IEC quality model 25010 to address sustainability [26].

Therefore it is important to follow up on the Karlskrona Manifesto principles and propose examples of how these principles can be applied in software systems requirements elicitation and engineering. This could lay the foundation for a standard template that can encourage and educate requirements engineers for software sustainability requirements.

B. Best practice documentation and templates

A "best practice" (BP) is a practice that is not only good but has proven to work well and produce good results and therefore is recommended as a model. According to Schatten et al.[27], a BP is the transfer of knowledge based on years of success, mistakes and failures from experienced developers to novice developers. These BP can be some good and bad decisions (anti-patterns) from concrete projects that are presented as abstracted scenarios. Designing and developing well-structured software is a challenge especially for young and novice developers. With the use of BP, such challenges can be eased for them with knowledge of how best to develop well-designed software systems from proven procedures.

Fricker at al. [28] presented the best requirements techniques that became industrial best practice based on a survey of a large number of industry projects. One of their core findings showed that projects incorporated stakeholder workshops, the study of existing systems, and re-using specifications. Workshops dominated requirements elicitation practice. Only few projects used techniques like observation, ethnography, surveys, or data mining.

Mike Perks [29] from IBM describes best practices for software development projects from development processes, requirements, architecture, design, construction of code, peer reviews, testing, quality and defects management, deployment, system operations and support, project management, and measuring software project success.

In requirements documentation, one best practice is to use a single and consistent template that all development team members should adhere to in requirements gathering and software development [30].

Parker et al. [31] identified the best practices for managing requirements as the following:

- Naming conventions. Defining and maintaining conventions for identifying releases from the approved requirement set through to the baselined release to the emergency fix or patch.
- Baseline requirements. Requirements, like software releases, must be baselined and those baselines must map directly to the releases they produce.
- Well-defined and understood change control process. Once a baseline is created, changes must be controlled, tracked, traced, approved, and reviewed.
- Requirements review. There must be a requirements review process, and it must be enforced
- Expectation of changes. Make sure changes can be made easily, but under strict access control rules (that include having full traceability).
- Version management. Requirement history should be maintained using methods that make it easy for analysts to look back.
- Requirements traceability. Without the ability to trace a requirement from the idea through to its de-

fined implementation, there is no ability to understand the impact of a proposed change.

- Information maintenance. Maintain attributes for dependencies, relationships, owners, stakeholders, users, funder, dates, costs, models, prototypes, diagrams, and governance about the requirement.
- Collaboration. Provide easy access to requirements information and automatically notify stakeholders of any change of status or change of the requirement to foster collaboration.
- Requirements in a single location. Keep requirements in a single location, preferably in a database designed to manage them.

For companies and organizations, BP are a key way for sharing knowledge and improving the quality of their operation processes [32]. Alwazae et al. [32] introduced the use of a best practice document template (BPDT) as a way for creating high quality documentation within organizations.

Learning from outside the software and requirements engineering domain, the United Nations food and agriculture organization (FAO) presented some good criteria for good practice which also considers sustainability [33].

This body of existing work around best practice documentation and templates was used as a foundation to develop the template presented in the paper at hand.

III. Research Design

The first author performed a mapping of the Karlskrona Manifesto principles onto the Software Development Life Cycle (SDLC) phases and the second author reviewed the mapping.

Based on existing literature on best practice templates [28] [31] [32] [33], the first author developed a first version of the best practice template to document how those Karlskrona manifesto principles can be used in software development activities. This template and some example instances were presented and assessed in an expert evaluation with 15 software developers with at least 3 years of experience in industrial software development at a workshop in the Lappeenranta University of Technology. The workshop is a mentoring program to educate young developers interested in software development career.

The feedback from the developers (more straight-forward, more concrete examples) was incorporated and then presented to the experts again for re-evaluation. Table 2 provide background details of the expert evaluators.

TABLE 2. EXPERT EVALUATORS BACKGROUND

Expert	Background	Company Type	Years of Experience
1	Software Tester	Software Development	5
2	Requirement Engineer	Software Development	3
3	Programmer	Software Development	4
4	UI Designer	Software Development	3
5	Business Analyst	Software Development	3
6	Software Developer	Software Development	4
7	Programmer	Software Development	3
8	IT Manager	Software Development	4
9	CEO / Software Developer	Startup Software Development	3
10	ICT Engineer	Telecom	4
11	Programmer	Finance	3
13	Product Tester / Integration Engineer	HR	3
14	Project Manager / UX expert	Software Development	4
15	Not Provided	Not Provided	Not Provided

IV. Karlskrona Manifesto for Sustainability Design and Software Development

This section provides an overview of how the Karlskrona Manifesto principles can be mapped onto software development life cycle phases.

Table 3 shows the exemplary mapping. There may be additional matches where further principles can be applied within a specific SDLC phase but this mapping is sufficiently extensive for exploring the concepts.

TABLE 3. KARLSKRONA MANIFESTO PRINCIPLES IN RELATION TO SDLC PHASES (adopted from [34])

SDLC Phases	Karlskrona Manifesto Principles
Phase 1. Project Definition	**P1-** This ensures that the project initiation considers sustainability in the overall project definition from the beginning. **P2-** Software sustainability has different dimensions that have to be taken into account from the beginning for better project management with different stakeholders. **P3-** Software projects usually involve stakeholders from different domains, incorporating their sustainability concerns provides better management of those concerns from multiple perspectives which can help the incorporation of sustainability for the software.
Phase 2. User Requirements Definition	**P2-** Recording and documenting user feedback on their perception of sustainability during requirements elicitation will foster better sustainability analysis during the system analysis and design phase.
Phase 3. System Requirements Definition	**P4-** During elicitation of system requirements to consider sustainability concerns for the system during the requirements definition even when it is not a core part of the user requirements. **P5-** Cross evaluate the consequential impacts of the system sustainability requirements and the environment in which the system will function.
Phase 4. Analysis and Design	**P2-** Applying this principle provides a blueprint for system evaluation from all sustainability dimensions (Economic, environment, social, individual and technical). **P4-** This principle provides a rethink of how to

	conduct analysis of system design with consideration of sustainability in order to facilitate development of sustainable system. **P6-** Application of this principle enables better visual and visible overview of the system from different levels of abstraction. **P8-** This will provide better understanding during analysis to make better choices that will help the potential users of the system in present and in future when the system evolves.
Phase 5. Development	**P2-** This will encourage developers during this phase to consider different sustainability dimensions especially technical, social and individual dimensions **P4-** Encourages the search for better avenues to make the system sustainable from the development perspective (developers) and also the functions of the system to aid longevity.
Phase 6. Integration and Testing	**P2-** Provides integration and test team to have a sustainability template that can be used to test the system for all sustainability dimensions based on the sustainability requirement output from phase 2, 3, and 4. **P4-** Application of this principle will aid consideration of sustainability in this phase even if the primary focus of system is not about sustainability.
Phase 7. Implementation	**P5-** Provides a beforehand reasoning for the development team to consider sustainability of the system, its production environment and when push live for use. **P7-** The use of this principle will aid consideration of seeking the involvement of different stakeholders to make the actualization of the system sustainability possible in the production environment and when pushed live.
Phase 8. Sustainment / Maintenance	**P9-** At this stage, this principle helps to create the conscious awareness so that when the system is in live environment, there will be continuous evaluation to assess the system sustainability and think of ways for optimizing and improving sustainability of the system from the different dimensions.

There has been progress on how to design the maintainability of software during/after development and how the security and usability can be improved over time. One thing lacking is how to consider the external impact of the software on the different dimensions of sustainability and engineering those considerations into the software. This is why it is important to conduct proper software sustainability requirements elicitation. The sustainability requirements process needed during SDLC is still evolving in terms of finding the most effective way to elicit software sustainability requirements.

After mapping the Karlskrona Manifesto principles to all the SDLC phases, the next section exemplifies the use of the Karlskrona principles during user and system requirements gathering in the first three phases of the SDLC. This will serve as a benchmark for the remaining SDLC phases because requirements are the first part of any system's design, development and improvement.

V. METHOD FOR DOCUMENTING SOFTWARE SUSTAINABILITY REQUIREMENTS BEST PRACTICE

This section covers details of the method for collecting and disseminating best practice for software sustainability requirements elicitation and engineering. Figure 1 shows the process flow of this method.

