Software Engineering for Sustainability

Habilitationsschrift

Dr. Birgit Penzenstadler
Software Engineering for Sustainability

Habilitation

Candidate: Dr. Birgit Penzenstadler
penzenst@in.tum.de

Date: January 28, 2015

Fachmentorat (Mentor Group): Univ.-Prof. Dr. Dr. h.c. Manfred Broy
Univ.-Prof. Dr. Debra Richardson
Univ.-Prof. Dr. Helmut Seidl
Summary

Due to various resource shortages and changes in our environment, sustainability and resilience have become important topics in today’s research. Also software engineering research has put sustainability on its list of pressing issues. It has long-term impact on research and practice, especially in requirements engineering (RE), as RE lays the foundation for systems development by analysis and modification of business processes and surrounding application domains and, consequently, provides the strongest leverage point for sustainability.

With regard to research on sustainability within software engineering, first steps have been taken in applying existing requirements engineering methods to specific case studies limited to environmental sustainability and defining first design guidelines for green web engineering. Currently, there is no common understanding of how software engineering, and especially requirements engineering, can help in supporting sustainability.

In my habilitation, I aim at closing this gap and seek answers to the questions of the meaning of sustainability in software engineering, the involved stakeholders, and how to support sustainability already within requirements engineering.

The first step for this was to develop definitions for what sustainability means for software systems and for software engineering. On that basis, we established concepts for stakeholders for sustainability, a sustainability reference goal model, and sustainability as a non-functional characteristic of software systems. These concepts were integrated in a requirements engineering approach for explicitly supporting sustainability. Evaluations were performed in university settings and in industrial settings for the individual concepts as well as for the integrated approach. All of these were reported on in a number of publications referenced in the report. In addition to that, the results have been integrated into teaching at Technische Universität München and the University of California, Irvine.

In general, software developers might not have established environmental sustainability as a top-ranked quality characteristic for their software systems, but they are becoming more aware of the issue and are willing to consider it if adequate guidance is available and cost-effective. The results of this project provide such guidance without much overhead in addition to the requirements engineering practices already in place at a company. An overview of the project is also provided at www.se4s.org.
### Contents

**I  Introduction**

1  **Introduction and Overview**
   1.1 Motivation ................................. 3
   1.2 Problem Statement .......................... 4
   1.3 Research Objectives ......................... 4
   1.4 Research Design .............................. 5
   1.5 Summary of Results .......................... 7
      1.5.1 WP 1: Understanding of Sustainability for Software Engineering 7
      1.5.2 WP 2: Concepts for Requirements Engineering for Sustainability 18
      1.5.3 WP 3: Feasibility Studies .................. 28
   1.6 Conclusion .................................. 31
      1.6.1 Summary of Conclusions .................. 31
      1.6.2 Impact and Implications ................. 32
      1.6.3 Limitations ............................... 33
      1.6.4 Future Work ............................... 34
   1.7 Integration into Teaching ................... 36
   1.8 Overview of Publications .................... 37

**II  Foundations**

2  **Sustainability in Software Engineering: A Systematic Literature Review** 45
3  **Systematic Mapping Study on Software Engineering for Sustainability** 57
4  **Towards a Definition of Sustainability in & for Software Engineering** 69
5  **ICT4S 2029: The Systems Supporting Sustainability in 15 Years?** 73
6  **Sustainability Design and Software: The Karlskrona Manifesto** 85
7  **Who is the advocate? Stakeholders for Sustainability** 97
8  **Towards Incorporating Sustainability while Taking SPM Decisions** 107
III Concepts

9 A Generic Model for Sustainability with Process- and Product-specific Instances 127

10 Supporting Sustainability Aspects in Software Engineering 133

11 RE4ES: Support Environmental Sustainability by Requirements Engineering 139

12 Infusing Green: Requirements Engineering for Green in and through software systems 147

13 Developing a Sustainability Non-Functional Requirements Framework 159

14 Safety, Security, ... now Sustainability! 169

15 Framing Sustainability as a Software Quality Property 179

IV Evaluation

16 RE@21: Time to sustain 193

17 Domestic Plant Guilds: A Software System for Sustainability 203

18 An Assessment Technique for Sustainability: Applying the IMAGINE Approach to Software Systems 213

19 Supporting Physicians by RE4S—Evaluating Requirements Engineering for Sustainability in the Medical Domain 223

V Education

20 Teach Sustainability in SE? 239

21 Jumpstart Sustainability in Seminars 245

22 Case Study-based RE Education: Evaluation and Lessons Learnt 255

23 Using Non-Profit Partners to Engage Students in RE 267

VI Bibliography

Bibliography 279

Bibliography 280
Acknowledgements

I would like to thank Prof. Dr. Dr. h.c. Manfred Broy, Prof. Dr. Debra Richardson, and Prof. Dr. Helmut Seidl for their supervision, mentoring, and feedback.

I would like to thank my research colleagues and collaborators for sharing their knowledge, giving feedback, and inspiring me: Bill Tomlinson, Lorenz Hilty, Daniel Mendez, Steve Easterbrook, Ankita Raturi, Juliet Norton, Six Silberman, Beth Karlin, Neil Young, Kristin Roher, Stefan Wagner, Xavier Franch, Camille Salinesi, Mahvish Khurum, Andre van der Hoek, Allison Cook, David Callele, Krzysztof Wnuk, Martin Mahaux, Eric Baumer, Marcel Pufal, Norbert Seyff, Leticia Duboc, Christoph Becker, Baki Cakici, Ruzanna Chitchyan, Georges Da Costa, Lynn Dombrowski, Malin Picha Edwardsson, Elina Eriksson, Gillian R. Hayes, Christina Herzog, Wolfgang Lohmann, Alistair Mavin, Melissa Mazmanian, Saeed Nayebeziz, Daniel Pargman, Donald J. Patterson, Jean-Marc Pierson, Kevin Simonson, Andrew W. Torrance, Patrick Heymans, Kai Petersen, Coral Calero, Henning Femmer, Veronika Bauer, Jonas Eckhardt, Patricia Lago, Rick Kazman, Niklaus Meyer, Maurizio Morisio, Hausi A Müller, Frances Paulisch, Giuseppe Scanniello, Olaf Zimmermann, Alex J. Stringfellow, Joseph J. LaViola Jr., Ian Sutherland, Paul Blazek, Hans Lundberg, Hagen Habicht, Markus Hartinger, and Andreas Fleischmann for inspiration, discussions, collaborations, feedback and support.

I would like to thank my students Christopher Arciniega, Oliver Feldmann, Susanne Klein, Ilya Krasnov, Joseph Mehrabi, Alejandra Rodriguez, and Anshu Singh, for their dedicated work.

I would like to thank my parents, my brothers, my cousins Jonathan Hall and Jacque Ferneau, Ken Chen, and all my wonderful friends for their love and support.
Part I

Introduction
1 Introduction and Overview

1.1 Motivation

Today, software is deeply entrenched in almost every aspect of our daily lives. At the same time, society is facing the ever increasing shortage of non-renewable resources and the limits to growth [45, 44]. On one hand, the use of information and communications technology (ICT) contributes significantly to the usage of our planet’s resources [19]. On the other hand, ICT bears a lot of potential for improving sustainability in all kinds of different application domains. One aspect of this potential for improvement is “greening through IT” [86]. “Greening through IT” is making our life more environmentally sustainable by technological support for our daily life. Another aspect is social sustainability by fostering communities with adequate software tools to improve their communication and collaboration. A third aspect is to find different economic models to move from a consumption-oriented society to a more dematerialized service society (i.e. decoupling the gross domestic product from resource extraction) [22].

Software engineering research has put sustainability on its list of pressing issues — for example, the topic of the biggest software engineering conference in 2012 was “Sustainable Software for a Sustainable World”. At the same time, it has long-term impact on research and practice, especially in requirements engineering (RE), as RE lays the foundation for systems development by analysis and modification of business processes and surrounding application domains and, consequently, provides the strongest leverage point for sustainability.

First steps have been taken in applying existing software engineering methods to specific case studies, mainly limited to environmental sustainability: Atkinson et al. [3] offer how to enhance web service agreements by adding additional specifications for energy consumption. Sierszceki et al. [80] present a study on Danfoss motor drives illustrating how and how much energy could be saved over the course of the products development and ten years of usage. Naumann et al. [50] provide a framework for “sustainable software engineering”. They investigate how web pages can be developed with little environmental impact, i.e., energy-efficiently, and offer a respective guideline for web developers. Gu et al. [20] propose a green strategy model that provides decision makers with the information needed to decide on whether to take “green” strategies and eventually how to align them with their business strategies. In contrast to the focus on energy-saving in both of these works, we consider a much broader definition of sustainability. Cabot et al. [12] report on a case study for sustainability as a goal for the ICSE’09 conference with i*-models to support decision making for future conference chairs. Stefan et al. [83] extend that with quantitative goal modeling techniques. Both works provide model instances for specific case studies while the work at hand uses a generic model for reference. Mahaux et al. [40] present a case study on a business information system for an event management agency and assessed how well some RE techniques support modeling of specific sustainability requirements. In contrast, our aim is to provide modeling means for integrating sustainability into any software system as a major objective.
The workshop series on Requirements Engineering for Sustainable Systems\(^1\) [10, 70] has brought a number of contributions on aspects like goal modeling, energy saving, complexity, sustainability-enhancing application domains, user participation, quality, and eco-aware design. All of those have in common that they explore a specific understanding of sustainability in a specific application context, but there is no common understanding for how a predefined interpretation of sustainability can be translated into the requirements for a software system.

### 1.2 Problem Statement

Currently, there is no common understanding of how software engineering, and especially requirements engineering, can help in supporting sustainability. This is mainly due to the fact that sustainability is interpreted in many different ways and specific solution approaches to satisfying one of these interpretations are specific for a particular context only. The problem starts with the lack of a common definition of sustainability. The reason for that is that sustainability is closely tied in with values. Consequently, interpretations of the concept differ significantly and approaches of how to incorporate sustainability remain limited in their effectiveness and impact.

To tackle this, there is need for a richer discussion and eventually a framework to be able to differentiate the understanding of sustainability in a specific context. Based on that differentiation, a systematic analysis approach is needed to support that understanding of sustainability throughout the software development process, and specifically throughout requirements engineering. Currently, there is no such framework available in software engineering, neither is there an analysis approach that emphasizes sustainability throughout requirements engineering.

### 1.3 Research Objectives

This habilitation aims at differentiating the broader understanding of sustainability and how to incorporate the concept into requirements engineering. Sustainability is a value. Therefore, its interpretation depends on the stakeholders. Sustainability can be differentiated into several dimensions that have to be analysed in detail, e.g., environmental, human, social, and economic [87, 18], and these aspects have to be combined to achieve sustainability. The habilitation at hand provides guidance in how to differentiate between the viewpoints on sustainability and how they relate to and can be constructively applied in requirements engineering.

The research questions defined for this habilitation were changed in their order (1 and 3 were exchanged) from how they had been presented in the goal definition, because as work progressed it became clear that the new order is more logical.

1. **RQ1:** What does sustainability mean with regard to software systems and how can it be understood for the different abstraction levels concerned during requirements engineering?

   The goal was to differentiate the understandings of sustainability and what that means for software systems, and then for software engineering and specifically requirements engineering. For this research question, we looked into the fields of sustainability science and transferred the lessons learned from there to a characterization that would work within the discipline of software engineering. The original phrasing “What does sustainability mean on the different abstraction levels

---

\(^1\)http://www.ics.uci.edu/~bpenzens/2014re4susy/
concerned during requirements engineering?” was slightly extended to include a characterization of sustainability with regard to software systems before going into more detail for specifically requirements engineering.

2. RQ2: **Who are the stakeholders that advocate sustainability? How can stakeholders be classified with respect to their interest in sustainability?**
   The goal was to understand what the different motivations are for supporting sustainability and which attitudes there are with regard to sustainability. For this research question, both stakeholders in software development and in different areas of sustainability concerns had to be considered.

3. RQ3: **How can sustainability in its different forms be emphasized in a requirements engineering approach?**
   The aspects of sustainability have a bearing on different levels of requirements engineering, for example, when considering business goals, system goals, and quality attributes. The goal was to understand how sustainability can be handled within a systematic requirements engineering approach from the rather abstract and generic goals at the beginning of a project to the more specific requirements later on. For that, different types of information systems were considered.

### 1.4 Research Design

From the research questions, we developed a work plan with deliverables as illustrated in Fig. 1.1. The habilitation aims at the following major contributions:
WP1 An analysis of the understanding of sustainability for software engineering. This includes

- A systematic literature review: This study contained two systematic mapping studies that were performed in iterations in 2011 and 2013 due to the fact that the field of interest expanded rapidly during that time.

- A characterization of software engineering for sustainability: On the basis of the different definitions in sustainability research, the characterization explains what sustainability means in and for software engineering.

- A stakeholder analysis: This part analyzed different methods for identifying stakeholders with an interest in sustainability and provides a reference checklist with a classification of these stakeholders.

- A value map: The Software Value Map [31] was extended to include the different dimensions of sustainability.

WP2 Concepts for an encompassing requirements engineering approach, including

- A reference model for decomposing sustainability: The reference model is a conceptual model that decomposes sustainability in different dimensions and the values that represent a dimension.

- A requirements engineering artifact model: The AMDiRE artifact model [48] was analyzed and tailored to be used with a requirements analysis approach that emphasizes taking into account sustainability.

- A requirements analysis approach: The analysis is guided by a set of questions that help exploring the problem space and solution space with regard to sustainability.

- An approach to sustainability as a quality: Considering sustainability as a quality that needs to be balanced with other software system qualities. A comparison with safety and security was elaborated as well as a reference model with quality criteria for sustainability that was based on available sustainability standards.

WP3 Evaluation of the feasibility of the concepts via case studies and the dissemination of the concepts. The studies explore different types of systems as well as different development contexts:

- The DriveNow System: This system by BMW, Sixt and Mini is a car sharing system with embedded and information system components.

- The jambit GmbH: This Software Development Company strives to improve their development processes with regard to sustainability.

- The Plant Guild Composer tool: This system is a garden design software for educating on how to apply permaculture principles.

- The Calico tool: This is a collaborative drawing tool for software engineers developed at the University of California, Irvine.
The Citizen Muscle Bootcamp: This is an online course on civic engagement by the Story of Stuff Project.

The Cognatio tool: This application improves the communication between patient and doctor in tracking progress and effects of medication.

1.5 Summary of Results

The following sections provide an overview of the major results. We start with the foundation, the literature studies and the characterization of what sustainability means in and for software engineering, the stakeholders, and the value map. We then present our concepts, the reference model for sustainability, the requirements engineering method with artifacts and analysis, and the quality model. Finally, we describe the case studies and a final mapping of the deliverables in the goal definition to the publications.

