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 crucial 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 (antipatterns) 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.

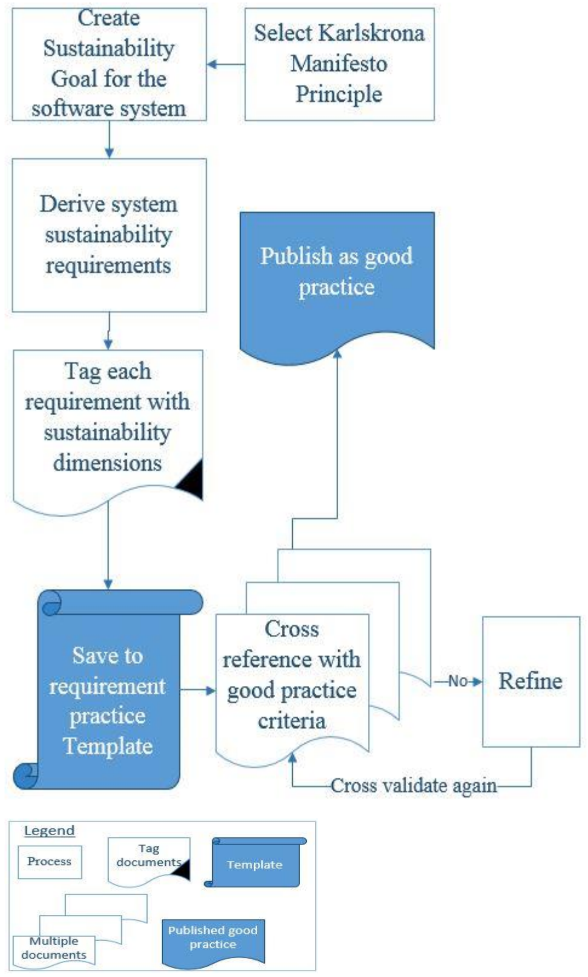
Create Sustainability Goal for the software system
Select Karlskrona Manifesto Principle
Derive system sustainability requirements
Publish as good practice
Tag each requirement with sustainability dimensions
Save to requirement practice Template
Cross reference with good practice criteria
No
Refine
Cross validate again
Legend
Process
Tag documents
Template
Multiple documents
Published good practice

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.