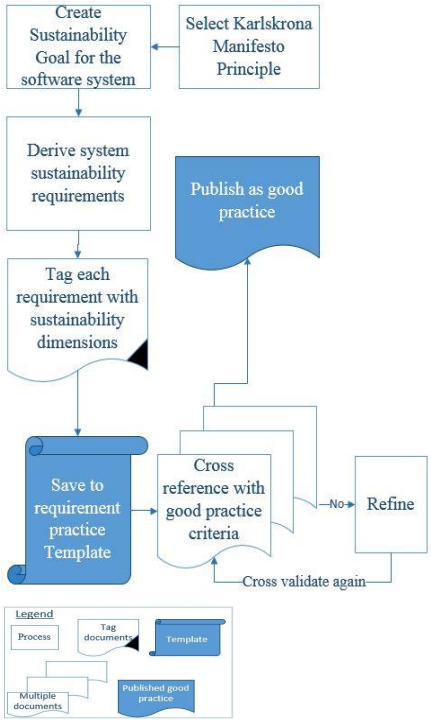


Figure 1. Method for documenting software sustainability requirement elicitation best practice

The proposed method is the first attempt towards exemplifying how the Karlskrona Manifesto principles can serve as a guide for eliciting sustainability requirements for software systems and how such process can be documented as good practice. Such documented good practice can then be reused or followed by software developers and different stakeholders interested in software sustainability.

The first step is to select from the nine Karlskrona principles a principle that relates to the system to be developed or improved. Table 3 shows our mapping of the Karlskrona manifesto to each software development phase.

The second step is to use the selected principle in generating sustainability goals for the system. These goals will serve as a base for creating the system requirements.

The third step involves deriving software sustainability requirements based on all the sustainability goals. These requirements must be measurable and tangible.

The fourth step involves tagging each of the derived sustainability requirements with each sustainability dimension (economic, environment, social, individual and technical).

The fifth step involves using the template that will be proposed in this paper to document the requirements using the good requirement practice template.

The sixth step validates the saved requirement practice using the following criteria [33]:

- Effective and successful: A "good practice" has proven its strategic relevance as the most effective way in achieving a specific objective; it has been successfully adopted and has had a positive impact on individuals and/or communities.
- Environmentally, economically and socially sustainable: A "good practice" meets current needs, in particular the essential needs of the world's poorest, without compromising the ability to address future needs.
- Technically feasible: Technical feasibility is the basis of a "good practice". It is easy to learn and to implement.
- Inherently participatory: Participatory approaches are essential as they support a joint sense of ownership of decisions and actions.
- Replicable and adaptable: A "good practice" should have the potential for replication and should therefore be adaptable to similar objectives in varying situations.
- Reducing disaster/crisis risks, if applicable: A "good practice" contributes to disaster/crisis risks reduction for resilience.

Based on these criteria, the collected requirements are validated, and if all necessary good practice criteria are satisfied the requirements are published as good requirements practice. If there is need for improvement, the requirements are refined again and cross-validated before being published as good requirements practice.

Table 4 provides the best practice template. Table 5 presents an example of the instantiated template for the sustainability best practice - how the Karlskrona Manifesto principles influenced the requirements elicitation process between the requirements engineer, the end user, the programmer, and the business analyst.

The field 'requirements' uses sample requirements from the illustrative case study of a web application for online hospitality service to rent homes for short stays.

Table 4. DESCRIPTION OF TEMPLATE FOR SOFTWARE SUSTAINABILITY REQUIREMENT ELICITATION BEST PRACTICE

Element	**Description**
Title	Which title best describes the best practice?
Date	What month and year is the "good practice" published or documented?
Authors	Who wrote the good practice document?
Target Audience	Who is the target group? To whom is this document useful?
Objective	What is the goal or aim of the best practice?
Location	What is the geographic location in which this practice can be applied for software system (country, region, town or village)? Examples: system for a country's, state, province health care system or banking system or a commercial software application
Stakeholders	Beneficiaries of this best practice? Who are the users, institutions and implementing agencies of the best practice?
Methodology	What methodology was used in documenting the best practice? What were the process steps involved?
Selected Karlskrona manifesto principles	What are the principles that served as guide for creating the best practice for requirement elicitation?
Requirements	What were the requirements used in the best practice? How was sustainability considered in the requirement?
Validation	How was the best practice validated? Did the best practice fulfil the best practice criteria?
Impact	What there an impact in the application of the best practice?
Lessons Learnt	What are the key take away from the application the best practice?
Sustainability	What are the dimensions of sustainability covered in the best practice application?
Contact Details	What is contact details of those responsible for the best practice?

Table 5. TEMPLATE OF SOFTWARE SUSTAINABILITY REQUIREMENT ELICITATION BEST PRACTICE

Element	**Description**
Title	Sustainability user awareness best practice of online hospitality service for short term house renting and sharing
Date	11-06-2018
Authors	Shola Oyedeji, Birgit Penzenstadler
Target Audience	Requirement engineers, Web developers, Business analyst
Objective	Document best practice in requirement elicitation for a web system in order to: • Create awareness among web application developers on how to elicit sustainability requirements • Encourage development of web systems with consciousness of sustainability for end users while using the web application
Location	Applicable worldwide for any web system
Stakeholders	Software requirement engineer, Programmers and Business analyst
Methodology	• Discussion among software development team on what sustainability means to them by going through the Karlskrona manifesto principles • Use the Karlskrona manifesto principles as guide during requirement elicitation during discussion with the end user with aid of the sustainability analysis chat • Record all the requirements in the user requirement specification (URS) and software requirements specification (SRS) • Dialogue about which requirements can better influence end user awareness about sustainability in the user and software requirements specification (URS and SRS) document. • Selected identified requirements • Discussion between with the requirement engineer, end user and programmers about these sustainability requirements to see if implementation is possible or if there is need for modification • Modify requirement in URS and SRS with a set of new requirements targeted towards sustainability based on discussion between the requirement engineer, end user and programmer
Selected Karlskrona manifesto principles	Principle 2: Sustainability has multiple dimensions Principle 6: System visibility is a necessary precondition and enabler for sustainability design Principle 7: Sustainability requires action on multiple levels
Requirements	**Functional Requirement** REQ 1 –Registration (user must be able to register using web form and receive a notification via email) • Sustainability requirement added to this general registration requirement is to include short sustainability tips/links in the registration notification email such as how to recycle common grocery items, use home energy, water, heater and nearest cycling station for getting bicycle commuting REQ 2- UI Search Results (Display search results for all homes with prices and availability to users) • The requirement for sustainability added to this search requirement is to include the CO2 emission for all homes based on the user (searcher location) to the search home (destination) and also add green level label for all homes based on user feedback on how easy to recycle, access to path way for walking or bicycle or public transportation and energy usage during their stay in a home **Non-Functional Requirements** REQ 3 – Performance (ensure good response time) • The sustainability consideration for this requirement is write good compact design codes during development that can determine the exact CPU usage for specific components of the web application and optimize them for less CPU usage • Create effective and efficient algorithm for data structures to help use minimum system resource which can in turn improve respond time and reduce application energy usage
Validation	Programmer, Business analyst and requirement engineer cross validate those requirements with the best practice criteria
Impact	Promote sustainability awareness among software developers and end users Provide opportunity to rethink how software requirement are elicited with consideration of sustainability
Lessons Learnt	1. Software developers don't like too much documentation, so this template has been simplified 2. Requirement engineers appreciated the mapping of Karlskrona manifesto with software development phases 3. Software developers said they would appreciate more documentation on software sustainability for agile development process though they find the mapping in Table 3 useful for them to understand how each of the Karlskrona manifesto relates to each of the software development phases 4. Developers started discussing about coming to office by bicycle or public bus transport instead of their car to reduce CO2 emission
Sustainability	The requirements in this template covers: Social Sustainability Environment Sustainability Individual Sustainability
Contact Details	Shola.oyedeji@lut.fi , birgit.penzenstadler@csulb.edu

VI. DISCUSSION

The systematic mapping of the Karlskrona Manifesto aids requirements engineers and software developers in understanding how the Karlskrona Manifesto for software sustainability design relates to the software development life cycle (see Table 3). The template (see Tables 4 and 5) provides a typical example of how best practices for software sustainability requirements can be documented. Table 4 provides details of what is expected in the template and Table 5 shows the template usage for documenting both functional and non-functional requirements. This best practice uses the example of an online hospitality web application.

In addition, with the work presented in this paper, we partially respond to research challenges identified by Chitchyan et al. [2] from the state of practice for software sustainability design in requirement engineering. They noted the following:

There is a lack of methodological support for sustainability design in requirement engineering because it is not part of most companies practice [2]. The method presented in this paper serves as support for helping requirements engineers, software developers and all stakeholders in documenting best practices from sustainability design in requirements engineering using a structured methodology.