1.5.1 WP 1: Understanding of Sustainability for Software Engineering

Systematic Literature Review and Systematic Mapping Study

We contributed the first Systematic Literature Review (SLR) [60, 54] in this field to aid researchers who are motivated to contribute to that topic by providing a body of knowledge as starting point. We provided an overview of different aspects of sustainability in software engineering research with regard to research activity, investigated topics, identified limitations, proposed approaches, used methods, available studies, and considered domains. The applied method is a SLR in five reliable and commonly-used databases according to the (quasi-standard) protocol by Kitchenham et al. The search string was (sustainability OR environment OR ecolog OR green) AND (software engineering OR requirement OR software system). Although we included keywords in our search string that rather point to environmental sustainability, we found publications related to different dimensions of sustainability.

We assessed the 100 first results of each database ordered by relevance with respect to the search query. Of 500 classified publications, we regard 96 as relevant for our research questions. We sketched a taxonomy of their topics and domains, and provide lists of used methods and proposed approaches. Most of the excluded publications were ruled out because of an unfitting usage of terms within the search query. In 2011, there was little research coverage on the different aspects of sustainability in software engineering while other disciplines were more active. As we found less than expected for a body of knowledge on sustainability in software engineering, we decided to extend the inclusion to publications that we classified as a research we could learn from when further investigating sustainability in software engineering. This lead to additional paper categories being analyzed for their potential to contribute to building up knowledge for an approach to sustainability in software engineering; classified as sustainability concepts, sustainability application domains, sustainable software solutions, and sustainable hardware solutions. The overview of the research topic taxonomy is provided in Figure 1.2. The full details of the study are provided in [60, 54], see Chapter 2.

As follow-up to the International Workshop of Green Software in 2012, we collaborated on a joint summary of the limitations and risks as well as potential further research topics [36]. This helped to outline
Figure 1.2: Taxonomy of Research Topics identified in the SLR Results in 2011
the research questions for a more in-depth Systematic Mapping Study (SMS) in 2013 as the two years after the first study had brought up a number of workshop series on the topic, a conference, and an increased number of related journal publications. The objective was detailed in seven research questions:

- RQ1: What research topics are being addressed?
- RQ2: How have these research topics evolved over time?
- RQ3: How is sustainability support performed (e.g., models and methods)?
- RQ4: Which of those models and methods are used in practice?
- RQ5: Which research type facets have been considered in the contributions?
- RQ6: Which application domains have been considered?
- RQ7: Which research groups are most active and what is the distribution between academics and practitioners?

This time, the results were also classified according to their software engineering knowledge area [23], research facet [89], application domains, and used methods. The numbers of publications from identified topic clusters mapped to knowledge areas and research facets are depicted in Fig. 1.3. The results show that the topic of SE4S has received wide-spread attention in the software engineering community over the past few years. Due to the fact of being a relatively new area of research, there is rather little reported evidence of establishment in practice. At the same time, industry has recognized the topic and use the term sustainability all over, reminding of the Green IT hype, but now broadened to sustainability. As Green IT practices are by now further established in practice, hope remains that the same will come true for other sustainability practices. A low number of evaluation and experience papers in the reported evidence (Fig. 1.3) also suggest that the research area including its solutions are still somewhat immature. Furthermore, a fair distribution over a range of journals and venues indicates that the research community is still forming. However, a large number of topic clusters, focus areas and application domains indicates that research is being conducted in broad coverage of the area of SE4S. The following list sums up the major conclusions from the reported evidence [71, 72].

- RQ1: The research topic clusters that have been addressed include a variety of aspects ranging across the future of society, urban architecture and integration, energy efficiency, life cycle assessment, environmental management, smart grids, cloud services, carbon consumption, traffic strategies, and virtualization. The majority of publications are in the knowledge areas of Software Design, Engineering Management, Models and Methods, Process, Quality, and Requirements (Fig. 1.3).

- RQ2: Evolution of the research topics over time reveals a strong general development over the last four years, especially in the topic clusters of future of society, life cycle assessment, and energy efficiency.

- RQ3: Sustainability support is performed by a variety of models and methods that include general purpose (interviews, statistics, surveys), software engineering (goals, stakeholders, services, processes), systems engineering (LCA), as well as methods from geo sciences, earth sciences, urban planning, and energy management.
- **RQ4**: The reported evidence of benefits and limitations for usage of the approaches in practice is very limited (Fig. 1.3).
- **RQ5**: The most prominent research type facets were *Exploratory* and *Solution*, underpinning the results of RQ4 (Fig. 1.3).
- **RQ6**: The application domains that were predominantly considered are Software Engineering and Lifecycle, Systems Engineering and ICT, Energy Efficiency, Mobile Services and Cloud, Business and Economics, ULS Computing, Mechanics and Manufacturing, Nature and Agriculture, Metropolitan Areas and Housing, and Software Engineering Education.
- **RQ7**: There are three rather active research clusters but research is performed all over the world and distribution between academia and industry is currently unbalanced with roughly 80% of reported evidence from academia, the rest distributed between industry and mixed collaborations.

The full details of the study are provided in Chapter 3.

Figure 1.3: Topic clusters from the systematic mapping study in relation to knowledge areas and research facets.
Characterization of Software Engineering for Sustainability

Why consider Software Engineering for Sustainability? How is current SE not sustainable? Current SE practice is to make systems faster and more encompassing, which can lead to exponential growth in resource consumption. That is unsustainable as we only have limited natural resources available. However, this is driven by the business behind the software and, therefore, the economy. To include sustainability as a concern in the businesses, the added value for the costs caused by sustainability has to be proven, for example improvements of the company image, which is primarily the responsibility of the business analysts. However, as software engineers we are responsible for the long-term consequences of our designs. This section examines what is software engineering for sustainability. Earlier versions of the characterization were developed in [56, 55, 75]. The following is a revised and improved version of those contributions.

Sustainability. The first known European use of the word sustainability occurred in 1713 in the book Sylvicultura Oeconomica by Hans Carl von Carlowitz for “sustained-yield forestry” [21]. Since then, sustainability has been defined as:

- “The capacity to endure” [84], indicating simply endurance over time.
- “Preserving the function of a system over an extended period of time” [35], to serve as a starting point for scoping in systems analysis.
- “Ethics expanded in space and time” [42], adding the notion of ethics and values.
- “The possibility that all forms of life will flourish forever” [14], adding a notion of prosperity that includes non-human forms of life.

We will later on use the different definitions and see that the first ones ([84] and [35], without the notion of values) are domain-independent, and that the latter ones ([42] and [14], the ones that refer to values) are domain-dependent and therefore specifically relevant for the application domain context.

Sustainable development has been defined by the UN World Commission on Environment and Development (WCED) as “meeting the needs of the present without compromising the ability of future generations to meet their own needs”. In addition, the WCED denotes three dimensions of sustainability — economic, social, and environmental [87]. The WCED definition is widely used but not actionable; it doesn’t point out the unsustainability of the use of non-renewable resources and doesn’t account for population growth [6]. To make up for these insufficiencies, there have been efforts to define frameworks. Heinberg [21] distilled ideas that had been proposed previously in five axioms of sustainability: (1) Any society that continues to use critical resources unsustainably will collapse. (2) Population growth and/or growth in the rates of consumption of resources cannot be sustained. (3) To be sustainable, the use of renewable resources must proceed at a rate that is less than or equal to the rate of natural replenishment. (4) To be sustainable, the use of nonrenewable resources must proceed at a rate that is declining, and the rate of decline must be greater than or equal to the rate of depletion. (5) Sustainability requires that substances introduced into the environment from human activities be minimized and rendered harmless to biosphere functions [21].

To analyze sustainability in more detail, we decomposed it into its different dimensions [63]. Most concisely, the dimensions of sustainability are characterized as follows: Individual sustainability refers
to maintaining human capital (e.g., health, education, skills, knowledge, leadership, and access to services). Social sustainability aims at preserving the societal communities in their solidarity and services. Economic sustainability aims at maintaining capital and added value. Environmental sustainability refers to improving human welfare by protecting the natural resources: water, land, air, minerals and ecosystem services. Technical sustainability refers to domain-independent longevity of systems and infrastructure and their adequate evolution with changing surrounding conditions [63].

Four of these five dimensions are already supported to some extent by traditional software quality characteristics and requirements can be dealt with. Currently, the least support exists for the environmental dimension. Why environmental sustainability is such a wicked problem has been investigated by a number of psychologists, amongst them Beattie [7]. He shows that we don’t see the immediate effects of our actions because of the accumulative nature of the effects and the consequential time delay, and that we don’t know how to make the best choices because of the large number of influencing factors. To help analyzing these later effects, there are different orders of effect of systems: The first order effects are direct effects of its physical existence, e.g. the energy that a software system consumes; the second order effects are the effects of the systems use and application; and the third order effects are systemic effects [14].

Scoping. To define what sustainability for any kind of system, exact scoping needs to be performed by answering the questions of what to sustain, for whom, over which time frame, and at what cost [85]. For a software system, there are two scopes that may be chosen for an analysis, domain-dependent and domain-independent.

Domain-dependent scope: The software system in its application context, under consideration of the purpose of the software system. This scoping includes an analysis of the economic, social, and environmental dimensions of sustainability on a higher level. This part of the analysis is largely within the responsibility of business analysts and domain experts. Following the code of ethics [1], it is our responsibility to support in translating the arising demands into system-related requirements.

Characterization 1: The Sustainability of a Software System and its Surrounding Application Context, is characterized by the domain experts, for example according to the Heinberg axioms. This scoping is domain-dependent and we call it domain-dependent sustainability, software engineers can translate those requirements into constraints for the software system.

Domain-independent scope: The artifacts and implementation of a software system, without any consideration of the purpose of the software system. This scoping focuses on the technical characteristics of the system and its operational environment. All domain-relevant information has already been refined into domain-independent constraints for a technical solution by business analysts. This is the responsibility of the software engineers.

Characterization 2: The Sustainability of a Software System is its capacity to endure, i.e. its energy efficiency, its maintainability over a long period of time, and its adherence to standards and laws. This scoping is independent of the application domain of the software system; therefore we call this domain-independent sustainability.
**Characterization 3:** Software Engineering for Sustainability denotes the concept of applying software engineering techniques to facilitate the refinement of higher level sustainability concerns as defined in the domain into lower level, technical requirements for the design and implementation of the system. This means the translation and refinement from Characterization 1 to Characterization 2.

Consequently, the questions for Software Engineering for Sustainability are:

1. To what extent are there generic approaches that apply to all domains to help refine the domain-dependent sustainability into specific constraints for the software system under consideration?

2. Which parts of sustainability are already dealt with by other qualities? How do these qualities overlap with sustainability? Which of their methods can be applied for sustainability?

1. For **domain-dependent analysis of these dimensions**, all stakeholders that are affected by the system need to be considered by domain experts and business analysts. Economic sustainability is addressed by business economics, individual and social sustainability are covered by social sciences and high-level human computer interaction, and environmental sustainability is mainly addressed via regulations and policies. Note that the understanding and definition of sustainability might vary significantly between domains. There are methods and approaches that help with pinning down the dimensions of sustainability on the domain level: Economic sustainability is partly covered by software economics, but often there is a lack of long-term thinking [51]. Individual and social sustainability are partly addressed by a subfield in human computer interaction, namely sustainable HCI. Green IT and energy efficiency efforts in software engineering cover the first order effects in the environmental dimension. Note that energy efficiency by itself does not necessarily lead to resource savings. This is due to potential rebound effects. The second and third order effects in the environmental dimension of software systems are largely domain-dependent and cannot be analyzed in general. However, software engineers can be aware of these effects when receiving requirements from domain experts and business analysts.

2. For the **transition to the domain-independent level**, all the sustainability requirements on the domain-dependent level are refined for the software system under development. On this level, they are refined into functional requirements, into process requirements, and into standard software quality requirements (Quality in Use and Product Quality). Part of the latter is the domain-independent sustainability of a software system (as in Characterization 1). For example, some aspects of safety are considered in different sustainability dimensions on the domain-independent level; in the technical one as it is a standard quality attribute for systems, then the individual and social ones (user protection), and as a consequence of that also the economic dimension (insecure systems will not have market success).

Table 1.1 serves to show how different software qualities feature into translating domain-dependent sustainability into domain-independent sustainability for software systems. In the top row, the sustainability dimensions are lined up as headers for the columns of the table. On the left side, the criteria from the ISO 25010 standard [26] are listed to differentiate within the levels of sustainability. The domain-dependent level of sustainability (Characterization 1) is decomposed into the Quality in Use criteria, and domain-independent level of sustainability (Characterization 2) is decomposed into the Product Quality criteria. The fields of the table denote how the criterion (denoted by the table row) is to be understood in the context of the sustainability dimension (denoted by the table column). For example, on the domain-dependent level of sustainability, the Quality in Use criterion *Satisfaction* is understood in the context of social sustainability as the satisfaction for the user community, the community support which should
satisfy purpose accomplishment, trust, pleasure, and comfort. This is partially dealt with by research in sustainable HCI [81].

There is a number of conclusions to be drawn from the above characterizations and the table:

- **Sustainability is a very broad concept that is closely tied in with personal values.** Consequently, it is used in very different forms. One end of the spectrum is the capacity to endure [84], which can be interpreted as “survival of the fittest”, and that leads to a concept of competence without considering values. The other end of the spectrum is life flourishing forever [14], where flourish needs interpretation that will include personal values. Consequently, domain-dependent sustainability is strongly dependent on the project context and tied in with the values of the stakeholders. This strong interdependence makes it impossible to clearly decompose domain-dependent sustainability for the general case. Finally, how it is prioritized again depends on the importance that these stakeholders assign to it.

- **The differentiation between sustainability and other quality concepts in software engineering is non-trivial**, because the differentiation again requires the interpretation of sustainability. This interpretation is given for the domain-independent sustainability in the lower part of the table, where it is mainly considered via the constraints derived from the domain-dependent sustainability. For the domain-dependent sustainability, the relation described in the table fields is more of a scoping that is still subject to interpretation in the individual project context. Therefore the table can be used to extract a sustainability profile for domain-dependent sustainability using the five dimensions of sustainability and the Quality in Use criteria.

- **There are potentially conflicting quality criteria**, which again is tied to the specifics of the chosen definition for domain-dependent sustainability; for example sustainability and privacy: On one hand the absolute availability of information would make sustainability easier in many domains, and that is in conflict with privacy. For example, if companies had to provide all information on how “green” (environmentally sustainable) they really are with consumption of any type of resources, their processes, and their waste production. On the other hand, privacy can be understood as part of the individual sustainability of a person, for example security breaches in online banking. Consequently the occurrence of the conflict again depends on the definition of the domain-dependent sustainability and which values that definition is based on.

- **Domain-dependent sustainability can be refined into different known types of requirements.** However, we do need to provide software engineers, and specifically requirements engineers, with the guidance to translate and refine high-level, domain-dependent sustainability goals into system-specific requirements. This is the contribution that the habilitation at hand provides.

This characterization served as a basis for the contribution in [8], see Chapter 6.
### Table 1.1: Which parts of the domain-dependent sustainability are already dealt with by other qualities? How do these qualities overlap with sustainability? Which of their methods can be applied for sustainability?

<table>
<thead>
<tr>
<th>Quality</th>
<th>Economic sustainability</th>
<th>Individual sustainability</th>
<th>Social sustainability</th>
<th>Environmental sustainability</th>
<th>Technical sustainability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost analysis</td>
<td>Effectiveness for individual, can be dealt with by standard SE</td>
<td>Effectiveness for community, can be dealt with by standard SE</td>
<td>Environmental impact analysis</td>
<td>Technical effectiveness in application domain, standard SE</td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td>Cost analysis</td>
<td>Efficiency for individual, can be dealt with by standard SE</td>
<td>Efficiency for community, can be dealt with by standard SE</td>
<td>Energy efficiency</td>
<td>Technical efficiency in application domain, standard SE</td>
</tr>
<tr>
<td>Satisfaction (purpose accomplishment, trust, pleasure, comfort)</td>
<td>Satisfied customers (all sub-characteristics) is partially dealt with by sustainable business analysis</td>
<td>Personal satisfaction (all sub-characteristics) is partially dealt with by sustainable HCI [1][2]</td>
<td>Community support (all sub-characteristics) is partially dealt with by sustainable HCI</td>
<td>Environmental impact analysis (EIA, for all sub-characteristics) is available but not generally applied in SE</td>
<td>Purpose accomplishment (functional part of satisfaction), standard SE</td>
</tr>
<tr>
<td>Safety (economic damage risk, health and safety risk, environmental harm risk)</td>
<td>Economic damage risk is partially dealt with by sustainable business analysis</td>
<td>Health and safety risk on small scale is partially dealt with by safety analysis</td>
<td>Health and safety risk on larger scale is partially dealt with by safety analysis</td>
<td>Environmental harm risk is partially dealt with by Life Cycle Assessment (LCA)</td>
<td>Standard SE</td>
</tr>
<tr>
<td>Controllability (context completeness, flexibility)</td>
<td>Business context completeness and flexibility is partially dealt with by business analysis</td>
<td>Personal context completeness and flexibility is dealt with by business analysis, social science, and sustainable HCI</td>
<td>Social context (in case there is) is dealt with by social science and business analysis, sustainable HCI</td>
<td>Environmental context completeness is dealt with by LCA</td>
<td>Standard SE</td>
</tr>
<tr>
<td>Product Quality</td>
<td>Functional suitability (completeness, structuredness, appropriateness)</td>
<td>Indirect by derived constraints</td>
<td>Indirect by derived constraints</td>
<td>Indirect by derived constraints</td>
<td>Standard software engineering (SE)</td>
</tr>
<tr>
<td>Performance &amp; efficiency (time behavior, resource utilization)</td>
<td>Indirect by derived constraints</td>
<td>Indirect by derived constraints</td>
<td>Indirect by derived constraints</td>
<td>Additional constraints from EIA and LCA, Energy efficiency</td>
<td>SE</td>
</tr>
<tr>
<td>Compatibility (co-existence, interoperability)</td>
<td>Indirect by derived constraints</td>
<td>Indirect by derived constraints</td>
<td>Indirect by derived constraints</td>
<td>n.a.</td>
<td>SE</td>
</tr>
<tr>
<td>Convenience (appropriateness, recognizability, learnability, operability, use error protection, user interface aesthetics, accessibility)</td>
<td>Indirect by derived constraints</td>
<td>Indirect by derived constraints</td>
<td>Indirect by additional constraints derived from individual safety and satisfaction</td>
<td>n.a.</td>
<td>SE</td>
</tr>
<tr>
<td>Reliability (maturity, availability, fault tolerance, recoverability)</td>
<td>Indirect by derived constraints</td>
<td>Reliability for the individual user, dealt with by standard SE</td>
<td>Reliability for the user community, dealt with by standard SE</td>
<td>n.a.</td>
<td>SE</td>
</tr>
<tr>
<td>Security (confidentiality, integrity, non-repudiation, accountability, authenticity)</td>
<td>Protection of economic investment</td>
<td>Protection of individual user</td>
<td>Protection of user community</td>
<td>n.a.</td>
<td>SE</td>
</tr>
<tr>
<td>Maintainability (modularity, reusability, analyzability, modifiability, reproducibility)</td>
<td>Additional constraints derived for longevity</td>
<td>n.a.</td>
<td>n.a.</td>
<td>Additional constraints derived for longevity</td>
<td>SE</td>
</tr>
<tr>
<td>Portability (adaptability, installability, replaceability)</td>
<td>Additional constraints derived for longevity</td>
<td>n.a.</td>
<td>n.a.</td>
<td>Additional constraints derived for longevity</td>
<td>SE</td>
</tr>
</tbody>
</table>
Stakeholders for Sustainability—Who is the advocate?

Improvement of sustainability needs a driving force, an advocate. An objective that has no stakeholder is not likely to receive sufficient attention to be realized and will eventually disappear. In an interview study at a small-to-medium sized software development company in Munich, we found the hypotheses confirmed that sustainability is a high-level concept that people would like to support but often there is a lack of knowledge of how to do that. More specifically, there is a lack of guidance of how to incorporate it into software development [32].

On the basis of these preliminary results, we set out to answer the questions of “Who are the people who actually have an interest in improving the sustainability of a specific software system or of the discipline of software engineering itself? And who are the devil’s advocates?” In [65], we elaborated four approaches of identifying stakeholders for sustainability in a given context: top-down by sustainability dimensions (individual, social, environmental, economic, and technical), by instantiation of a generic list, by an organizational diagram (bottom-up), and iteratively by an activity model according to the generic sustainability model.

We document the stakeholders in a Stakeholder Model that allows to list and describe all stakeholders involved in a project. Stakeholders comprehend individuals, groups, or institutions having the responsibility for requirements and a major interest in the project. The Stakeholder Reference List in Tab. 1.2 is the result of an analysis of which stakeholders are impacted by or might be interested in a specific sustainability dimension. The left column lists the sustainability dimensions, the middle column the potentially affected stakeholders, and the right column the rationale of why that specific stakeholder is affected by that sustainability dimension. Thereby, some stakeholders show up multiple times, for example legislation, as they affect multiple dimensions of the sustainability of a system and its environment. The table serves as checklist for ensuring that goals have been elicited and constraints have been collected from all stakeholders.

The conclusion is: As the stakeholders are key for determining whether or not any objective is achieved, identifying the stakeholders for sustainability is crucial for successfully implementing and evaluating sustainability support in a given context [65]. Furthermore, some of the stakeholders are relevant for the domain-dependent characterization of sustainability in a specific development context, for example activists and community representatives, while others are more relevant for the domain-independent implementation of a specific domain-dependent characterization of sustainability, for example developers and maintenance personnel.

Value Map

In [67], we identify the value aspects related to sustainability for software requirements selection. An exemplary dialogue between a consultant and a product manager illustrates how the proposed approach can be used while taking product management and requirements selection decisions. This contribution provided software product managers with guidance on how to incorporate value aspects related to sustainability while taking software product management and requirements selection decisions. The Software Value Map (SVM) by Khurum [31], which was used as the basis for this part of the project, provides a consolidated view on value aspects relevant for taking software product management and development decisions based on the Balanced Score Card [29] approach. In addition to the SVM’s application for
<table>
<thead>
<tr>
<th>Sustainability Dimension</th>
<th>Stakeholder</th>
<th>Description/Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual Sustainability</td>
<td>User</td>
<td>The user is affected by the system in various ways. For example, users of online learning courses educate themselves through software.</td>
</tr>
<tr>
<td></td>
<td>Developer</td>
<td>The developer is heavily involved in creating the system. Aspects like sustainable pace and growth of the developer must be considered.</td>
</tr>
<tr>
<td></td>
<td>Employee representative</td>
<td>The mental and physical safety of individuals needs to be maintained. Employee representatives watch rights of employees involved.</td>
</tr>
<tr>
<td></td>
<td>Legislation (individual rights)</td>
<td>Systems must respect the rights of their users. A legislation representative is a proxy for privacy and data protection laws.</td>
</tr>
<tr>
<td>Social Sustainability</td>
<td>Legislation (state authority)</td>
<td>The state has a strong interest in understanding a system’s influence on the society. In contrary to the individual rights legislation representative, the state authority representative speaks from the perspective of the state as a whole.</td>
</tr>
<tr>
<td></td>
<td>Community representative</td>
<td>In addition to the state authority, other communities such as the local government (e.g. the mayor) or non-government clubs might be affected by a software system. A complete analysis must take their views into account.</td>
</tr>
<tr>
<td></td>
<td>CRM</td>
<td>The Customer Relationship Manager (CRM) is in charge of establishing long-term relationships with their customers and creating a positive image of the company.</td>
</tr>
<tr>
<td></td>
<td>CSR manager</td>
<td>Some companies created the dedicated position of the Corporate Social Responsibility (CSR) manager, who develops a company-specific vision of social responsibility.</td>
</tr>
<tr>
<td>Economic Sustainability</td>
<td>CEO</td>
<td>The chief executive officer integrates sustainability goals into a company’s vision.</td>
</tr>
<tr>
<td></td>
<td>Project manager</td>
<td>It is very important to have the project manager agree in what ways the project should support sustainable aspects as he decides on prioritisation with conflicting interests.</td>
</tr>
<tr>
<td></td>
<td>Finance responsible</td>
<td>As sustainable software engineering often also affects the budget, many financial decisions have to be made to implement a sustainable software engineering model in a company.</td>
</tr>
<tr>
<td>Environmental Sustainability</td>
<td>Legislation (state authority)</td>
<td>Environment protection laws are in place to ensure sustainability goals. These laws must be reflected in the requirements model for a system.</td>
</tr>
<tr>
<td></td>
<td>CSR manager</td>
<td>The CSR manager is one of the responsible persons for environmental aspects and potential impacts on the business context.</td>
</tr>
<tr>
<td></td>
<td>Activists/Lobbyists</td>
<td>Nature conservation activists and lobbyists (e.g., WWF, Greenpeace, BUND).</td>
</tr>
<tr>
<td>Technical Sustainability</td>
<td>Admin</td>
<td>The administrator of a software system has a strong motivation for long-running, low-maintenance systems, making his work easier.</td>
</tr>
<tr>
<td></td>
<td>Maintenance</td>
<td>The hardware maintenance is interested in a stable, long-term strategy for installation of hardware items.</td>
</tr>
<tr>
<td></td>
<td>Customer</td>
<td>Users are interested in certain longevity of the systems they are using. This refers to user interface and required sof- and hardware.</td>
</tr>
</tbody>
</table>
sustainability concerns, a set of value aspects not yet covered by the Software Value Map has also been included, where each aspect is described and given a rationale. The identified value aspects can be used as criteria for taking requirements selection decisions according to the process depicted in Fig. 1.4. As a follow-up, a survey was conducted and replicated a year later at the same conference that targets academics and practitioners [68] to gain more insight on these value aspects and how they are handled in practice. The results made clear that environmental sustainability is a consideration for most software developers but only has minor effects when put into practice for software product decisions.

1.5.2 WP 2: Concepts for Requirements Engineering for Sustainability

Generic Reference Model for Sustainability

Software systems in most cases have an economic purpose and/or fulfill human or social needs of their users. The economic purpose is analyzed by economy itself; the latter goals are analyzed in software engineering by user-centric techniques, such as service orientation. Yet, as software systems have an impact on the environment, environmental sustainability should be added to these latter aspects for software development projects. The problem is that without applicable guidance, sustainability remains an intangible ideal. Therefore, we need a definition and a concrete decomposition of sustainability to relate it to software systems development. It is not sufficient to analyze environmental sustainability on its own, but instead its interplay with other aspects in order to define appropriate actions and understand their effects. The principal idea is to analyze the dimensions of sustainability, their values with respective indicators, and activities to support them. These elements compose a conceptual model that allows for analyzing and constructing actions both for a company or a product point of view. In this contribution, we propose a generic sustainability model with instances for companies and projects from various case studies. We thus enable analysis, support and assessment of environmental sustainability in software engineering.

According to the most cited definition for sustainable development [87], sustainability is characterized by the three dimensions economic, social, and environmental. This characterization is extended with a fourth dimension human (or individual) by Goodland [18], portraying the individual development of every human over their life time. When analyzing the sustainability of IT systems, these four dimensions apply, and an additional dimension, technical, supports better structuring of concerns with respect to software systems. Most concisely, we characterized the dimensions of sustainability in [65, 8] as:

- Individual sustainability refers to maintaining human capital (e.g., health, education, skills, knowledge, leadership, and access to services).
- **Social sustainability** aims at preserving the societal communities in their solidarity and services.

- **Economic sustainability** aims at maintaining capital and added value.

- **Environmental sustainability** refers to improving human welfare by protecting the natural resources: water, land, air, minerals and ecosystem services.

- **Technical sustainability** refers to longevity of systems and infrastructure and their adequate evolution with changing surrounding conditions.

We present a reference model for sustainability that decomposes sustainability into five dimensions: environmental, individual, social, economic, and technical sustainability. The model provides activities and relates them to values they support and assessable indicators. It is intended to serve as a reference model for a process engineer who instantiates the model for a software development company or for a requirements engineer who instantiates it for a specific system under development.

An excerpt of the generic sustainability model is provided in Figure 1.5, see [62] for details. The model consists of three levels: the top level contains the dimensions; the middle level contains values, indicators, and regulations; and the lower level contains activities. The elements in the generic sustainability model have the following types:

- **Dimensions**: A dimension is an aspect of or viewpoint on sustainability, for example, environmental sustainability. As described above, there are five dimensions to be considered when analyzing sustainability for software systems engineering. A dimension is detailed in a set of values.

- **Values**: A value is a moral or natural good that is perceived as an expression of a specific dimension. Each of the five dimensions is represented by a set of values. Values do not necessarily belong exclusively to one dimension but can be considered for a number of dimensions, for example, healthy environment, which applies for both the environmental as well as the individual dimension.

- **Indicators**: An indicator is a qualitative or quantitative metric that expresses a specific degree or score with regard to a value, for example, satisfaction indices as qualitative metric and carbon emissions or return on investment as quantitative metrics. A set of indicators approximates a value.

- **Regulations**: A regulation is an optional element that affects (i.e. supports or enforces) a value, for example, emission regulations. Regulations commonly set limits for a specific indicator to be of legal use. Many values belonging to the different dimensions are heavily regulated, either supported or restricted in order to protect them. For example, freedom of the individual is supported by the human rights, and healthy air is supported by the European Union's directive on carbon emissions.