They also noted a need for a mentality change to make people transition from their old ways of eliciting requirements and developing software to new way of sustainability design in requirements engineering. Documenting best practices using the proposed template presented here educates and promotes awareness among those involved in the requirements engineering process of software development. This can be one way of persuading them to see benefits of eliciting software requirements and developing software system in a new way with support for sustainability design.

Overall, the mapping of the Karlskrona Manifesto principles in Table 3, the method (see Figure 1), and the template for documenting (see Table 4) provide guidance to support requirements engineers and software developers in software sustainability requirements elicitation and in documenting best practices from the requirements process.

In our opinion, instantiating the Karlskrona Manifesto for sustainability design for software processes, practices and methods will go a long way to create awareness about software sustainability and increase broader engagement for different stakeholders within academia and industry.

The following are some of the limitations of our work:

- The mapping of the Karlskrona Manifesto principles to software development process phases in this current version may be incomplete as of now and require a further iteration of the mapping process (meaning: there could be principles that are not listed for a specific phase despite being applicable), but the mapping is sufficiently complete to provide solid grounds for discussion.
- The template may be too restrictive and not capture all relevant information potentially provided by those documenting the best practice. However, if templates get too lengthy, which can easily occur when trying to accommodate all possibilities, they are less likely to be picked up by practitioners (see next point).
- If structure and guidance become too detailed, engineers may refuse to use them, find them too specific to apply, or apply the principles without putting sufficient critical thought into it. Consequently, that is why the template for documenting best practices has been simplified for straightforward and self-explanatory documentation.

VII. CONCLUSION

This paper presents a mapping of the application of the Karlskrona Manifesto principles to software development activities and a template for documenting their usage in best practices, supported by an example instance of its usage. An expert group evaluated this template in two iterations.

The proposed approach can be used as guide by requirements engineers during software requirements elicitation and documenting software sustainability requirements best practices. Furthermore, software developers can also benefit from using it for rethinking how they develop software using the mapped Karlskrona Manifesto principles as guide during each stage of the software development life cycle.

Future work includes the application of the proposed methodology in industrial case studies and using the template to document best practices from those case studies. Specifically, during the evaluation, the expert group requested a mapping of the Karlskrona Manifesto to agile software development method, especially to Scrum. Consequently, we plan this mapping and adaptation for the first industry case study.

REFERENCES

[1] S. Bonini and S. Görner, "The business of sustainability : Putting it into practice," *Insights Publ.*, p. 6, 2011.

[2] R. Chitchyan, L. Duboc, C. Becker, S. Betz, B. Penzenstadler, and C. C. Venters, "Sustainability Design in Requirements Engineering : State of Practice," pp. 533–542, 2016.

[3] M. Mahaux and C. Canon, "Integrating the Complexity of Sustainability in Requirements Engineering," *First Int. Work. Requir. Eng. Sustain. Syst.*, 2012.

[4] U. K. Jannat, "Green Software Engineering Adaption In Requirement Elicitation Process," vol. 5, no. 08, pp. 94–98, 2016.

[5] United Nations, "Sustainable Development Goals Available at: http://www.undp.org/content/undp/en/home/sustainable-development-goals.html . Accessed on 25-04-2018," no. September 2000, pp. 8–23, 2015.

[6] Nielsen, "Nielsen global online study. Available online at: http://www.nielsen.com/eu/en/insights/news/2015/green-generation-millennials-say-sustainability-is-a-shopping-priority.html Accessed on 3-03-2018," *Web Rep.*, 2015.

[7] S. Oyedeji, A. Seffah, and B. Penzenstadler, "Sustainability Quantification in Requirements Informing Design," *6th Int. Work. Requir. Eng. Sustain. Syst.*, vol. i, 2017.

[8] G. saval Martin, mahaux, patrick heymans, “Requirements Engineering: Foundation for Software Quality,” *Requir. Eng. Found. Softw. Qual.*, vol. 4542, no. January, pp. 247–261, 2007.

[9] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, “Safety, security, now sustainability: The nonfunctional requirement for the 21st century,” *IEEE Softw.*, vol. 31, no. 3, pp. 40–47, 2014.

[10] M. Fowler and J. Highsmith, “The agile manifesto,” *Softw. Dev.*, vol. 9, no. August, pp. 28–35, 2001.

[11] B. R. Group, “The Business Rules Manifesto,” *Bus. Rules Group. Version Available online http//www.businessrulesgroup.org/brmanifesto.php Accessed 12-11-2017*, no. c, pp. 1–2, 2003.

[12] I. Gent, “THE RECOMPUTATION MANIFESTO,” *Available online: https://www.software.ac.uk/blog/2016-10-05-recomputation-manifesto Accessed on 12-11-2017*, p. 9479.

[13] M. Dick, J. Drangmeister, E. Kern, and S. Naumann, “P66-Green software engineering with agile methods,” *2013 2nd Int. Work. Green Sustain. Software, GREENS 2013 - Proc.*, pp. 78–85, 2013.

[14] A. R. & D, “Agile Project Management: Best Practices and Methodologies,” *Altexsoft*, 2015.

[15] F. Paetsch and F. Maurer, “Requirements Engineering and Agile Software Development,” pp. 1–6, 2003.

[16] C. Becker *et al.*, “Sustainability Design and Software: The Karlskrona Manifesto,” *Proc. - Int. Conf. Softw. Eng.*, vol. 2, pp. 467–476, 2015.

[17] B. Penzenstadler, M. Martin, and S. Camille, “RE4SuSy: Requirements engineering for Sustainable systems,” *CEUR Work. Proceedings, Retrieved from Http//ceur-ws.org/Vol-1216/*, vol. 995, 2013.

[18] B. Christoph, “Sustainability and longevity: Two sides of the same quality?,” *CEUR Workshop Proc.*, vol. 1216, pp. 1–6, 2014.

[19] C. Becker *et al.*, “Website for The Karlskrona manifesto for sustainability design,” *arXiv1410.6968 [cs] Available online Http//sustainabilitydesign.org/karlskrona-manifesto/ Accessed 10-10-2017*, vol. 20, no. May, p. 2014, 2014.

[20] C. Becker *et al.*, “The Karlskrona manifesto for sustainability design,” *arXiv1410.6968 [cs]*, vol. 20, no. May, p. 2014, 2014.

[21] K. Roher and D. Richardson, “Sustainability requirement patterns,” *2013 3rd Int. Work. Requir. Patterns, RePa 2013 - Proc.*, pp. 8–11, 2013.

[22] A. Raturi, B. Penzenstadler, B. Tomlinson, and D. Richardson, “Developing a sustainability non-functional requirements framework,” *Proc. 3rd Int. Work. Green Sustain. Softw. - GREENS 2014*, pp. 1–8, 2014.

[23] G. Saval, M. Mahaux, and P. Heymans, “Discovering Sustainability Requirements: An Experience Report,” *REFSQ*, 2013.

[24] B. Christoph *et al.*, “Requirements: The key to sustainability,” *IEEE Softw.*, vol. 33, no. 1, pp. 56–65, 2016.

[25] M. Al Hinai and R. Chitchyan, “Engineering Requirements for Social Sustainability,” *Proc. ICT Sustain. 2016*, 2016.

[26] G. A. García-mireles, “Exploring Sustainability from the Software Quality Model Perspective,” in *13th Iberian Conference on Information Systems and Technologies (CISTI)*.

[27] A. Schatten, S. Biffl, M. Demolsky, E. Gostischa-Franta, T. Östreicher, and D. Winkler, *Best Practice Software-Engineering: Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. 2010.

[28] S. A. Fricker, R. Grau, and A. Zwingli, “Requirements Engineering : Best Practice Requirements Engineering State-of-Art,” 2015.

[29] M. Perks and IBM, “Best practices for software development projects. Available online : https://www.ibm.com/developerworks/websphere/library/techarticles/0306_perks/perks2.html. Accessed on 18-06-2018,” 2006.

[30] altexsoft, “Software Documentation Types and Best Practices. Available online : https://www.altexsoft.com/blog/business/software-documentation-types-and-best-practices/ Accessed on 18-06-2018,” 2017.

[31] P. Kevin and S. Serena, “Requirements Engineering : Best Practice,” no. July, 2015.

[32] M. Alwazae, E. Perjons, and P. Johannesson, “Applying a Template for Best Practice Documentation,” *Procedia Comput. Sci.*, vol. 72, pp. 252–260, 2015.

[33] FAO, “Good practices template,” *Food Agric. Organ. United Nations*, no. July, pp. 1–5, 2014.