- **Activities**: An activity is a measure taken to contribute to a specific value or a set of values, for example, optimize resource consumption. The impact of these activities on a value is measured by the indicators it influences. For the travel example, using a train instead of an aircraft improves the emissions account of the traveler. For each value, there is a number of activities that can be implemented to support a value. Thereby, the impact of an activity is measured by the indicators it influences.
Figure 1.5: Illustrative Excerpt of the Reference Model for Sustainability [63, 62]
The generic sustainability model is intended to serve as reference and as a basis for the instantiation of company- or system-specific instances. In the instantiation process, the generic values are replaced by context-specific goals. The instantiation consists of two phases, an analysis phase and an application and assessment phase. During the analysis phase, the generic values that are provided by the generic model are instantiated into context-specific goals. These goals need to be prioritized to help solving conflicts between potentially contradicting goals. Then activities to implement the goals are defined as well as indicators that make it possible to assess the state of this goal at the moment as well as in future situations. When values and indicators for a certain dimension are selected, the activities that have a positive impact on the indicators can be related. For the application and assessment phase, a responsible for each activity needs to be defined and a quality responsible monitors the overall status of the indicators. For the full details on the reference model and its application, see [63] for an encompassing description in Chapter 9. Instantiations were elaborated in case studies on a car sharing system [33, 15] and a software development company [32]. The excerpt in Figure 1.5 is part of the generic reference model that was used as basis to derive the instances in the case studies. In addition, the specific contexts of the instances were used to derive more elements for and thereby enhance the generic reference model.

By using the generic sustainability reference model for eliciting additional goals with regard to sustainability, requirements engineers can support their clients in satisfying the new objective of sustainability (of a software system in its application domain) that is gaining increasingly more attention by customers, as shown in the car sharing system study [33, 15]. Furthermore, the reference model can be used to enhance the sustainability aspects of development processes, as shown in application at the software development company [32]. On the basis of the elicited goals, the related indicators can be integrated into a sustainability profile for a quality model.

Artifact Model and Analysis Approach

To guide requirements engineers through incorporating sustainability as a major goal in software systems development, we follow an approach that can be integrated in established practices and be negated by software engineers, as sustainability in any characterization is only one of many objectives for a system. The software engineering community has achieved to integrate safety and security as additional objectives into requirements engineering [13, 39], and now we propose to do the same for sustainability [73]. We show that requirements engineering can accommodate the new objective of improving the environmental sustainability of software systems using its current techniques and incorporating simply a few more instantiations of known requirements types. We introduced the idea of supporting sustainability in requirements engineering in general in [53], and specifically for environmental sustainability in [76]. We adapted our AMDiRE approach [48, 5, 4](Artifact Model for Domain-independent Requirements Engineering) that includes a detailed artifact model that captures the basic modeling concepts used to specify RE-relevant information, for documenting sustainability requirements. In [58], we explain how to build the foundation for requirements engineering on a basic system model and how this basis enables to integrate sustainability requirements into the different artifacts. It turned out that instead of having to define additional artifacts for sustainability requirements, it made more sense to incorporate the sustainability aspects on the different abstraction levels in the artifacts that were already defined.

For the description of how to elaborate requirements that reflect the different aspects of sustainability, we
limit ourselves to a few concepts that are commonly agreed on as content items or information elements for gathering and refining requirements, all depicted in the overview in Figure 1.6. These are Business Processes, Domain Models, Stakeholders, Objectives, Constraints, System Vision, Usage Model, as well as Quality Requirements, Process Requirements, Deployment Requirements, and System Constraints.

Figure 1.6: Activity Model with Content Items and Information Flow for Requirements Engineering for Sustainability

Starting from scratch for developing any type of software system, how would we elaborate requirements for sustainability within a generic requirements engineering approach? There are a few questions to help guide the way:

Q1 *Does the system under consideration have an explicit purpose for domain-dependent sustainability?* If yes, this can be analyzed in depth. If no, it can be considered whether such an aspect is desirable and feasible to add. If, again, that is not the case, then the analysis details the potentials for domain-independent sustainability of that IT system (further explored in Q2), but depending on the kind of system this might still lead to considerable improvements of the environmental, economic, or social impact of the system. In case the system is widely used, that optimization is worth the effort.

Q2 *Which impact does the system under consideration have on the surrounding operational environment?* Any system has an impact on the environment, as any system is applied in a real world context of some kind, which is situated within our natural environment. Consequently, it has to be analyzed as to what are the direct (first order), indirect (second order), and systemic as well as potential rebound effects (third order). This potentially includes a very large scope, especially for third order effects, but systemic thinking [43] facilitates such an analysis process and may lead to significant insights.

Q3 *Is there an explicit stakeholder for sustainability?* In case there is a specific stakeholder who advocates for environmental sustainability, there is already a significant voice that issues objectives, constraints and considerations to support that quality in the system under consideration. In case
there is no such advocate, it can be decided to establish such a role. Otherwise, the least representative for sustainability that should be established is a domain expert responsible for providing information on applying respective standards, legislation, and regulations.

**Q4 What are the sustainability goals and constraints for the system?** Independent of whether the system has an explicit purpose for supporting domain-dependent sustainability or not, there certainly is a number of objectives that pertain to the different dimensions of sustainability that may be chosen to apply. For example, a social network might not have an explicit environmental purpose, but it certainly has objectives supporting social purposes with regard to domain-dependent sustainability. Furthermore, any system will at least have some constraints with respect to the operational environment, as stated in Q2.

The sustainability requirements and constraints are propagated throughout the content items in requirements engineering as illustrated in Fig. 1.6. This includes the activities **derive sustainable system vision**, **specify sustainable interaction** and **refine and deduce sustainability requirements**.

The impact of this contribution is mainly determined by the question how much difference the consideration of sustainability actually makes in requirements engineering. If we can make a sustainability purpose explicit in a system, then the difference is significant. If such a purpose is not given, secondary influence can be achieved by adding sustainability objectives. The latter has less impact on the environment but is still feasible, the more the bigger the user community of a system. In the long run, our hypothesis is that we will not be able to end resource depletion by greening existing systems but only by disruptive change and completely transforming our systems [34]. Creating the mindset for that starts with acknowledging the need for incorporating sustainability as an explicit objective in systems development.

Another way to ensure prioritizing environmental sustainability is to enforce policies based on the compliance-driven economy. One open issue is the standardization of (environmental and general) sustainability as explicit quality objective in software development, for example within the IEEE 830 Recommendation for Software Requirements Specifications and the ISO 25000 on software quality, informed by the ISO standard families on environmental management [24] and social responsibility [25]. This contribution is published in [58], see Chapter 12.

**Sustainability as a Quality**

After dealing with sustainability requirements and supporting their elicitation, analysis, and realization systematically as described above, yet missing was a quality model that helps assessing the sustainability of a software system during and after development. This was dealt with in three parts: We started with gathering lessons learned from safety and security [73], then built a non-functional requirements framework [77], and finally derived a quality model for the different aspects of sustainability [82].

**Lessons Learned from Safety and Security**

Decades ago, the discipline of software engineering faced and solved shortcomings in its processes by including safety and security as new qualities for software systems in response to the rising importance of these issues. They were introduced as qualities for software systems after the first safety hazards and
the first security threats occurred. Consequently, we can learn from these prior research efforts and apply lessons learned to this new context. Safety is “an emergent property that arises when the system components interact within an environment” [38, p. 67]. Whether the term “emergent” adequately expresses the characteristics can be discussed, but the important point is that there is a part that is only assessable while the system is in operation, and that is where some effects of violations of safety rules may be observed. In contrast to safety, sustainability but must be dealt with in two parts: there is an inherent part of sustainability that shows up in the 1st order and some 2nd order effects, and there is the part that manifests itself in 2nd and 3rd order effects. Security is the degree of resistance to, or protection from, harm to information (e.g., unauthorized access). A security policy is the set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information. A similar mechanism of policies and requirements is needed for supporting and enforcing environmental sustainability. Aiming for standardization of environmental sustainability to support specifying constraints, the next step is to extend software engineering standards. For example, the ISO 29148 on requirements engineering could explicitly include a section on sustainability in the software requirements specification document template.

Relationship There is a number of relations that can be seen between safety, security, and sustainability. Safety is part of individual and social sustainability for preserving human life (no injuries) and environmental (no chemical or other hazardous accidents), but also has aspects in economic sustainability (a product that is not safe will not let a company reach long-term economic goals). Security is also part of various dimensions, the technical one as it is a standard quality attribute for systems, then individual and social (as the users shall be protected), and as a consequence of that also the economic dimension (insecure systems will not have market success). This makes clear that the relation between those qualities has to be evaluated for the project-specific understanding of domain-dependent sustainability. Privacy might serve as a more specific example to illustrate that juxtaposing quality characteristics is non-trivial. On one hand the absolute availability of information would simplify domain sustainability in many areas, but that contradicts privacy. For example if companies had to provide all their information on manufacturing and organization processes to demonstrate how environmentally sustainable they really are. And on the other hand, privacy can be seen as a part of individual human sustainability, for example concerning privacy breaches in online banking. Again, it depends on the definition of the domain-dependent sustainability and, consequently, on which values that definition is built on.

Lessons Learned Three of the lessons learned from safety and security are about goals conflicts, isolation, and solution-driven engineering.

- **Goal Conflict:** In traditional qualities considered during software engineering we already face a number of potential conflicts, for example between code maintenance and code performance, or between the development time and the desired quality of a software system. Consequently, the question arises what kinds of conflicts exists between the five dimensions of sustainability and their related goals. Leveson stated “A classic myth is that safety conflicts with achieving other goals and that tradeoffs are necessary to prevent losses.” [38, p.416] Similarly, there is a misperception that sustainability will conflict with a company’s economic goals, because companies do not have to pay for the emissions they produce. If, on the other hand, companies had to include these expenses, sustainability would form a natural part of their economic bottom line.

The economic dimension aligns with the environmental one in terms of resource savings (energy,
materials, waste), but they may conflict when it comes to additional certifications, building a (environmentally and socially) sustainable supply chain, and turning to more expensive alternative solutions in case they are more environmentally friendly. The reason for that is mainly that up to now, the negative environmental impacts that are caused by our economy are hardly charged. Therefore, the goal of environmental sustainability does not get assigned monetary value but only image value, which is likely to be ranked secondly. These conflicts are also discussed in [63]. Another potential conflict, at least for some systems, is a trade-off between energy efficiency and dangerous materials. This is one potential goal conflict in case energy efficiency would require using more dangerous material. Although not a software system in itself, a lightbulb might serve as example: New energy-saving lamps are much more energy-efficient than the old light bulbs, but at the same time contain toxic mercury that imposes a threat when a lamp breaks as well as phenol, naphthalene and styrene. In the case at hand, considerate users will make sure the lamp is not in close proximity to their heads, but as legislation has banned the old lightbulbs already in some countries, they will have to be used for now. Resolving such a conflict for a particular case means to assign weights to each of the goals and prioritize whether the energy saving is greater or whether the risk and long-term negative impacts of the dangerous materials are greater.

**Isolation:** A second common misperception is that safety, security, and sustainability can be dealt with in isolation of other quality characteristics. In isolation, requirements are replaced by solution-specific constraints, or the scope is simply set too small so that not all related aspects are taken into account and the result is a suboptimal solution. However, the description of the relation between safety, security and sustainability as well as the characterization of the juxtaposed relationship between privacy and sustainability shows that isolation is not an option as the characteristics are entangled. Instead, there has to be a systematic approach to define a quality model that includes a sustainability profile according to the respective understanding of domain-dependent sustainability.

**Solution-driven Engineering:** Firesmith stated “Security requirements are often solution mechanisms in disguise” [16]—and therefore “The user logs into the system with his security code.” is not a good security requirement. This leads to suboptimal solutions as a solution is already prescribed instead of analyzing the problem space and separating the solution finding. The same holds for sustainability requirements—e.g. when energy-saving mechanisms are applied instead of questioning and analyzing the process that the mechanism is required for. One way to ensure prioritizing environmental sustainability is to enforce policies based on the compliance-driven economy. If sustainability policies and standards are put in place, and software engineers prioritize them in the systems they develop, future technology may significantly contribute indirectly to influencing the behavior of users who interact with those systems as well as directly to saving the planet [73].

The challenge of incorporating sustainability into software engineering is an issue of requirements prioritization, which is usually handled by negotiation between system stakeholders. Perhaps the solution requires an explicit stakeholder for environmental sustainability. Another way to ensure prioritizing environmental sustainability is to enforce policies based on the compliance-driven economy. The details of this contribution are published in [73], see Chapter 14.

**Sustainability Requirements Framework**
To evaluate how well sustainability could be dealt with in a requirements framework, we adapted Glinz’ faceted classification [17] in a combination with sustainability models [77], as depicted in Table 1.5.2. It describes the different subcategories for sustainability requirements. It combines the sustainability dimensions and orders of effects in order to create fields that allow for breadth in coverage of sustainability topics as well as cater to the temporal scales within sustainability. We can also use the orders of effect to temporally scope a requirement when applicable. The primary challenge, for sustainability goals as with many other software quality attributes, lies in operationalization, validation and verification.

Table 1.5.2 illustrates the framework with examples from a Hotel Resource Tracking System (HRTS). There are two components of the HRTS: The Management Subsystem (MGMT), which is available to the the management team, contains resource consumption detail of every room in the hotel. The Guest Subsystem (GUEST) is available to each guest of the hotel and details resource consumption of the guest during their visit to the hotel and allows for guests to make sustainable decisions regarding their stay. The examples show that from short-term (first order) to long-term (third order) effects, there are aspects in every dimension of sustainability that are relevant for the system. For more specificity, they are also classified according to their representation, type of satisfaction, kind, and role.