[34] S. Oyedeji, B. Penzenstadler, and A. Seffah, “Proposal for a Software Sustainability Design Catalogue,” no. May, pp. 1–28, 2018.

Publication V

S. Oyedeji, B. Penzenstadler, M. O. Adisa and A. Wolf
Validation Study of a Framework for Sustainable Software System Design and Development

In: *6th International Conference on ICT for Sustainability, ICT4S*

Reprinted with permission from
CEUR Workshop Proceedings
2019

Validation Study of a Framework for Sustainable Software System Design and Development

Shola Oyedeji
LUT School of Engineering Science (LENS)
LUT University
Lappeenranta, Finland
shola.oyedeji@lut.fi

Mikhail .O. Adisa
IT Service Management Consultant
IT Solutions
Abuja, Nigeria
olamikhx@gmail.com

Birgit Penzenstadler
Department of Computer Engineering and Computer Science,
California State University Long Beach
Long Beach, USA
LUT School of Engineering (LENS)
LUT University
Lappeenranta, Finland
birgit.penzenstadler@csulb.edu

Annika Wolf
LUT School of Engineering Science (LENS)
LUT University
Lappeenranta, Finland
annika.Wolff@lut.fi

***Abstract*—Sustainability in software design is an evolving area that requires more practical guidance on how software engineers and businesses could innovate and design software systems that consider sustainability as a guiding principle for supporting a sustainable environment, reducing the negative impact of ICT and at the same time promoting software system design for sustainability. This paper presents our early results for validating a Framework for Sustainability of Software System Design (FSSSD) based on the Software Sustainability Design Catalogue (SSDC). The SSDC exemplifies the use of Karlskrona Manifesto principles for sustainability design and how to promote sustainability design principles for software systems.**

***Index Terms*—Sustainable design, sustainability, software sustainability, information and communication technology, Karlskrona manifesto, Sustainability design principles**

I. Introduction

Sustainability is receiving a wide range of research from different sectors. Currently, there is not enough research results with guidelines and frameworks to support software designers and companies on how to design and develop software with sustainability at the core [1]. One of the main problems for sustainability in software design is that for software designers there are few existing tools that wrap core principles of sustainability together which can support effective software sustainability design and development [2]. For companies, the challenge is that there is little understanding of how sustainability can be understood by software and requirements engineering professionals to facilitate sustainability design as an established part of the software development process within companies [3][4][5].

The sustainable development goals (SDGs) [6] in 2015 got signed by more than 190 world leaders, this shows the importance of sustainability today in all aspects of our lives. Though there is no direct mention of software sustainability in the 17 SDGs, software as a catalyst for all sectors of the economy [7] serves as a key element for the implementation and actualization of those SDGs. According to the 2016 mobile industry impact report [8], the United Nations Sustainable Development Goals provide the opportunity for engagement to address the most pressing global challenges, but they cannot be realized without the business community. The report stresses the need for companies to implement the SDGs, working with governments and the international community to expand connectivity, lower barriers to access, and build a future of dignity and opportunity, where no one is left behind and ensure that tools and applications are developed with vulnerable communities in mind [8].

Sustainable development is also driving software innovations for creating new opportunities of cutting costs, adding value and for gaining competitive advantage [9]. García-Berna et al. [10] points out the practices applied by practitioners in companies for sustainability and the need for standards as a way of seeking more sustainable software businesses. The importance of sustainability as a driving force for companies is further highlighted in these reports: Sustainability Nears a Tipping Point [11]; Ericsson energy and carbon report [12]; Microsoft 2015 Citizenship Report [13]. In summary, software is a core of all human activities today and a major facilitator in the way humans produce and use products and services [14]. The way software is designed and the requirements to ensure sustainability in software design are factors that are challenging for software designers, requirement engineers and companies [15].

The Karlskrona Manifesto for Sustainability Design (KMSD) [16] was initiated as a starting point for tackling this challenges in software engineering. Based on these KMSD

principles and the Software Sustainability Design Catalogue (SSDC) [1], the Framework for Sustainability of Software System Design (FSSSD) was created [1]. This paper presents the first results of applying the Framework for Sustainability of Software System Design (FSSSD) [1].

The next section covers related research work. Section III presents the study design. Section IV covers the first case study and section V details the second case study. Discussion is in section VI and concluding remarks in section VII.

II. Background

Software development practices and processes that are widely used in industry for software design and development lack in addressing sustainability [17]. There is currently no single point of reference for researchers and practitioners where the sustainability measures are gathered and exemplified [26]. The issue of lack of understanding on how to effectively and efficiently integrate the different sustainability dimensions (economic, social, individual, environmental and technical) [18] into software design, development and wider engineering processes [9] [19] has hindered the adoption of sustainability in software development.

There have been different research efforts suggesting the need to further research on how sustainability can be supported in software requirements and design stages for all the different sustainability dimensions [20] [21] [22]. Further research also shows sustainability requires multidimensional and interdisciplinary approach [3][7][23][24][25] in order to fully achieve sustainability in software design, development and measurement.

From the requirements engineering phase, sustainability has been considered as a non-functional requirement [26][27][28], and Roher et al. [29] suggests the use of sustainability requirement patterns (SRPs) as a way to guide software requirements engineers in eliciting sustainability requirements in the requirements engineering process. However, there is a lack of examples to show how these are applied in the industry.

Researchers from the Human Computer Interaction (HCI) community believe sustainable HCI can facilitate and support sustainability in the design and development of new interfaces to promote sustainability awareness [30]. Froehlich et al. [31] show eco feedback can serve as a key way of promoting sustainability awareness among users of software systems. One key example of an eco-feedback application [32] shows a positive result in persuading and changing users habit towards sustainability. Successful application of eco feedback is when information has been tailored to encourage users towards sustainability through user emotional engagement [33] [34].

Some of the design issues in design of sustainability for better user experience of software systems are highlighted by Kem-Laurin [35]. Kem-Laurin propose the use of sustainability user experience framework as a way to guide designers to mitigate these problems. The challenge according to Eli Blevis [36] and Fallman [37] is that sustainability is not yet a core part of HCI. This has hindered the ability of designers to properly evaluate design choices for software systems especially with the different sustainability dimensions.

The challenges covered in this background section motivate the application of FSSSD to two case studies in order to show and suggest how to better support sustainability in software design and development.

III. Study Design for Framework Validation

This section describes the Framework for Sustainability of Software System Design (FSSSD) and the rationale behind choosing the two case studies used in the research.

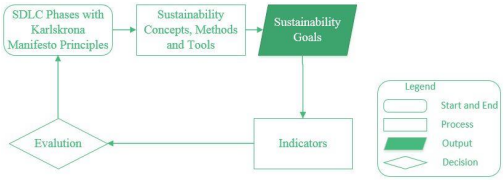


Figure 1. Framework for Sustainability of Software System Design (FSSSD) [1]

The FSSSD (Figure 1) was created to assist developers to incorporate sustainability goals and requirements during software system design and development covering the software development life-cycle (SDLC) phases. For the purpose of better understanding, the FSSSD (Figure 1) is transformed into tabular form (Table 1) [1].

TABLE I. Framework for Sustainability of Software System Design (FSSSD) [1]

SDLC phases and KMSD principles	**Sustainability goals**	**Sustainability concepts, Methods and Tools**	**Indicators**
Phase 1. Project Definition, P1, P2 and P3	Design for sustainable efficiency, reusability	biomimicry, sustainable business canvas	Carbon footprint, material footprint, end of life footprint.
Phase 2. User Requirements Definition, P2	Increase sustainability awareness among users.	Sustainability requirement template	Total number of sustainability requirements, priority assign to sustainability requirements.
Phase 3. System Requirements Definition, P4, and P5	Design for efficiency, sustainability awareness and interoperability.	Cradle to cradle, Goal model.	Total number of system goals relating to sustainability dimensions.
Phase 4. Analysis and Design, P2, P4, P6 and P8	Design for reuse and efficiency, localization, interoperability	Life-cycle sustainability assessment, social return on investment, sustainability analysis radar chart	Number of first-, second- and third-order impacts of system identified.
Phase 5. Development, P2 and P4	Design for reuse, design for module replicability, design for efficiency, sustainability awareness, efficiency, design for easy	Biomimicry, cradle to cradle	Number of coding choices influenced by sustainability, number of features (functions) added to systems to inform users about sustainability through

	service and maintenance		functions like eco feedback.
Phase 6. Integration and Testing, P2 and P4	Design for easy assembly and disassembly, design for durability	Cradle to cradle, sustainability analysis radar chart, life-cycle sustainability assessment	How much information from sustainability analysis chart was used during integration and testing such as the number of systems functions tested against sustainability concerns such as the first-order (immediate) impact, possible second-order (enabling) and potential third order (structural) impacts of the system
Phase 7. Implementation, P5 and P7	Design for easy use, design to induce conscious sustainability awareness, design to educate users about sustainability, design for easy recycle.	Biomimicry, cradle to cradle	The priority assign to sustainability by developers and the system owners/users during after implementation
Phase 8. Sustainment/ Maintenance, P9	Proper design for serviceability, design for easy replacement of code modules, design for continuous user engagement through sustainability awareness.	Life-cycle sustainability assessment, sustainability analysis radar chart, cradle to cradle.	Number of improvements to system based on sustainability requirements either from users' feedback or developers.

The approach applied in the selection of each case study was to choose two different case studies where one case study has the ultimate goal of sustainability from the beginning and the other case study uses the framework to improve an existing system.

The goal is to see what difference will occur from these two different case studies in different application context. The first case study - about a pension benefit tracker application - does not have sustainability as the central core and the second case study - about an energy usage display for university staff and students - is motivated by sustainability.

IV. Case Study One: Pension Benefit Tracker Application

The pension benefit tracker is an application from a pension company in Nigeria that wants to track pension benefit applications submitted by clients from all over the company's branches in different states of Nigeria. Currently, the pension applications are done manually from each branch and those applications are sent via courier service to the head office. This usually causes the following problems:

1. Zonal managers don't have direct access to know the status of applications submitted through them and have to directly place phone calls to the Head office to know the application status.
2. Customer service staff are unable to know why an application is pending, unless they contact the benefit department.
3. Time consumption, as all status updates are through customer service at the head office alone.
4. Files can go missing in transit because application files are handled manually.
5. Double application and too much physical involvement because of follow up in person

The company intended to develop a new pension benefit application tracker application for these key stakeholders, the benefit department, the customer service unit, the zonal managers and the clients with the aim of:

1. Identifying ways of improving the pension benefit application process and enhance communication.
2. Designing and implementing a web-based solution that will ensure effective and efficient benefit processing for users.

The below Figure 2 is the first Use case diagram for the application.

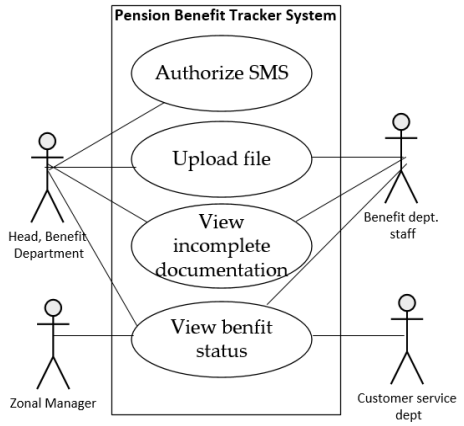


Fig 2. Use Case diagram pension benefit tracker

Figure 2 shows the use case diagram of the system for pension benefit tracker application after initial analysis. Figure 3 presents the process model of the pension benefit application after a second analysis, factoring in all the aforementioned problems without using FSSSD. Figure 3 shows that sustainability was not the core of this case study, based on the process model, as stakeholders are just interested in solving the problems stated in the case study.

Table 2 presents the details for applying FSSSD to the pension benefit tracker application (case study one). The documentation

for this case study using FSSSD covers the project initiation, user requirements and system requirements phases only (see Table 2) because that is the current development stage of the project.

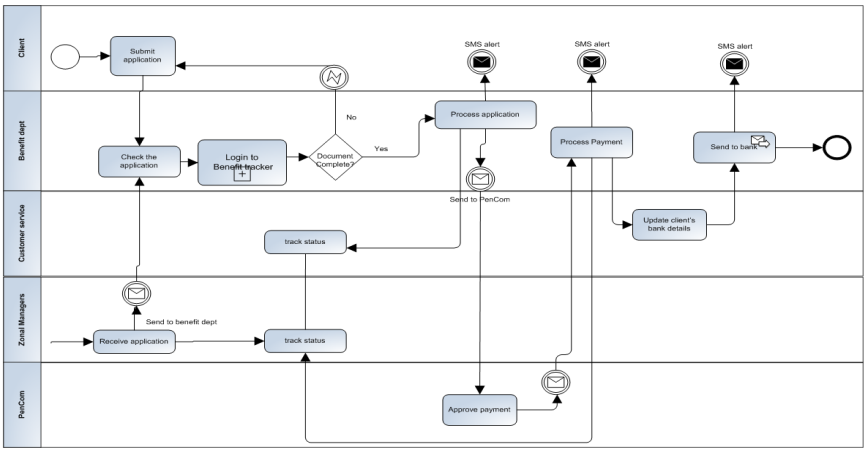


Fig 3. New Process Model for Pension Application after second analysis

TABLE II. APPLICATION OF FSSSD IN CASE STUDY ONE

SDLC Phases and Karlskrona Manifesto Principles	**Sustainability Goals**	**Sustainability Concepts, Methods and Tools**	**Indicators /Measure / Metric**
Phase 1. Project Definition Provide end users with easy to use interface for tracking pension payment, ensure each module for tracking can be updated to include new branches, Provide flexibility such as bulk and single upload, ensure easy integration with other existing pension systems, present report of system usage to track energy consumption in a way to educate users about sustainability, add bug reports	Design for: Easy integration, Reusability, Developers work satisfaction, Maintainability, Energy efficiency	Motivated by the cradle to cradle approach ensuring that the pension tracker application is design and developed in a way that it can be reused for future pension related purposes and easily integrated with other bigger pension system within the company	1. How many state branches can easily integrate the systems with less Backlog Management Index (BMI)? 2. What is the number of reports from IT staff about how to improve system energy efficiency? 3. How satisfied are the developers with the development of the application
Phase 2. User Requirements Definition 1. Provide tracking of pension benefit payment application from request submission to payment 2. Status notification should be sent to users after each stage of the pension benefit application	Reduce development cost, increase efficiency	Sustainability requirement Template	How efficient is benefit department able to track new pension benefit applications and send notification successfully
Phase 3. System Requirements Definition 1. The pension tracker application should be accessible online via web at any branch 2. The application should have ability to enable Managers, pensioners and other stakeholders check application status 3. Provide automatic status communication and notification at each stage of benefit application 4. Allow bulk or single file upload 5. Provide SMS authorization from managers in benefit department 6. Send SMS notification to applicants 7. Send Incomplete documentation notification to	Design for efficiency, sustainability awareness	Social and individual dimension of sustainability	1. How satisfied are users with visual problem with the magnifying display? 2. Do users use the option of email notification and does it reduce company cost for sending SMS? 3. How many positive responses came from users base on the "Save the planet, Reduce environmental waste" tag message?

benefit department staff 8. Provide email notification as an option for all users 9. Provide option of different display to magnify fonts for users with visual problems 10. Provide option to preview pension application and save electronically 11. Add a tag message below each notification "Save the planet, Reduce environmental waste" 12. Provide energy report for system usage			4. How many initiatives were suggested from IT department base on the system energy report?

After application of the FSSSD with the sustainability design catalogue (SSDC), see Table 2, the IT department made some changes to the system requirements such as addition of the following system requirements in Table 2, SDLC phase 3:

1. Email notification option instead of only SMS function as seen in Figure 3 in which only SMS is shown (system requirement 8 in Table 2).
2. Provide option of different display to magnify fonts for users with visual problems especially older staff (system requirement 9 in Table 2).
3. Provide option to preview pension application and save electronically instead of printing and filling locally to reduce cost, paper waste and energy usage (system requirement 10 in Table 2)
4. Add a tag message below each notification "Save the planet, and reduce environmental waste" to raise sustainability awareness among staff and clients (system requirement 11 in Table 2).
5. An energy report that enables developers to improve efficiency (system requirement 12 in Table 2).

V. Case Study Two: Energy Usage and Carbon Emission Display for University Staff and Students

This is a university setting project to raise the awareness of the public (university staff and students) about energy usage and the carbon emissions through activities in the university. The project requires a web application interface which will display the energy usage and carbon emission. The goal is to let the public know more about the electricity consumption of each building in the university and understand the relation between the electricity consumption and carbon emission (CO_2).
Using the FSSSD, the involved students and their supervisors documented the project to show how sustainability was considered in the project (see Table 3). Figure 4 shows the interface design for the project and Figure 5 covers an overview of the sustainability business canvas for the project.