Table 1.3: Sustainability Requirements Framework

<table>
<thead>
<tr>
<th>Orders of Effect</th>
<th>Environmental</th>
<th>Human</th>
<th>Social</th>
<th>Economic</th>
<th>Technical</th>
</tr>
</thead>
<tbody>
<tr>
<td>First:</td>
<td>The HRTS system shall track resource consumption in the hotel. Resources to be tracked are: energy, water, food, linens, and stationery. Facets: Quantitative representation, Hard satisfaction, Date kind, Prescriptive role</td>
<td>The MGMT subsystem shall provide data about resource consumption averages per guest and enable rewards to be awarded to guests with below average resource footprints. Facets: Operational representation, Hard satisfaction, Function kind, Prescriptive role</td>
<td>The system shall encourage participation as a community in greener living practices by providing a means for management and guests to interact. Facets: Quantitative representation, Soft satisfaction, Specific Quality kind, Prescriptive role</td>
<td>The MGMT subsystem shall provide inventory restocking recommendations for resources (only stationary, linens, food). Facets: Quantitative representation, Hard satisfaction, Date kind, Prescriptive role</td>
<td>The system shall be energy efficient. Facets: Quantitative representation, Soft satisfaction, Performance kind, Prescriptive role</td>
</tr>
<tr>
<td>Second:</td>
<td>The system shall track hotel wide resource inputs and outputs that will be stored in an external resource tracking database in a standardized format. Facets: Quantitative representation, Hard satisfaction, Date kind, Prescriptive role</td>
<td>The GUEST subsystem shall provide hotel guests with a useful interface that displays their resource footprint and impacts of consuming different amounts of resources. Facets: Qualitative representation, Soft satisfaction, Specific Quality (usability) kind, Assumptive role</td>
<td>The system shall provide managers with contextual information regarding overall hotel resource consumption that displays the hotels adherence to existing sustainability and corporate responsibility practices. Facets: Declorative representation, Soft satisfaction, Date kind, Prescriptive role</td>
<td>The MGMT system shall identify and report high resource consuming areas of the hotel. Facets: Declorative representation, Soft satisfaction, Function kind, Prescriptive role</td>
<td>The system shall require minimal specialized hardware. Facets: Declorative representation, Soft satisfaction, Constraint kind, Prescriptive role</td>
</tr>
<tr>
<td>Third:</td>
<td>The system is envisioned to become the quasi standard for HRMS by providing a good example in reducing negative environmental impact. Facets: Declorative representation, Soft satisfaction, Specific Quality kind, Assumptive role</td>
<td>The GUEST subsystem shall provide hotel guests with persistent historical resource consumption data across all their visits to the hotel. Facets: Operational representation, Hard satisfaction, Date kind, Prescriptive role</td>
<td>The system shall provide the option for a guest to log into their “Hotel World” account and save the data of that particular stay to their account. Facets: Declarative representation, Hard satisfaction, Function kind, Prescriptive role</td>
<td>The system shall enable financial cost cutting due to reduced resource consumption in hotels. Facets: Qualitative representation, Soft satisfaction, Specific Quality kind, Prescriptive role</td>
<td>The system shall support easy data exportation. Facets: Operational representation, Soft satisfaction, Date kind, Prescriptive role</td>
</tr>
</tbody>
</table>

While we believe that creating theoretical frameworks for sustainability is an achievable goal, a major challenge will be encouraging the adoption of sustainability in requirements and software engineering. We identified three criteria that have an impact on how successful other software qualities are—specificity, visibility, and support. Specificity is necessary to decompose and refine requirements such that they can be realized. Visibility is necessary so the customer is aware of this quality. And support is required to enable developers to implement requirements related to a specific quality. By providing means for each of these three criteria, sustainability can successfully be integrated as a major software quality. This contribution is published in [77], see Chapter 13.

Quality Model

Finally, we developed a quality model for sustainability that provides a tailorable reference model for the five dimensions (Environmental, Economic, Individual, Social, and Technical) to be applied by software development companies to their organization processes as well as their systems under development [82]. The quality model uses goals, tasks, and indicators as conceptual elements and is based on, inter alia, the ISO 25010 on Systems and Software Quality Models, ISO 14000 on Environmental Management, ISO 26000 on Social Responsibility, COCOMO 2.0 [11], and Quamoco [88]. The detailed models are too large to show in this report, but are available in [82].

The conclusion from that part of the work is that a reference model including criteria from all of those standards on all the dimensions of sustainability is very large and at the same time very abstract. The ISO standards on Environmental Management and on Social Responsibility are on a very high level of abstraction compared to COCOMO and Quamoco. The reason for that is that the ISO standards are on the abstraction level of domain-dependent sustainability whereas the other models are on the more technical level of domain-independent sustainability. Consequently, a reference model that includes such a wide range of abstraction may be encompassing but is rather limited in its practicability.

Instead, we took a different approach to framing sustainability as a software quality property in [37]. The sustainability analysis framework, depicted in Fig. 1.7 extends the existing Third Working Draft of ISO/IEC 42030 Architecture Evaluation [27]. Gray boxes denote generalized pre-existing components from the working draft. While the ISO draft specifically targets evaluations, the potential context of our framework is broader, embracing any activity that relies on a sound representation of qualities, including requirements engineering, design-decision making, trade-off analyses, and quality assessments. We do not focus on any specific context, but rather illustrate how the framework can be used to frame sustainability quality requirements and concerns.

As shown in Fig. 1.7, an Evaluation Criterion can be a quality requirement. In particular, as we focus on characterizing the sustainability-related software qualities, we need to address how quality requirements relate to sustainability—Sustainability Quality Requirements. In this context, they may include both traditional quality requirements (e.g., performance, usability, security, or maintainability) and sustainability related ones (e.g., energy efficiency). Moreover, whenever we specifically target sustainability (cf. Fig. 1.7 where the association ‘aims at’ links the Evaluation Objective to the Sustainability Dimension), we must perform trade-offs among the various qualities classified as belonging to each of the four dimensions. In particular, we observe that traditional software decision-making considers trade-offs either between different technical sustainability criteria (e.g., performance versus availability), or between
technical sustainability criteria and economic sustainability criteria (e.g., performance versus costs). In contrast, sustainability-related software decision-making involves trade-offs between environmental sustainability criteria (e.g., energy efficiency) and social, economic, and technical sustainability criteria. The application of our framework in practice enables business software developers to specifically consider the neglected environmental and social dimensions in relation with the technical and economic dimensions. This contribution is published in [37], see Chapter 15.

1.5.3 WP 3: Feasibility Studies

In our conducted case studies, we collaborated with partners from industry (BMW AG, jambit GmbH), academic projects (Plant Guild Composer), hypothetical studies (Facebook), academic spin-offs turning into start-ups (Calico), non-profit organizations (Story of Stuff), and academic development projects (Plant Guild Composer).

- The DriveNow Car Sharing System: The DriveNow system by BMW, Sixt and Mini served as feasibility study for the reference model for decomposing sustainability.

- The jambit GmbH: This Software Development Company served as feasibility study that the reference model is also beneficial when applied to development processes.

- The Plant Guild Composer tool: This garden design software was used to demonstrate the feasibility of the analysis approach and artifact model for a system dedicated to environmental sustainability.

- The Calico tool: This collaborative drawing tool for software engineers was used as second feasibility study for the analysis approach and artifact model for a system with no specific sustainability purpose.

- Facebook: The online social network service was used as hypothetical study to show what difference it could have made to include a broader understanding of sustainability from the start of development.
- **The Citizen Muscle Bootcamp**: This online course on civic engagement by the Story of Stuff Project was used as third feasibility study for the analysis approach and artifact model for a system dedicated to social and environmental sustainability.

- **The Cognatio tool**: This application strives to improve the communication between patient and doctor in tracking progress and effects of medication served as final feasibility study for a system dedicated to individual and social sustainability.

The method included interviews and a domain analysis as a foundation, and then requirements elicitation and conceptual modeling with a sustainability analysis. The goal was to show how enriching the requirements engineering process with sustainability considerations might improve the sustainability of the resulting software system (by using the approach described in Sec. 1.5.2 and in [57, 58]) and to further improve our modeling concepts and guidelines by the elicited feedback from the partners.

**Car Sharing System**

The DriveNow system (http://www.drive-now.com/), an electric car rental and sharing system operated in collaboration by BMW, Sixt, and Mini, was analyzed by Krasnov and Penzenstadler [33] with regard to sustainability aspects in the business context, by Feldmann and Penzenstadler [15] with regard to supporting sustainability in the usage processes, as well as by Rodriguez and Penzenstadler [78, 79] with regard to future scenarios using the IMAGINE [9] technique. These studies showed that the sustainability goal model (Sec. 1.5.2) provides the benefit of being able to discover additional requirements when used for software systems. For example, the sustainability analysis showed the benefits of extending the service to rural areas combined with the option to receive bonus points when returning the car to higher traffic areas.

**Software Development Company**

The jambit GmbH is a small-to-medium sized software engineering and consulting company with about 120 employees. In most of their projects, they work according to the Rational Unified Process and, according to their own statements, they care about the environment and are willing to make an effort at improving their own footprint. In [32], we show how the goal model (Sec. 1.5.2) provides benefits by optimizing environmental impact when used for development and organizational processes in software companies. In addition, the model also pointed out possible further improvements for social sustainability in the company.

**Plant Guild Composer**

In permaculture, a domestic plant guild can foster human independence from extraneous materials. A domestic plant guild is a family of plants that can sustain itself and provide people with many of their essential needs (e.g., food, building materials, etc.) [49]. It is an enabler of simple, sustainable living, i.e., reduces cost of living, ecological footprint, and the need for consumer goods. The design and construction of a plant guild requires time and expert knowledge, two factors that prevent many from incorporating one. Therefore, support tools make it easier for people to develop and establish domestic plant guilds. The requirements for such a tool are explored in [52] using the sustainability approach.
described in Sec. 1.5.2. This study shows benefits of our approach for sustainability application domains like permaculture in terms of structuredness and completeness in requirements elicitation.

**Collaborative Software Tool**

Yet another application domain is software tools for software developers, for example Calico, a collaborative drawing tool for software engineers, developed at the University of California, Irvine [41]. In the study reported in [2], we explore how reengineering the requirements for Calico could have an impact on the sustainability effects of Calico on the environment and its users by modeling the artifacts described in Sec. 1.5.2 and discussing newly discovered requirements and constraints. For example, by considering the environmental sustainability dimension, energy-savings mechanisms could be added. By considering social sustainability, mechanisms for better connecting the collaborators could be evaluated. Finally, there had not been a structured requirements specification for Calico before, so technical sustainability was also improved by elaborating such a specification. However, the conclusion is that for a system with little direct impact on the application environment, i.e. that triggers few effects outside the system itself, the improvement is limited. The reason for that is that the domain-dependent understanding of sustainability in the context of Calico is mainly focused on technical sustainability.

**Analysis of an Online Social Network**

How would it change the requirements of a software system without a specific sustainability purpose if sustainability had been considered all along the way? The online social networking service Facebook was taken as example to illustrate this. Initial sustainability goals that could be identified were:

- **Business goals:** Long-term evolution (economic and technical sustainability), ROI (economic sustainability), Large marketshare (economic sustainability)
- **Usage goals:** Connect people (social sustainability), Share content (individual and social sustainability), Trigger communication (individual and social sustainability), Spread news (individual, social, economic sustainability)
- **System goals:** High availability (individual, social, economic, and technical), High reliability (individual, social, economic, and technical), Long-term maintenance (social, economic, technical)

The analysis led to the conclusion that including sustainability from the start would have meant two changes:

- The user interface would be simpler to require less energy and thereby run well on older devices (environmental and technical sustainability).
- The privacy policy and settings would have been available from the start as opposed to years later due to user complaints (individual and social sustainability).

Furthermore, it would be possible to enhance the system with some more options to use it for purposes towards sustainability, but as that is not their initial system vision, this remains the choice of the user. The full details are provided in [57].
Online Course on Civic Engagement

For software systems as mediators of and educators for sustainability, we performed a study on the online course system on civic engagement, called the Citizen Muscle Bootcamp (CMB), by the non-profit organization The Story of Stuff Project. The domain-dependent understanding of sustainability in this case is that the CMB is designed for improving the environmental and social sustainability of our society by educating people. To be able to fulfill this objective, the domain-dependent understanding of sustainability includes that the system itself needs to be technically and socially sustainable. We redesigned the CMB course in collaboration with the Story of Stuff after its first prototype had only been little successful. The requirements were elaborated using our approach (Sec. 1.5.2), this time performed by a larger group of 100 students in teams of 5 as class project within a requirements engineering seminar, reported on in [74] (see Chapter 23). Specifically for improving the social and technical sustainability of the system, in [30], we describe how the online course will be evaluated with regard to participant engagement and retention to inform the development of future versions of the course.

Healthcare and Telemedicine

As a final evaluation study, we chose yet another application domain that is likely to increase in the future and has significant impact on the sustainability of individuals as well as society as a whole: the healthcare system. Aligned with the common objective of slowly transforming our current system of “sick care” into a system of actual “health care”, telemedicine and enhanced communication with physicians are crucial factors. In [46], we explore such a system, i.e. an app for dermatology that serves as information gathering device for a patient (by tracking the development of a mole or other skin spot and the prescribed medication) and additional information source and monitoring device for the treating physician. The domain-dependent understanding of sustainability is that Cognatio is designed for improving the individual sustainability of the patient and the social and economic sustainability of the healthcare system. Also in this case, our approach helped to discover additional requirements to enhance these sustainability aspects, for example by facilitated communication between patient and doctor.

1.6 Conclusion

In the following, we summarize our conclusions, the impacts and implications, the limitations, and future work.

1.6.1 Summary of Conclusions

In this habilitation, we set out to explore what sustainability means with regard to software engineering and how it can be supported specifically already in requirements engineering. This chapter provided an overview of the habilitation contributions to the research questions on software engineering for sustainability.

What does sustainability mean with regard to software systems and how can it be understood for the different abstraction levels concerned during requirements engineering? To answer this question, we conducted two systematic literature studies, iteratively elaborated a characterization of
software engineering for sustainability, and analyzed the affected stakeholders and explored their related
values.
In summary, sustainability with regard to software systems can only be understood in the context of the
specific application domain and context. This domain-dependent understanding (higher abstraction level)
can then be broken down and translated into a domain-independent understanding (on a lower abstraction
level) that can be realized in the specific software system. The responsibility of the requirements engineer
is to facilitate this translation.

Who are the stakeholders that advocate sustainability? How can stakeholders be classified with
respect to their interest in sustainability? To answer this question, we performed an analysis of differ-
ent stakeholder elicitation techniques in software engineering and of stakeholder reference models. We
developed a reference list of stakeholders impacted by the different dimensions of sustainability. Fur-
thermore, we developed a value map with a list of values for each sustainability dimension that helps
guiding decision-making in software product management.

How can sustainability in its different forms be emphasized in a requirements engineering ap-
proach for information systems? To answer this question, we elaborated a number of concepts and
models based on the insights acquired from the first research question. We finally evaluated the feasi-
bility of these concepts in a number of studies that explored the applicability and usefulness in different
application domains.
In summary, the proposed concepts to support an emphasis of sustainability in requirements engineering
are a stakeholder analysis, a value map, a sustainability reference model that can be instantiated as goal
model for a specific system, a requirements analysis approach with documentation artifacts, and concepts
for the inclusion in a quality modeling approach.
The major conclusions we draw from our work are:

- Sustainability has to be understood within the specific context a system under development resides
  in.
- Requirements engineers can facilitate the translation from high-level sustainability goals into system-
specific requirements and constraints using the concepts proposed in this work.
- Sustainability in its different forms can be improved for the systems under consideration, but the
  prioritization of sustainability concerns remains with the stakeholders and depends on their values.