TABLE III. FSSSD application in case study two (Energy usage and carbon emission display for staff and students)

SDLC Phases and Karlskrona Manifesto Principles	**Sustainability Goals**	**Sustainability Concepts, Methods and Tools**	**Indicators /Measure**
Phase 1. Project Definition Raise awareness from the public (university staff and students) about energy usage and the carbon emissions through activities in the university.	Design for sustainability awareness, efficiency, reusability, easy integration, maintainability and energy efficiency	Sustainable Business Canvas was used to breakdown the project goals and scope into environment, society, economy, process, value and people in order to have better clarity on the sustainability goals of the project and derive basic benchmarks for evaluating the project at the end.	1. What is the impact of the project on promoting sustainability awareness within the university? 2. How many users participate in the weekly sustainability challenge? 3. What are the new initiatives from departments towards sustainability based on the application usage?
Phase 2. User Requirements Definition 1. Provide information on energy usage within the university 2. Show the carbon emission 3. Allow weekly sustainability challenge and show winners 4. Section for user community to connect and discuss 5. Provide feature to share things to social media	Increase sustainability awareness through energy usage and carbon emission information to users	Sustainability requirement template (template that shows the sustainability analysis of the five dimensions and the three orders of effects from the design catalogue) [1]	1. Can users see information about energy usage and carbon emission? 2. How effective is the weekly sustainability challenge? 3. How many users participate in the weekly sustainability challenge? 4. Do users share their experience via social media portal?
Phase 3. System Requirements Definition 1. Information about energy usage and carbon emission should be available via the central display screen and web portal	Design for sustainability awareness, maintainability and energy efficiency	Environmental, Social and individual dimension of sustainability	1. Can users understand the energy and carbon emission information presented? 2. How easy can users join the

2. The application should translate the carbon emission data base on energy usage into meaningful information for better user understanding such as distance between Lappeenranta and other cities 3. The web interface should allow users participate in the weekly challenge 4. Users are able to share their weekly challenge results via Facebook and Twitter. 5. The application should allow users form community of interest for different sustainability goals. 6. Provide API to allow for easy integration with other applications			weekly challenge? 3. Does the application to form community of different sustainability goals? 4. Can users successfully share their weekly challenge on Facebook and Twitter? 5. Does the API allow easy information access?
Phase 4. Analysis and Design 1. Identify the first, second and third order impact of the application on user energy usage and sustainability awareness 2. Find areas to improve the application implementation base on the different sustainability dimensions especially environment, social and technical dimensions	Design for sustainability awareness, reuse, efficiency and localization	Sustainability analysis radar chart was used for the sustainability analysis to show the he first, second and third (immediate, enabling, and structural) impacts of the application.	1. What is the potential percentage of energy usage reduction in the university? 2. What is the level of user awareness overtime about energy usage and carbon emission? 3. What is the impact of the user community for users' motivation towards sustainability within the university?
Phase 5. Development	Design for sustainability awareness, efficiency, reuse, design for module replicability, design for easy service and maintenance	Cradle to cradle concept influence the development to develop each module in the application in a way that support evolution as user requirements changes over time and ensuring sustainability is the core of all development	1. What is the defect density of the application? 2. What is the energy efficiency of the application? 3. How many modules relating to sustainability awareness was successfully developed? 4. Can users successfully use the application for all application functions such as join a community, participate and share weekly sustainability results, understand displayed energy usage and carbon emission information?

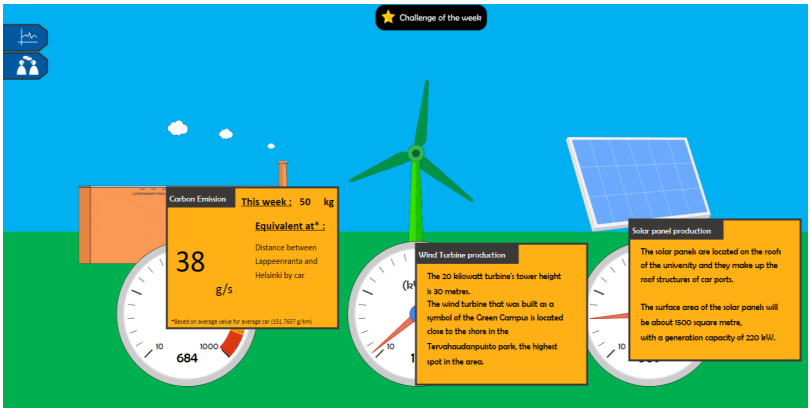


Fig 4.Sustainability awareness via energy usage interface

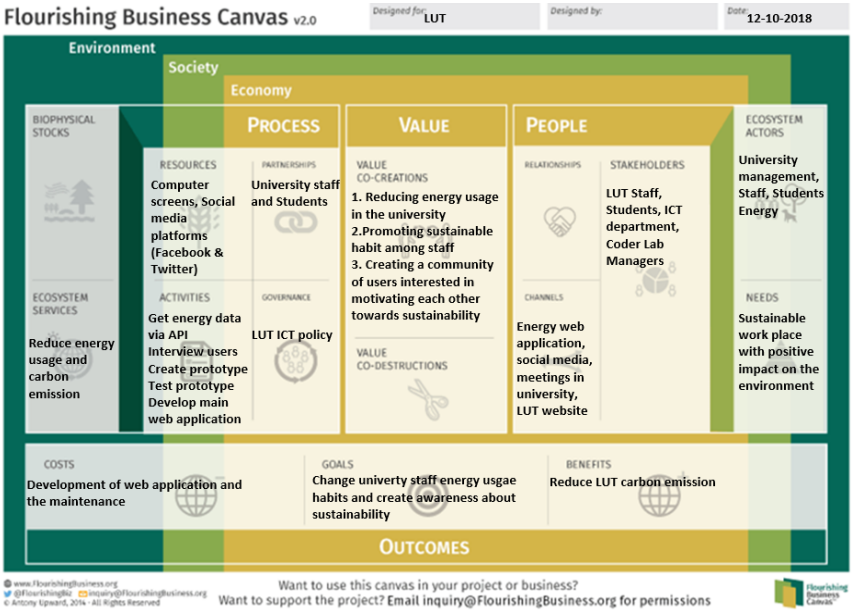


Fig 5. Sustainable Business Canvas for Case Study Two [38]

VI. DISCUSSION

For the project initiation in the first case study, normally project managers will only evaluate projects by considering whether the software system meets all user requirements after development and testing as a yardstick for satisfying all project requirements. The application of FSSSD in case study one (Table 2) shows that indicators used for evaluating the project up to the current development stage included the level of developer satisfaction (individual dimension of sustainability) and the number of IT staff reporting on how to improve the system energy efficiency (environment and technical dimension). This confirms a new perspective towards software project evaluation with sustainability dimensions now considered by stakeholders in case study one. The use of FSSSD also led to new system requirements (Table 2) with the potential to improve the system efficiency and consideration of sustainability based on the system context.

Based on the initial response from stakeholders in case study one, it indicates that as a company their major interest was to check if FSSSD - as guide in the application of sustainability in software system design and development - would save them cost and improve staff productivity. The use of developers satisfaction for the pension benefit tracker is one example because the company believes if there is means of checking staff satisfaction, it could offer a means of improving working conditions which will in turn improve productivity over time. This will help them reduce the cost of operations and improve profit margin.

Case study two provides a different use of FSSSD as sustainability is the core of the application design. As noted in [33] [34], with better tailored information through eco feedback, user habits can change positively towards sustainability over time. The second case study (see Table 3, Figures 4 and 5) shows the presentation of energy usage data converted into carbon emission. With the use of FSSSD as guide, the application in case study two was designed in a way that the carbon emission information was displayed in order to educate users about their energy consumption habits in each department. The system presented the percentage of carbon emission in form of distance between one city to another with the goal to provide better understanding for the public about the impact of their energy consumption on the environment.

Feedback and comments (Table 4) from stakeholders in case study one and two indicates that developers and engineers complained there are few industry case studies for software development that shows how sustainability was applied. The second challenge was in motivating software requirements engineers and designers to incorporate the use of the new sustainability artifacts for sustainability in requirements and software development because most of them are used to the old ways of developing software systems and therefore require extensive discussion on the usage of the artifacts in FSSSD.

In general, the early feedback and comments (Table 4) from case study one and two shows that the Framework for Sustainability of Software System Design (FSSSD) provides guidance and support for sustainability in software design requirements and development. The tools, methods and concepts provided as

sample in the framework helped in providing new insights into how sustainability can be incorporated into software project design and development especially the Sustainable Business Canvas, Goal model, Sustainability Requirement Template, Biomimicry, Cradle to cradle concept and Sustainability Analysis Radar Chat diagram. In addition, FSSSD also persuades stakeholders to rethink their software project with sustainability as a means of developing a better product that is cost effective over a long time and supports good corporate social responsibility. Table 4 summarizes the feedback on the usage of FSSSD from the case studies.

TABLE IV. DIRECT QUOTES FEEDBACKS AND COMMENTS FROM PARTICIPANTS AND STAKEHOLDERS IN USING FSSSD (CASE STUDY ONE AND TWO)

Role	**SDLC Phase**	**Positive**	**Challenges**
CTO	Project Definition	1. The SSDC was good way to understand the different aspect of sustainability for different kind of software system. The SSDC made it possible for me and my team to know more about sustainability in software development with those guidelines provided for each software system. 2. The FSSSD provides new insight for sustainability in software project with consideration of sustainability principles 3. Combination of the SSDC and FSSSD provides an avenue to consider our software impacts and see how we can minimise it. 4. FSSSD introduces new methods for evaluating our applications especially the environmental and individual dimensions of sustainability 5. The Sustainable Business Canvas brings in a totally new factors into software project definition with sustainability concepts and dimensions as guide	1. Very difficult to understand how to apply some of the sustainability concepts because its new to me and my team 2. We have a challenge to find concrete examples online to see how sustainability was applied to software project definition especially in industry 3. It was challenging to give my staff additional task of reading the Framework manual to understand how to apply it
Software developer, Project coordinator	User requirement definition	1. The sustainability requirement template was useful as guide during requirement gathering because it provides us with means of discussing sustainability with users and categorising user requirements base on sustainability dimensions	It was difficult at first to understand how to explain the different dimensions of sustainability to key stakeholders (users) during discussion gathering requirements on how to improve the existing system
System analyst, software developer	System Requirements Definition	1. I was able to learn new things about how sustainability can influence gathering system requirements and identifying new system requirements using the FSSSD 2. The goal model diagram is really a good tool to breakdown sustainability goals base on requirements into business, usage and system goal. 3. The goal model diagram made it easy to explain, discuss and improve the project goals and system requirements using the business, usage and system goal diagram.	1. The only issue is lack of examples to show how sustainability has been used in different software requirements elicitation at the beginning when using FSSSD but after couple of meetings discussing about sustainability with the research guy things became clearer. 2. Some of the research especially about sustainability in system requirements I saw on google from some researchers are too complex to apply
System analyst, Programmers, Software developer	Analysis and Design	1. The sustainability goals and suggested tools from FSSSD was a good starting point to guide us during the analysis and design phase. 2. The sustainability analysis radar chat was a new interesting tool because it shows some new requirements to add after brainstorming on each of the first, second and third impacts	Brainstorming on how to connect the first, second and third order impact in each of the sustainability dimensions was not easy because each of us have different views on what is the right thing to put but eventually we looked at some of the examples provided by the researcher guy in using FSSSD.

VII. CONCLUSION

Software design and development in the real world is continuously changing with the adoption of new software development methods and paradigms, such as agile, to reduce the development time from different SDLC phases and shortened time to market. However, sustainability is currently not at the core of the general development methodology in companies. Sustainability as a main principle and value provides a competitive advantage for companies and software designers /developers but the major challenge is the lack of understanding on how to institutionalize sustainability in software design and development projects.

This paper summarizes our early results on applying the Framework for the Sustainability of Software System Design (FSSSD) (Figure 1 and Table 1) in two case studies. The FSSSD provides support for sustainability in software design through the aspect of promoting sustainability goals at each stage of a software development life cycle phase with aid from different sustainability concepts, tools and methods as seen in case study 1 (Table 2) and case study 2 (Table 3 and Figure 4, 5). It also encourages a sustainability-oriented software development mindset over time with usage of FSSSD, because sustainability becomes part of the core fundamental values for software design and development practice.

Discussions with stakeholders and feedback in each of the case studies (Table 4) shows the major challenge in application of sustainability to software design and development is the lack of readily available software system industry examples and best practices of how core principles of sustainability are applied and exemplified in software projects.

Another challenge is in shifting developers' mindsets to adopting sustainability in a way that translates into their software design and development decisions and practices. The concept of sustainability dimensions (social, individual, environmental, economic, and technical) only becomes interesting to apply in software design if it can provide companies with opportunities for cutting costs and offer a competitive advantage in one way or another through usage of the framework.

The next phase is to repeatedly apply the FSSSD to different kinds of software projects and record best practices from each of these projects that can then be disseminated to interested stakeholders. Our template for documenting software sustainability requirement elicitation best practice during software design and development [39] can serve as template for such documentation.

ACKNOWLEDGMENT

The authors would like to thank Tom Mistretta and Alexandre Devinez for their work in the Living Lab at LUT University to support this research work.

REFERENCES

[1] S. Oyedeji, A. Seffah, and B. Penzenstadler, "A catalogue supporting software sustainability design," *Sustainability*, vol. 10, no. 7, pp. 1–30, 2018.

[2] N. Shedroff, *Design is the Problem:The Future of Design Must be Sustainable*. Rosenfeld Media, 2009.

[3] R. Chitchyan, L. Duboc, C. Becker, S. Betz, B. Penzenstadler, and C. C. Venters, "Sustainability Design in Requirements Engineering : State of Practice," pp. 533–542, 2016.

[4] M. Mahaux and C. Canon, "Integrating the Complexity of Sustainability in Requirements Engineering," *First Int. Work. Requir. Eng. Sustain. Syst.*, 2012.

[5] U. K. Jannat, "Green Software Engineering Adaption In Requirement Elicitation Process," vol. 5, no. 08, pp. 94–98, 2016.

[6] United Nations, "Sustainable Development Goals Available at: https://www.un.org/sustainabledevelopment/sustainable-development-goals/ Accessed on 28-12-2018," no. September 2000, pp. 8–23, 2015.

[7] S. Oyedeji, A. Seffah, and B. Penzenstadler, "Classifying the Measures of Software Sustainability," in *Proceedings of the 4th International Workshop on Measurement and Metrics for Green and Sustainable Software Systems co-located with 12th International Symposium on Empirical Software Engineering and Measurement (ESEM 2018)*, 2018.

[8] G. Deloittte, "2016 Mobile Industry Impact Report: Sustainable Development Goals," 2016.

[9] C. Calero and M. Piattini, "Introduction to Green in software engineering," *Green Softw. Eng.*, pp. 1–327, 2015.

[10] J. García-Berna, J. Carrillo de Gea, B. Moros, J. Fernández-Alemán, J. Nicolás, and A. Toval, "Surveying the Environmental and Technical Dimensions of Sustainability in Software Development Companies," *Appl. Sci.*, vol. 8, no. 11, p. 2312, 2018.

[11] H. Knut, R. Martin, S. Ingridvon, A. Michael, K. David, and K. Nina, "Sustainability Nears a Tipping Point Sustainability Nears a Tipping Point," *MIT Sloan Manag. Rev.*, vol. 53, no. 2, pp. 69–74, 2012.

[12] Ericsson, "Energy and Carbon Report," no. June, p. 12, 2013.

[13] Microsoft, "Microsoft 2015 Citizenship Report," 2015.

[14] S. Oyedeji, A. Seffah, and B. Penzenstadler, "Sustainability Quantification in Requirements Informing Design," *6th Int. Work. Requir. Eng. Sustain. Syst.*, vol. i, 2017.

[15] G. saval Martin, mahaux, patrick heymans, "Requirements Engineering: Foundation for Software Quality," *Requir. Eng. Found. Softw. Qual.*, vol. 4542, no. January, pp. 247–261, 2007.

[16] C. Becker *et al.*, "Sustainability Design and Software: The Karlskrona Manifesto," *Proc. - Int. Conf. Softw. Eng.*, vol. 2, pp. 467–476, 2015.

[17] B. Penzenstadler, "What does Sustainability mean in and for Software Engineering ?," *1st Int. Conf. ICT Sustain.*, 2013.

[18] B. Penzenstadler and H. Femmer, "A generic model for sustainability with process- and product-specific instances," *GIBSE 2013 - Proc. 2013 Work. Green Softw. Eng. Green by Softw. Eng.*, no. June 2015, pp. 3–7, 2013.

[19] C. C. Venters *et al.*, "Software sustainability: The modern tower of babel," *3rd Int. Work. Requir. Eng. Sustain. Syst. Work. Proc.*, vol. 1216, pp. 7–12, 2014.

[20] M. Al Hinai and R. Chitchyan, "Engineering Requirements for Social Sustainability," *Proc. ICT Sustain. 2016*, 2016.

[21] B. Penzenstadler, "Software Engineering for Sustainability."