1.6.2 Impact and Implications
The impact of our contribution is mainly determined by the question how much difference the consid-
eration of sustainability actually makes in requirements engineering. If we can make a sustainability
purpose explicit in a system, then the difference is significant. If such a purpose is not given, secondary
influence can be achieved by adding sustainability objectives and optimizing the system. The latter has
less impact on the environment but is still feasible, the more the bigger the user community of a system.
In the long run, our hypothesis is that we will not be able to end resource depletion by optimizing existing
systems but only by disruptive change and transforming our systems. However, creating the mindset for
that starts with acknowledging the need for incorporating sustainability as explicit quality objective in systems development.

For requirements engineers, the implication of the presented work is an opportunity to be able to better support a characteristic for software systems in their application domain that has continuously been gaining more attention over the last years. The concepts enable them to react to shifted priorities and align their development activities accordingly. Most importantly, the concepts don’t require to adopt a specific approach to requirements engineering, but allow to consult a number of reference models and checklists for better guidance during elicitation and analysis activities.

1.6.3 Limitations

The limitations of the work at hand are the breadth of the objective, the reliability of data, the differentiation from values, and the measurability of the impact.

Breadth of the Objective

This habilitation investigated a broad understanding of sustainability in relation to software engineering and analysis of what all those aspects could be and what their impact could be. Consequently, less depth was dedicated to a specific type of software systems. Looking at different application domains and types of software system allowed for a better overall understanding of the diverse understandings of sustainability.

Reliability of Data

For the literature studies, there are a number of limitations that are discussed in depth in the respective publications, for example the database reliability and the completeness of results. In summary, they are data collections at specific points in time in an emerging research field, which is why they should be updated in frequent intervals.

For the conducted interviews, the gathered amount of data was too small to quantitatively draw conclusions. Therefore, interviews were only used as qualitative data source to serve as additional input for conceptual work and to informally gather feedback on intermediate results.

For the feasibility studies, the number of involved stakeholders per system was limited to a few external stakeholders who committed time to support elicitation of the requirements and to provide feedback on the results. Due to the context the feasibility studies were conducted in — different application domains and different system types — we achieved a qualitative evaluation of the concept. That qualitative data indicates the usefulness of the proposed concepts and helps to gain a better understanding of the challenges in practice in a specific project context.
Differentiation from Values

Sustainability is strongly tied in with values. Consequently, if there is no one amongst the stakeholders for a software system who puts emphasis on sustainability, then it will not be designed into the system. The important point here is that a requirements engineer fulfills tasks that they get assigned. The discussion of whether the values behind the understanding of domain-dependent sustainability of their clients is not a software engineering discussion. Nevertheless, requirements engineers should follow a code of ethics (as, for example, proposed by the ACM [1]) in their work.

Measurable Impact of Concepts

The feedback that was gathered in the feasibility studies is limited to general notions like new requirements being discovered and further aspects in the problem space exploration opening up. Consequently, we could show differences in the resulting specifications when explicitly considering sustainability during requirements engineering. As research community, we are currently not capable of measuring such complex circumstances in a system of variables for gathering empirical evaluation statistics. Conducting controlled experiments with a comparative case study in value-based software engineering, where one team develops with sustainability being an emphasized objective and another team developing the system without such an objective, would only make sense if it was possible to specify such a system of variables.

A qualitative version of such a comparative approach was used when re-specifying requirements for existing systems while considering sustainability, and this led to the observations of discovering additional requirements and exploring the problem space more extensively.

1.6.4 Future Work

The future work we envision to carry out within the next years includes topics related to cost modeling, legal constraints, and tooling.

Cost Modeling

An aspect worth discussing is the connection between stakeholders, goals, and cost modeling. The stakeholders are made explicit in the sustainability reference model by tracing back to the rationale of a goal, as the information source (e.g. a domain expert) or the issuer of a goal. With respect to assigning costs to the goals there is a limitation, as this only makes sense for business goals, but not for values that cannot be expressed in return on investment. Some goals, for example the protection of the environment, do not have monetary value in themselves and their qualitative value is hard to measure. At the same time, it is important to define measures to ensure the realization of these goals and to show that the approach can make a difference in those resulting measures.

Consequently, instead of assigning costs to the sustainability goals, their contribution to higher causes must be made explicit, for example the contribution to objectives commonly agreed on by governments like the sustainable development goals from Rio+202, or the Vision 2050 [90]. Currently, there is no system of variables available to assign costs to such values and it is questionable if there ever will be.

2http://www.uncsd2012.org/
With regard to the extra effort spent on the RE4S activities during requirements engineering, this is limited to consulting a few reference models on the way, which adds only small amounts of time to the elicitation and analysis activities.

**Legal Constraints**

As a consequence of the fact that environmental goals have not yet been prioritized sufficiently by the economy, legislation has established a number of environmental regulations that companies have to adhere to. These regulations will still be extended in the future, which makes legislation probably the most important stakeholder representing environmental sustainability in particular. Individual and social sustainability are also taken care of by law, for example by worker’s rights, which are supported and represented by worker unions.

At which point we need new laws and a different legislation to make sure that important questions of sustainability are incorporated into IT systems is an open issue. Furthermore, it would be interesting to look at other examples such as functional safety and also to a certain extent security, where such laws exist. To orchestrate that, a collaboration with an environment specialist (for example, from earth system sciences) and a law specialist (for example, for environmental regulations) needs to be set up. This will allow to analyze a set of cases and explore the options for where influence by legislation is advisable.

**Tooling**

One aspect that is relevant for any proposed concept is to provide adequate tooling to disseminate them and to facilitate their application in practice. This applies to the reference model for sustainability as well as to guidance for the overall RE4S approach. The tools developed within the scope of the habilitation are useful for initial dissemination, but for more wide-spread application in practice, improved versions have to be developed. We envision a web-based guidance that provides easy access to the reference models and related examples in form of a knowledge database. This knowledge base can be extended over time with further examples from case studies and with recommender systems that actively guide the user, i.e., the requirements engineer.
1.7 Integration into Teaching

Within this habilitation, we supervised two Bachelor’s theses, two Master’s theses, and three individual research studies.

- Alejandra Rodriguez, Technische Universität München, Guided Research, 2012
  An Assessment Technique for Sustainability: Applying the IMAGINE Approach to Software Systems [78, 79]

- Ilya Krasnov, Technische Universität München, BSc Thesis, 2013
  Artefacts and Techniques to Support Environmental Sustainability in Specifying a Car Sharing Platform [33]

- Susanne Klein, Technische Universität München, MSc Thesis, 2013
  Instantiating a Generic Sustainability Goal Model for a Software Development Company [32]

- Oliver Feldmann, Technische Universität München, BSc Thesis, 2013
  Sustainability Aspects in Specifying a Car Sharing Platform [33]

- Anshu Singh, University of California, Irvine, MSc Thesis, 2014
  Quality Models for Sustainability [82]

- Chris Arciniega, University of California, Irvine, Individual research project, 2014
  Green Calico—Reengineering Requirements for Sustainability for a Collaborative Drawing Tool [2]

- Joseph Mehrabi, University of California, Irvine, Individual research project, 2014
  Cognatio—An app for supporting doctor-patient communication for skin treatment [46]

Furthermore, we developed a seminar series on sustainability in informatics [66, 61, 59] and integrated sustainability considerations into requirements engineering lectures as reported on in [69, 74], also by choosing team development projects that had a focus on environmental causes. Both the seminar series and the additional session on sustainability in the requirements engineering class were well received by the students.

The starting point was a conceptual outline of how to integrate the topic of sustainability into teaching informatics, especially software engineering [66]. In 2011, we started this with a Bachelor’s seminar on Sustainability in Informatics at TUM and a Master’s seminar on Requirements Engineering for Green IT-Systems at the Universitat Polytecnica de Catalunya.

Finally, we integrated the concepts into courses on requirements engineering by means of student team projects. This was first done at TUM in a collaboration with BMW on the DriveNow car sharing system [69], and later on at the University of California, Irvine, in a collaboration with the Story of Stuff Project on their online course system for civic engagement [74].

3 http://www4.in.tum.de/lehre/seminare/SS11/sustainability/
4 http://www4.in.tum.de/lehre/seminare/WS1112/RE4Green/
5 http://www4.in.tum.de/lehre/seminare/WS1112/UPC-EnviroSiSE/
6 http://www.drive-now.com/
7 http://storyofstuff.com/
1.8 Overview of Publications

Apart from the key publications in peer-reviewed proceedings and journals, I published grey literature, e.g., technical reports, short (position) papers and a not peer-reviewed book chapter. Table 1.4 provides an overview of all contributions published within this habilitation project.

For reasons of completeness, I added them as respective second reference to the publication that they are related to. All contributions are published except one workshop paper which was under review at the time of writing the essay. The publications are to be found in the later chapters of the habilitation thesis in the order presented in this table.

Furthermore, Table 1.5 provides an overview of how the results and publications described above map to the deliverables listed in the goal definition.
<table>
<thead>
<tr>
<th>Foundations</th>
<th>Conference</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sustainability in Software Engineering: A Systematic Literature Review</td>
<td>EASE'12 [60, 54]</td>
<td></td>
</tr>
<tr>
<td>Systematic Mapping Study on Software Engineering for Sustainability (SE4S)</td>
<td>EASE’14 [71, 72]</td>
<td></td>
</tr>
<tr>
<td>Towards a Definition of Sustainability in and for Software Engineering</td>
<td>SAC’13 [55, 56]</td>
<td></td>
</tr>
<tr>
<td>ICT4S 2029: What will be the Systems Supporting Sustainability in 15 Years?</td>
<td>ICT4S’14 [75]</td>
<td></td>
</tr>
<tr>
<td>Sustainability Design and Software: The Karlskrona Manifesto</td>
<td>ICSE’15 [8]</td>
<td></td>
</tr>
<tr>
<td>Who is the advocate? Stakeholders for Sustainability</td>
<td>GREENS’13 [65]</td>
<td></td>
</tr>
<tr>
<td>Towards Incorporating Sustainability while Taking Software Product Management Decisions</td>
<td>IWSPM’13 [67, 68]</td>
<td></td>
</tr>
<tr>
<td>Concepts</td>
<td>Workshop</td>
<td></td>
</tr>
<tr>
<td>Supporting Sustainability Aspects in Software Engineering</td>
<td>CompSust’12 [53]</td>
<td></td>
</tr>
<tr>
<td>RE4ES: Support Environmental Sustainability by Requirements Engineering</td>
<td>Workshop</td>
<td></td>
</tr>
<tr>
<td>Infusing Green Requirements Engineering for Green in and through software systems</td>
<td>RE4SuSy’14 [58, 57]</td>
<td></td>
</tr>
<tr>
<td>Developing a Sustainability Non-Functional Requirements Framework</td>
<td>Workshop</td>
<td></td>
</tr>
<tr>
<td>Safety, Security, ... now Sustainability!</td>
<td>Journal</td>
<td></td>
</tr>
<tr>
<td>Framing Sustainability as a Software Quality Property</td>
<td>Journal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACM’15 [37]</td>
<td></td>
</tr>
<tr>
<td>Evaluation</td>
<td>Workshop</td>
<td></td>
</tr>
<tr>
<td>RE@21: Time to sustain</td>
<td>RE4SuSy’13 [64]</td>
<td></td>
</tr>
<tr>
<td>Domestic Plant Guilds: A Software System for Sustainability</td>
<td>Workshop</td>
<td></td>
</tr>
<tr>
<td>An Assessment Technique for Sustainability: Applying the IMAGINE Approach to Software Systems</td>
<td>RE4SuSy’13 [79, 78]</td>
<td></td>
</tr>
<tr>
<td>Supporting Physicians by RE4S: Evaluating Requirements Engineering for Sustainability in the Medical Domain [47, 46]</td>
<td>Workshop GREENS’15</td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td>Conference</td>
<td></td>
</tr>
<tr>
<td>Teach Sustainability in SE?</td>
<td>CSEE’11 [66]</td>
<td></td>
</tr>
<tr>
<td>Jumpstart Sustainability in Seminars</td>
<td>Conference</td>
<td></td>
</tr>
<tr>
<td>Case Study-based RE Education: Evaluation and Lessons Learnt</td>
<td>CSEERC’12 [59]</td>
<td></td>
</tr>
<tr>
<td>Using Non-Profit Partners to Engage Students in RE</td>
<td>Workshop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REET’14 [74, 30]</td>
<td></td>
</tr>
</tbody>
</table>
### Table 1.5: Mapping of publications to deliverables

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systematic Literature Studies</td>
<td>[60, 71]</td>
</tr>
<tr>
<td>Characterization of SE4S</td>
<td>[56, 55, 73, 8]</td>
</tr>
<tr>
<td>Stakeholder Analysis</td>
<td>[69, 65]</td>
</tr>
<tr>
<td>Value Map</td>
<td>[67]</td>
</tr>
<tr>
<td>Sustainability Reference Model</td>
<td>[63, 64]</td>
</tr>
<tr>
<td>Artifact Model</td>
<td>[58, 69]</td>
</tr>
<tr>
<td>Analysis Approach</td>
<td>[76, 58, 79]</td>
</tr>
<tr>
<td>Quality Modeling</td>
<td>[77, 73, 37]</td>
</tr>
<tr>
<td>Feasibility Studies</td>
<td>[52, 30, 74, 47]</td>
</tr>
</tbody>
</table>
Part II

Foundations
The Foundations part contains the publications with regard to the systematic literature studies [60, 71], the characterization of software engineering for sustainability [55, 75, 8], and the stakeholder analysis [65] and value map [67]. These publications were partially developed in collaborations. The following list sets the context for the elaboration of each of the publications.

- **Sustainability in Software Engineering: A Systematic Literature Review [60]**
  This first literature study was performed in collaboration with Prof. Xavier Franch, Prof. Coral Calero, and Veronika Bauer. It was published as main conference paper at the 2012 edition of the International Conference on Evaluation and Assessment in Software Engineering. In addition, a Technical Report is available with the full details of the protocol [54].

- **Systematic Mapping Study on Software Engineering for Sustainability [71]**
  The second literature study was performed with Prof. Debra Richardson, Prof. Coral Calero, Prof. Xavier Franch, Ankita Raturi, and Henning Femmer. The contribution was published as main conference paper at the International Conference on Evaluation and Assessment in Software Engineering 2014. Again, a Technical Report is available with the full details of the protocol [72].

- **Towards a Definition of Sustainability in & for Software Engineering [55]**
  This work was single-authored and based on earlier work that had been published as poster abstract at the First International Conference on ICT for Sustainability in 2013 [56]. Feedback from that conference and further follow-up research lead to the publication of this contribution as a main conference paper at the 28th Annual ACM Symposium on Applied Computing in 2013.