[22] E. Blevis, C. Preist, D. Schien, and P. Ho, "Further Connecting Sustainable Interaction Design with Sustainable Digital Infrastructure Design," *Proc. 2017 Work. Comput. Within Limits - LIMITS '17*, pp. 71–83, 2017.

[23] G. G. Calienes, "Requirements Prioritization Framework for

developing Green and Sustainable Software using ANP - based Decision Making," pp. 1–9, 2013.

[24] B. Penzenstadler, "RE4ES: Support Environmental Sustainability by Requirements Engineering," *First Int. Work. Requir. Eng. Sustain. Syst.*, 2012.

[25] P. Bozzelli, Q. Gu, and P. Lago, "A systematic literature review on green software metrics," *Sis.Uta.Fi*, 2013.

[26] A. Raturi, B. Penzenstadler, B. Tomlinson, and D. Richardson, "Developing a sustainability non-functional requirements framework," *Proc. 3rd Int. Work. Green Sustain. Softw. - GREENS 2014*, pp. 1–8, 2014.

[27] C. C. Venters *et al.*, "The Blind Men and the Elephant Towards an Empirical Evaluation Framework for Software Sustainability," vol. 2, no. 1, pp. 1–6, 2014.

[28] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, "Safety, security, now sustainability: The nonfunctional requirement for the 21st century," *IEEE Softw.*, vol. 31, no. 3, pp. 40–47, 2014.

[29] K. Roher and D. Richardson, "Sustainability requirement patterns," *2013 3rd Int. Work. Requir. Patterns, RePa 2013 - Proc.*, pp. 8–11, 2013.

[30] L. Bonanni, D. Busse, J. Thomas, E. Blevis, M. Turpeinen, and N. J. Nunes, "Visible - Actionable - Sustainable: Sustainable Interaction Design in Professional Domains," *Proc. CHI 2011*, pp. 2413–2416, 2011.

[31] J. Froehlich, L. Findlater, J. Landay, and C. Science, "The Design of Eco-Feedback Technology," pp. 1999–2008, 2010.

[32] A. Spagnolli *et al.*, "Eco-feedback on the go: Motivating energy awareness," *Computer (Long. Beach. Calif).*, vol. 44, no. 5, pp. 38–45, 2011.

[33] R. J. Yun, A. Aziz, and B. Lasternas, "Design Implications for the Presentation of Eco- feedback Data," *Arch. Des. Res.*, vol. 28, no. 4, pp. 95–106, 2015.

[34] P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, and M. Winckler, *Interaction – INTERACT 2013 14th IFIP TC 13 International Conference*, no. 1. 2013.

[35] K.-L. Kramer, *User Experience in the Age of Sustainability. Available at: http://dx.doi.org/10.1016/b978-0-12-387795-6.00001-9.* 2012.

[36] E. Blevis, "Sustainable Interaction Design : Invention & Disposal , Renewal & Reuse," pp. 503–512, 2007.

[37] D. Fallman, "Design-oriented Human — Computer Interaction," no. 5, pp. 225–232, 2003.

[38] "Strongly Sustainable Business Model Canvas Http://www.ssbmg.com/ Accessed on 18-04-2019."

[39] S. Oyedeji and B. Penzenstadler, "Karlskrona Manifesto: Software requirement engineering good practices," *CEUR Workshop Proc.*, vol. 2223, pp. 15–23, 2018.

ACTA UNIVERSITATIS LAPPEENRANTAENSIS

847. GRADOV, DMITRY. Experimentally validated numerical modelling of reacting multiphase flows in stirred tank reactors. 2019. Diss.

848. ALMPANOPOULOU, ARGYRO. Knowledge ecosystem formation: an institutional and organisational perspective. 2019. Diss.

849. AMELI, ALIREZA. Supercritical CO2 numerical modelling and turbomachinery design. 2019. Diss.

850. RENEV, IVAN. Automation of the conceptual design process in construction industry using ideas generation techniques. 2019. Diss.

851. AVRAMENKO, ANNA. CFD-based optimization for wind turbine locations in a wind park. 2019. Diss.

852. RISSANEN, TOMMI. Perspectives on business model experimentation in internationalizing high-tech companies. 2019. Diss.

853. HASSANZADEH, AIDIN. Advanced techniques for unsupervised classification of remote sensing hyperspectral images. 2019. Diss.

854. POPOVIC, TAMARA. Quantitative indicators of social sustainability applicable in process systems engineering. 2019. Diss.

855. RAMASAMY, DEEPIKA. Selective recovery of rare earth elements from diluted aqueous streams using N- and O –coordination ligand grafted organic-inorganic hybrid composites. 2019. Diss.

856. IFTEKHAR, SIDRA. Synthesis of hybrid bio-nanocomposites and their application for the removal of rare earth elements from synthetic wastewater. 2019. Diss.

857. HUIKURI, MARKO. Modelling and disturbance compensation of a permanent magnet linear motor with a discontinuous track 2019. Diss.

858. AALTO, MIKA. Agent-based modeling as part of biomass supply system research. 2019. Diss.

859. IVANOVA, TATYANA. Atomic layer deposition of catalytic materials for environmental protection. 2019. Diss.

860. SOKOLOV, ALEXANDER. Pulsed corona discharge for wastewater treatment and modification of organic materials. 2019. Diss.

861. DOSHI, BHAIRAVI. Towards a sustainable valorisation of spilled oil by establishing a green chemistry between a surface active moiety of chitosan and oils. 2019. Diss.

862. KHADIJEH, NEKOUEIAN. Modification of carbon-based electrodes using metal nanostructures: Application to voltammetric determination of some pharmaceutical and biological compounds. 2019. Diss.

863. HANSKI, JYRI. Supporting strategic asset management in complex and uncertain decision contexts. 2019. Diss.

864. OTRA-AHO, VILLE. A project management office as a project organization’s strategizing tool. 2019. Diss.

865. HILTUNEN, SALLA. Hydrothermal stability of microfibrillated cellulose. 2019. Diss.

866. GURUNG, KHUM. Membrane bioreactor for the removal of emerging contaminants from municipal wastewater and its viability of integrating advanced oxidation processes. 2019. Diss.

867. AWAN, USAMA. Inter-firm relationship leading towards social sustainability in export manufacturing firms. 2019. Diss.

868. SAVCHENKO, DMITRII. Testing microservice applications. 2019. Diss.

869. KARHU, MIIKKA. On weldability of thick section austenitic stainless steel using laser processes. 2019. Diss.

870. KUPARINEN, KATJA. Transforming the chemical pulp industry – From an emitter to a source of negative CO2 emissions. 2019. Diss.

871. HUJALA, ELINA. Quantification of large steam bubble oscillations and chugging using image analysis. 2019. Diss.

872. ZHIDCHENKO, VICTOR. Methods for lifecycle support of hydraulically actuated mobile working machines using IoT and digital twin concepts. 2019. Diss.

873. EGOROV, DMITRY. Ferrite permanent magnet hysteresis loss in rotating electrical machinery. 2019. Diss.

874. PALMER, CAROLIN. Psychological aspects of entrepreneurship – How personality and cognitive abilities influence leadership. 2019. Diss.

875. TALÁSEK, TOMÁS. The linguistic approximation of fuzzy models outputs. 2019. Diss.

876. LAHDENPERÄ, ESKO. Mass transfer modeling in slow-release dissolution and in reactive extraction using experimental verification. 2019. Diss.

877. GRÜNENWALD, STEFAN. High power fiber laser welding of thick section materials - Process performance and weld properties. 2019. Diss.

878. NARAYANAN, ARUN. Renewable-energy-based single and community microgrids integrated with electricity markets. 2019. Diss.

879. JAATINEN, PEKKO. Design and control of a permanent magnet bearingless machine. 2019. Diss.

880. HILTUNEN, JANI. Improving the DC-DC power conversion efficiency in a solid oxide fuel cell system. 2019. Diss.

881. RAHIKAINEN, JARKKO. On the dynamic simulation of coupled multibody and hydraulic systems for real-time applications. 2019. Diss.

882. ALAPERÄ, ILARI. Grid support by battery energy storage system secondary applications. 2019. Diss.

883. TYKKYLÄINEN, SAILA. Growth for the common good? Social enterprises' growth process. 2019. Diss.

884. TUOMISALO, TEEMU. Learning and entrepreneurial opportunity development within a Finnish telecommunication International Venture. 2019. Diss.

ISBN 978-952-335-456-2
ISBN 978-952-335-457-9 (PDF)
ISSN-L 1456-4491
ISSN 1456-4491
Lappeenranta 2019