- **ICT4S 2029: The Systems Supporting Sustainability in 15 Years? [75]**
  This work was a collaboration with many coauthors who contributed a fictional abstract on what the systems supporting sustainability in 15 years would be. I lead and coordinated the effort and authored the introduction, background, and synthesis and conclusion of the paper. The first five coauthors were involved in editing and feedback, while the later coauthors provided the remaining design fiction. The coauthors were Bill Tomlinson, Eric Baumer, Marcel Pufal, Ankita Raturi, Debra Richardson, Baki Cakici, Ruzanna Chitchyan, Georges Da Costa, Lynn Dombrowski, Malin Picha Edwardsson, Elina Eriksson, Xavier Franch, Gillian R. Hayes, Christina Herzog, Wolfgang Lohmann, Martin Mahaux, Alistair Mavin, Melissa Mazmanian, Sahand Nayebaziz, Juliet Norton, Daniel Pargman, Donald J. Patterson, Jean-Marc Pierson, Kristin Roher, M. Six Silberman, Kevin Simonson, Andrew W. Torrance, and Andre van der Hoek. This main conference paper was published at the Second International Conference on ICT for Sustainability in 2014 and received a nomination for the best paper award.

- **Sustainability Design and Software: The Karlskrona Manifesto [8]**
  This work originated in the 3rd International Workshop for Requirements Engineering for Sustainable Systems (RE4SuSy) which I organized at the 22nd International Conference on Requirements Engineering. In collaboration with Christoph Becker I prepared working sessions for the workshop on what we envisioned to become the foundation for a manifesto. During the workshop and the following days, a working group emerged that has been collaborating continuously since the workshop to author a manifesto on sustainability design. This effort lead to a main conference paper accepted for the Software Engineering In Society track at the International Conference on Software Engineering for Sustainability.
Who is the advocate? Stakeholders for Sustainability [65]
This work was a collaboration with Henning Femmer at TUM. The idea originated from interviews with different categories of stakeholders with varying interpretations of sustainability and lead to an analysis of stakeholder identification approaches and the deduction of a reference list. The paper was published at the 2nd International Workshop on Green and Sustainable Software.

Towards Incorporating Sustainability while Taking SPM Decisions [67]
This paper was elaborated in collaboration with Mahvish Khurum and Kai Petersen from the Blekinge Institute of technology. Mahvish Khurum published the Software Value Map as decision tool for software product managers and in this paper, we extended the SVM with the different dimensions of sustainability. The paper was published at the 7th International Workshop of Software Product Management in 2013 and received the best paper award.
2 Sustainability in Software Engineering: A Systematic Literature Review
Sustainability in Software Engineering: 
A Systematic Literature Review

Birgit Penzenstadler, Veronika Bauer
Technische Universität München, Germany
{penzenst|bauerv}@in.tum.de

Coral Calero
Universidad de Castilla - La Mancha, Spain
Coral.Calero@uclm.es

Xavier Franch
Universitat Politècnica de Catalunya, Spain
 franch@essi.upc.edu

Abstract—Background: Supporting sustainability in software engineering is becoming an active area of research. We want to contribute the first Systematic Literature Review (SLR) in this field to aid researchers who are motivated to contribute to that topic by providing a body of knowledge as starting point, because we know from own experience, this search can be tedious and time consuming.

Aim: We aim to provide an overview of different aspects of sustainability in software engineering research with regard to research activity, investigated topics, identified limitations, proposed approaches, used methods, available studies, and considered domains.

Method: The applied method is a SLR in five reliable and commonly-used databases according to the (quasi-standard) protocol by Kitchenham et al. [1]. We assessed the 100 first results of each database ordered by relevance with respect to the search query.

Results: Of 500 classified publications, we regard 96 as relevant for our research questions. We sketch a taxonomy of their topics and domains, and provide lists of used methods and proposed approaches. Most of the excluded publications were ruled out because of an unfitting usage of terms within the search query.

Conclusions: Currently, there is little research coverage on the different aspects of sustainability in software engineering while other disciplines are already more active. Future work includes extending the study by reviewing a higher number of publications, including dedicated journal and workshop searches, and snowballing.

I. MOTIVATION AND BACKGROUND

Sustainability is currently an omni-present term in calls for research proposals and conference sessions (ICSE, CAiSE, RE, etc.). However, in literature, there is no overview of the current state of the art in supporting sustainability in software engineering research and practice. Consequently, researchers who are motivated to contribute to that topic (like the first author [2]) have to invest much time in finding a basic body of knowledge through literature research of many unrelated leads.

This paper reports on our systematic literature review with the objective of retrieving a solid basis of knowledge on the support of sustainability in software engineering. The full protocol is available online as technical report [4].

A. Definition of Sustainability

To clarify our research objective, we define our understanding of sustainability and what we mean by sustainability and how we want to apply it to software engineering. The most cited definition of sustainable development [5] is to “meet the needs of the present without compromising the ability of future generations to satisfy their own needs.” According to [5], sustainable development needs to satisfy the requirements of the three dimensions of society, economy, and environment. A fourth dimension, human sustainability, is less present in the public discussion. According to [6], it should be included, as it is the basis for the other dimensions. All four dimensions of sustainability are further detailed on in our SLR protocol [4].

B. Sustainability Aspects in Software Engineering

Sustainability aspects can be brought to bear both during the development and use of software systems. We distinguish four aspects of sustainability in SE (orthogonal to the dimensions introduced in Sec. I-A). The development process viewpoint includes:

- Development process aspect: Sustainability in the initial system development process (with responsible use of ecological, human, and financial resources). This aspect focuses on the initial conceptual and constructional development and we distinguish it from the late phase of actual system production for reasons of analysis.
- Maintenance process aspect: Sustainability of the software system during its maintenance period until replacement by a new system (with continuous monitoring of quality, knowledge management).

The product viewpoint encompasses the aspects of sustainability during production and usage:

- System production aspect: Sustainability of the software system as product with respect to its use of resources for production (using green IT principles and sustainably produced hardware components). The actual system production happens after most of the initial development process and considers, inter alia, mass production aspects, logistics and factory organization issues.
• **System usage aspect:** Sustainability in the usage processes in the application domain triggered by the software system as product (responsible in impact on environment, using green business processes).

We expect these aspects to have different scales of impact, growing from small to large in the order presented above, so that the system usage aspect potentially has the biggest impact (and, therefore, improvement potential). However, this is also dependent of the system under analysis.

For our SLR, we are looking for all four aspects of sustainability in software engineering. The aspects imply different levels of abstraction and varied granularity, but nevertheless we are interested in the state of research for each of them.

### C. A Body of Knowledge for Sustainability in SE

Our research aim for the next years is to support the development of ICT systems for environmental sustainability (ICT4ES) with an adequate software engineering approach that integrates the knowledge of related disciplines that are concerned with sustainability. For that we need to build up on existing knowledge is SE as well as disciplines that have been related closer to sustainability, for example, environmental informatics.

This research aim requires accumulating a body of knowledge for various reasons: justifying the basis for future research, learning as much as possible from other domains related to the topic, and providing a basis for other researchers as well as students who consider learning about and contributing to this area. One commonly accepted research method for accumulating a body of knowledge is a study in form of a systematic literature review [3].

### D. Research Questions

The overall research objective of the study is to find out what the current state of the art in supporting sustainability in software engineering research and practice is. This is further detailed in the following research questions:

- RQ1: How much activity was there in the last 20 years?\(^2\)
- RQ2: What research topics are being addressed?
- RQ3: What are the limitations of current research?
- RQ4: How is sustainability support performed?
- RQ5: Which methods are in use?
- RQ6: Are there case studies available?
- RQ7: Which domains are already considered?

### E. Related Work

There are systematic literature reviews on different topics in software engineering, but so far none has been conducted that investigates the relation between sustainability and software engineering.

\(^2\)Our hypothesis is that most publications will be much younger, so a time span of 20 years ensures that we include all relevant ones.

Mahaux et al. [7] performed a preliminary search on the DBLP Computer Science Bibliography database\(^3\). For articles with the prefixes “sustainab-” OR “ecolog-” OR “environmental-” in the title, the database returned over 3000 results (in January 2010), but filtering on important software-related venues lead to as few as 11 results. They propose that a systematic literature review should be conducted.

In contrast to [7], we are interested in publications from all scientifically sound venues and journals as we see great potential for learning from other domains. Therefore, we did not restrict this systematic literature review to software-related venues, which is the main reason why we received more results.

### II. Search Design and Process

The search design and procedure follow the guidelines in [1]. As SLR research questions we directly adopted those enumerated in Sec. I-D. The search process for this study is based on an automated search of the following digital libraries:

- ACM Digital Library [http://dl.acm.org](http://dl.acm.org)
- SpringerLink [www.springerlink.com](www.springerlink.com)
- ScienceDirect / Scopus [http://www.sciencedirect.com](http://www.sciencedirect.com)
- Web of science [http://apps.webofknowledge.com/WOS](http://apps.webofknowledge.com/WOS)

#### A. Search String

The aim for our search string is to capture all results that relate sustainability or environmental issues with software engineering or requirements for software systems. The reasons for searching for requirements is that in this early development phase sustainability issues should emerge. The search string used on all databases is:

```plaintext
(sustainab* OR environment* OR ecolog* OR green) AND

software engineering OR requirement OR software system)
```

Although we explicitly list keywords in our search string that rather point to environmental sustainability, we expect to find all dimensions of sustainability.

### B. Inclusion Criteria

We chose the following inclusion criteria in order to select the relevant publications to answer our research questions:

- Publication date between 1/1/1991 - 31/12/2011
- Requirements phase of software development process
- Explicit mentioning of software engineering
- Scientific soundness
- Relevance with respect to research questions
- Analysis of sustainability-relevant application domains
- Coverage of a SW ecosystem or SW sustainability

\(^3\)http://dblp.mpi-inf.mpg.de/dblp-mirror/index.php
C. Exclusion Criteria

- “Environment” used in the sense of system environment, not nature.
- “Ecosystem” used as population of interacting systems, for example, agents.

D. Roles and Responsibilities

- Birgit Penzenstadler (TUM, principal researcher): IEEE-EXplore, result classification, detailed analysis
- Zolboo Ochirsukh, Elena Mircheva, Duc Tien Vu, Tuan Duc Nguyen (TUM, student research assistants): search on ACM, Web of Science, ScienceDirect, SpringerLink
- Veronika Bauer (TUM, expert reviewer): assessment of search result classification, review of detailed analysis
- Coral Calero (UCLM, expert reviewer): assessment of search result classification and detailed analysis
- Xavier Franch (UPC, expert reviewer): review of detailed analysis

E. Article Selection Process

The process was conducted as follows:
1) The researchers execute the search on each database and save the references in bibliography files.
2) The principal researcher reads all titles and abstracts and checks the inclusion and exclusion criteria for each entry. Major criterion is the topic of the content.
3) The principal researcher classifies the papers and articles according to type, topic, and domain.
4) The expert reviewers reassess the classification and inclusion/exclusion of search results. After their reassessment, we introduce an additional result classification: domain-specific papers that are interesting to learn from but not focused on software engineering.
5) The principal researcher extracts statistics and analyses the included results in further detail. This is followed by a second assessment from the expert reviewers.

F. Data Analysis

The data is tabulated to show:
- The databases and numbers of query results. (RQ1)
- Listed by database for included publications:
  - Author, reference, date (RQ1)
  - Publication type and type of content (RQ6)
  - Topic of content (RQ2, RQ4, RQ5)
  - Application domain (RQ7)
  - Benefit for our body of knowledge (RQ4)
- The number of relevant publications per year. (RQ1)
- The respective venues and journals. (RQ1)

Furthermore, the findings for RQ3, RQ4 and RQ5 are reported on separately. Due to the limitation of space, we provide the full list of references of the primary study as online appendix [8].

III. Results

The overall number of results for each database is listed in Tab. I.

<table>
<thead>
<tr>
<th>Database</th>
<th>Date</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE Xplore</td>
<td>27/12/11</td>
<td>319,960</td>
</tr>
<tr>
<td>ACM Digital Library</td>
<td>26/12/11</td>
<td>104,217</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>29/12/11</td>
<td>500,004</td>
</tr>
<tr>
<td>ScienceDirect / Scopus</td>
<td>29/12/11</td>
<td>10,749</td>
</tr>
<tr>
<td>Web of Science</td>
<td>29/12/11</td>
<td>80,503</td>
</tr>
</tbody>
</table>

All results were ordered “by relevance” as displayed by the databases. From these results, we considered the first 100 results of each database in our first iteration of the study. In total, we reviewed 500 publications.

The following abbreviations are used to categorize the results in Tab. II-VI:
- **Publication**: Kind of publication, e.g., Journal Article (A), Conference Paper (CP), Workshop Paper (WP), Book Chapter (BC), Letter to the editor (L)
- **Type**: Kind of content presented in the publication, e.g., method, experience report, empirical study, tool
- **Topic**: Short hint on principal content and keywords of the paper or article
- **Domain**: Application or technology domain considered in the publication, e.g., transport, aviation, embedded systems, information systems, human aspects
- **Benefit**: Classification of why we consider this publication to be relevant with respect to the research questions: Sustainability in software engineering (S in SE), sustainability-related application domains (S App Dom), sustainability (modeling) concept (S Concept), sustainable software solutions (S SW Sol), sustainable hardware solutions (S HW Sol)

RQ1: How much activity was there in the last 20 years?

We summarized the number of relevant publications per database in Tab. VII, per year in Tab. VIII, and per publication type in Tab. IX. In the last two years, there was a significant increase in the number of publications, and there was no publication included that was older than 2005, so our hypothesis for RQ1 holds. None of the results we included are older than 2005, but we did have older search results in the query evaluations, so this is not due to restricted availability online. Although we executed the search queries in late December, we already found journal articles dated to January 2012 in the results which we included as they were fully available.

While the venues were relatively distributed, there was an accumulation of publications from “Environmental Modeling & Software” as well as the “Journal of Cleaner Production”. The fact that we classified many of the publications as “software solutions” or “sustainability-related application
### Table II
**INCLUDED RESULTS FROM IEEE XPLOR, FULL REFERENCES IN APPENDIX [8]**

<table>
<thead>
<tr>
<th>Author and Reference</th>
<th>Year</th>
<th>Pub. Type</th>
<th>Type</th>
<th>Topic</th>
<th>Domain</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>May et al.</td>
<td>2006</td>
<td>CP</td>
<td>method</td>
<td>composing biological workflows through web services</td>
<td>parallel processing</td>
<td>$S$ App Dom</td>
</tr>
<tr>
<td>Zimmer et al.</td>
<td>2011</td>
<td>CP</td>
<td>method</td>
<td>design for resilience of networked critical infrastructures</td>
<td>eco-informatics</td>
<td>$S$ App Dom</td>
</tr>
<tr>
<td>Alberts et al.</td>
<td>2010</td>
<td>CP</td>
<td>metrics</td>
<td>sustainability of performance of software</td>
<td>education</td>
<td>$S$ App Dom</td>
</tr>
<tr>
<td>Zhou et al.</td>
<td>2006</td>
<td>CP</td>
<td>method</td>
<td>green manufacturing in structural engineering</td>
<td>soc tech</td>
<td>$S$ App Dom</td>
</tr>
</tbody>
</table>

### Table III
**INCLUDED RESULTS FROM ACM, FULL REFERENCES IN APPENDIX [8]**

<table>
<thead>
<tr>
<th>Author and Reference</th>
<th>Year</th>
<th>Type</th>
<th>Topic</th>
<th>Domain</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greiner et al.</td>
<td>2007</td>
<td>A</td>
<td>model</td>
<td>habitat suitability models for fire management</td>
<td>ecological knowledge</td>
</tr>
<tr>
<td>Glass et al.</td>
<td>2009</td>
<td>A</td>
<td>challenge</td>
<td>computer science challenges at research</td>
<td>high-performance computing</td>
</tr>
<tr>
<td>O'Halloran et al.</td>
<td>2010</td>
<td>CP</td>
<td>method</td>
<td>data mining for biodiversity predictions in forests</td>
<td>forestry, data mining</td>
</tr>
<tr>
<td>Ponsenmakur et al.</td>
<td>2011</td>
<td>CP</td>
<td>method</td>
<td>tool sustainability in software engineering</td>
<td>education</td>
</tr>
<tr>
<td>Zhong et al.</td>
<td>2010</td>
<td>CP</td>
<td>prototypes</td>
<td>design of self-propelled walking bread crust</td>
<td>digital manufacturing</td>
</tr>
<tr>
<td>Andrade et al.</td>
<td>2009</td>
<td>A</td>
<td>method</td>
<td>radiation management of two Algae over boards</td>
<td>GIS tool</td>
</tr>
<tr>
<td>Mistry et al.</td>
<td>2000</td>
<td>A</td>
<td>method</td>
<td>computer reconstruction of the biological networks</td>
<td>ecological networks</td>
</tr>
<tr>
<td>Panarat et al.</td>
<td>2006</td>
<td>A</td>
<td>method</td>
<td>technology-driven knowledge modeling in ecology</td>
<td>education</td>
</tr>
<tr>
<td>Bowers et al.</td>
<td>2007</td>
<td>CP</td>
<td>method</td>
<td>decision analysis tools for research and interactive technologies</td>
<td>technology economics</td>
</tr>
<tr>
<td>Anando et al.</td>
<td>2009</td>
<td>A</td>
<td>model</td>
<td>environmental monitoring</td>
<td>environmental assessment</td>
</tr>
<tr>
<td>Perera et al.</td>
<td>2009</td>
<td>A</td>
<td>method</td>
<td>knowledge discovery for causal unions classification</td>
<td>environmental monitoring</td>
</tr>
<tr>
<td>Mousaizadeh et al.</td>
<td>2007</td>
<td>A</td>
<td>method</td>
<td>public participation modeling in management of groundwater contamination</td>
<td>groundwater management</td>
</tr>
<tr>
<td>Deck et al.</td>
<td>2011</td>
<td>CP</td>
<td>model</td>
<td>water supply and demand systems in nature change in energy consumption</td>
<td>energy sustainability</td>
</tr>
<tr>
<td>Thachuk et al.</td>
<td>2007</td>
<td>A</td>
<td>model</td>
<td>assessing the sustainability of coastal lakes</td>
<td>environmental monitoring</td>
</tr>
<tr>
<td>Shih et al.</td>
<td>2010</td>
<td>A</td>
<td>model</td>
<td>butterfly and meadow ecology for conservation-aware ubiquitous learning</td>
<td>mobile learning</td>
</tr>
<tr>
<td>Cushing et al.</td>
<td>2007</td>
<td>CP</td>
<td>method</td>
<td>database design for ecologists</td>
<td>ecosystem information</td>
</tr>
<tr>
<td>Cushing et al.</td>
<td>2006</td>
<td>CP</td>
<td>overview</td>
<td>eco-informatics and natural resource management</td>
<td>eco-informatics</td>
</tr>
<tr>
<td>Kang et al.</td>
<td>2007</td>
<td>A</td>
<td>model</td>
<td>GIS-based quality land management systems for nutrient planning</td>
<td>land management</td>
</tr>
<tr>
<td>Amid et al.</td>
<td>2010</td>
<td>CP</td>
<td>tool</td>
<td>tool for estimating the energy consumption of software</td>
<td>energy simulation</td>
</tr>
<tr>
<td>Chavarria et al.</td>
<td>2011</td>
<td>CP</td>
<td>model</td>
<td>simulations targeting to evaluate renewable energy technology</td>
<td>renewable energy</td>
</tr>
<tr>
<td>Vice-Blanc et al.</td>
<td>2011</td>
<td>HC</td>
<td>tool</td>
<td>optical network and cloud as architecture for a state, future internet</td>
<td>cloud</td>
</tr>
<tr>
<td>Dockum et al.</td>
<td>2008</td>
<td>A</td>
<td>model</td>
<td>explores the means of “sustainable” behavior</td>
<td>life cycle analysis</td>
</tr>
<tr>
<td>Ramamoorthy et al.</td>
<td>2009</td>
<td>CP</td>
<td>method</td>
<td>composing eco-sciences through financial and fiscal instruments</td>
<td>economic</td>
</tr>
<tr>
<td>Markus et al.</td>
<td>2007</td>
<td>A</td>
<td>tool</td>
<td>role-playing game for collective awareness of wise medical usage</td>
<td>environmental education</td>
</tr>
<tr>
<td>Saldanha et al.</td>
<td>2007</td>
<td>CP</td>
<td>method</td>
<td>factor approach to ecological data analysis</td>
<td>environmental management</td>
</tr>
<tr>
<td>Alves et al.</td>
<td>2010</td>
<td>A</td>
<td>tool</td>
<td>tool to detect and predict urban growth patterns</td>
<td>urban pattern</td>
</tr>
<tr>
<td>Kacar et al.</td>
<td>2008</td>
<td>CP</td>
<td>model</td>
<td>sustainable input-output in community-based nonprofits</td>
<td>education</td>
</tr>
<tr>
<td>Butterfield et al.</td>
<td>2010</td>
<td>CP</td>
<td>tool</td>
<td>model transfer scheme for energy harvested WSN gateways</td>
<td>energy</td>
</tr>
<tr>
<td>Umar et al.</td>
<td>2011</td>
<td>CP</td>
<td>model</td>
<td>persistent coexistence for a linear response ornithological model</td>
<td>modeling</td>
</tr>
<tr>
<td>Vance et al.</td>
<td>2007</td>
<td>CP</td>
<td>model</td>
<td>persistent coexistence for a linear response ornithological model</td>
<td>modeling</td>
</tr>
<tr>
<td>Posman et al.</td>
<td>2011</td>
<td>CP</td>
<td>method</td>
<td>causal information visualization of predictor data</td>
<td>sustainability design</td>
</tr>
<tr>
<td>Pa et al.</td>
<td>2011</td>
<td>CP</td>
<td>framework</td>
<td>urban media framework of social innovation and service design</td>
<td>service design</td>
</tr>
</tbody>
</table>

### Table IV
**INCLUDED RESULTS FROM SCIENCE DIRECT, FULL REFERENCES IN APPENDIX [8]**

<table>
<thead>
<tr>
<th>Author and Reference</th>
<th>Year</th>
<th>Type</th>
<th>Topic</th>
<th>Domain</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahmed et al.</td>
<td>2011</td>
<td>A</td>
<td>tool</td>
<td>car-parking regulations for sustainable transport development</td>
<td>transportation</td>
</tr>
<tr>
<td>Ahmed et al.</td>
<td>2010</td>
<td>A</td>
<td>model</td>
<td>modeling for systems, their dynamic and their complexity</td>
<td>policy system</td>
</tr>
<tr>
<td>Adelantado et al.</td>
<td>2011</td>
<td>A</td>
<td>tool</td>
<td>technical assessment and evaluation of environmental models</td>
<td>environmental modeling</td>
</tr>
<tr>
<td>Adelantado et al.</td>
<td>2010</td>
<td>A</td>
<td>tool</td>
<td>method for the systemic sensing of biophysical environments</td>
<td>environmental modeling</td>
</tr>
<tr>
<td>Benza et al.</td>
<td>2011</td>
<td>A</td>
<td>tool</td>
<td>dynamic simulation and visualization for mathematical models</td>
<td>environmental modeling</td>
</tr>
<tr>
<td>Benza et al.</td>
<td>2010</td>
<td>A</td>
<td>tool</td>
<td>tool for managing the dynamic of a virtual environment</td>
<td>environmental modeling</td>
</tr>
<tr>
<td>Benza et al.</td>
<td>2009</td>
<td>A</td>
<td>tool</td>
<td>software tool designed to simulate ensemble forecasts of numerical variables</td>
<td>environmental modeling</td>
</tr>
<tr>
<td>Conte et al.</td>
<td>2008</td>
<td>A</td>
<td>tool</td>
<td>software package with the aim of enabling the ongoing conservation management</td>
<td>biodiversity protection</td>
</tr>
<tr>
<td>Dus et al.</td>
<td>2010</td>
<td>A</td>
<td>tool</td>
<td>tool for the management of ecosystem modeling</td>
<td>environmental modeling</td>
</tr>
<tr>
<td>Bottcher et al.</td>
<td>2009</td>
<td>A</td>
<td>tool</td>
<td>software package for modeling of contaminant transport in groundwater</td>
<td>environmental modeling</td>
</tr>
<tr>
<td>Kit et al.</td>
<td>2011</td>
<td>A</td>
<td>tool</td>
<td>sensor-based identification of urban shade in india using sensing data</td>
<td>aerial photography</td>
</tr>
</tbody>
</table>

### Table V
**INCLUDED RESULTS FROM SPRINGER LINK, FULL REFERENCES IN APPENDIX [8]**

<table>
<thead>
<tr>
<th>Author and Reference</th>
<th>Year</th>
<th>Pub. Type</th>
<th>Topic</th>
<th>Domain</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al.</td>
<td>2010</td>
<td>CP</td>
<td>study</td>
<td>spatial aspects of sustainable urban development</td>
<td>economy</td>
</tr>
<tr>
<td>Liu et al.</td>
<td>2009</td>
<td>CP</td>
<td>study</td>
<td>multi-source remote sensing data for estimating ecological water requirement</td>
<td>environmental modeling</td>
</tr>
</tbody>
</table>


domain”, some more as “sustainability concepts” and only few as “sustainability in software engineering” implies that there is still relatively little research published that could be considered for building up a body of knowledge.

Table VII
INCLUDED RESULTS PER DATABASE

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Included Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEEXplore</td>
<td>5 out of 100</td>
</tr>
<tr>
<td>ACM Digital Library</td>
<td>32 out of 100</td>
</tr>
<tr>
<td>Springer Link</td>
<td>5 out of 100</td>
</tr>
<tr>
<td>Science Direct</td>
<td>30 out of 100</td>
</tr>
<tr>
<td>Web of Science</td>
<td>24 out of 100</td>
</tr>
<tr>
<td>Total</td>
<td>96 out of 500</td>
</tr>
</tbody>
</table>

Table VIII
INCLUDED RESULTS PER YEAR

<table>
<thead>
<tr>
<th>Year</th>
<th>Number of Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997 - 2005</td>
<td>0</td>
</tr>
<tr>
<td>2006</td>
<td>9</td>
</tr>
<tr>
<td>2007</td>
<td>12</td>
</tr>
<tr>
<td>2008</td>
<td>11</td>
</tr>
<tr>
<td>2009</td>
<td>11</td>
</tr>
<tr>
<td>2010</td>
<td>21</td>
</tr>
<tr>
<td>2011</td>
<td>29</td>
</tr>
<tr>
<td>2012</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>96</td>
</tr>
</tbody>
</table>

Table IX
INCLUDED RESULTS PER PUBLICATION TYPE

<table>
<thead>
<tr>
<th>Publication Type</th>
<th>Number of Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Journal articles</td>
<td>65</td>
</tr>
<tr>
<td>Book chapters</td>
<td>2</td>
</tr>
<tr>
<td>Conference papers</td>
<td>27</td>
</tr>
<tr>
<td>Workshop papers</td>
<td>1</td>
</tr>
<tr>
<td>Letters to the editor</td>
<td>1</td>
</tr>
<tr>
<td>Technical reports</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>90</td>
</tr>
</tbody>
</table>

RQ2: What research topics are being addressed?

For a quick illustrated overview, we have generated a weighted topic cloud from keywords, taken from the titles and abstracts, that visualizes the topics in Fig. 1. We have derived a taxonomy for the addressed research topics in Fig. 2 that abstracts from some of the details listed in the original classifications tables in Tab. II-VI. The dimensions of the taxonomy are the degree of domain specificity, from general purpose to domain-specific research and the indexing between analytical approaches (frameworks and assessment) and constructive approaches (methods and tools). The taxonomy shows a tendency towards domain-specific, constructive approaches. There are not many publications rated as general purpose, and there is little methodical guidance for supporting sustainability.

Both the keyword cloud and taxonomy rely strictly on keywords taken from titles and abstracts. Nevertheless, their reproduction might reveal slightly varied results, but we do not consider that a problem as we use them only to give an overview of topics without deriving any further statistics from them.

RQ3: What are the limitations of current research?

To identify limitations of current research, we reviewed our classification of topics and application domains in Tab. II-VI. We performed a pragmatic and informal gap analysis that resulted in three major limitations:

- **High complexity.**
  
  Reason: Due to the high connectivity between the different aspects of sustainability, (software) systems engineering becomes highly complex. This is visible in knowledge management approaches, e.g., [9] and decision support systems, e.g., [10].
  
  Conclusion: High complexity requires clear concept definitions and consistent, traceable models. One method to cope with high system complexity that might prove helpful is systems’ thinking [11].
- **High domain-specific.**
  
  Reason: The frameworks and methods we found within the results are highly domain-specific, e.g., [12], [13]. This is also visible in the higher density of domain-specific approaches in Fig. 2.
  
  Conclusion: Effective approaches for supporting sustainability require specific domain knowledge.

- **Software engineering.**
  
  Reason: There is only one approach in software engineering that explicitly addresses sustainability. It is a reference framework with specific application in web engineering [14].
  
  Conclusion: An encompassing reference framework for SE is still missing.

**RQ4: How is sustainability support performed?**

Constructive support for sustainability is performed by frameworks, models, methods, and metrics (Tab. X). Thereby, most approaches are specific to a special application domain, as visible by the density on the domain-specific side in Fig. 2.

- Frameworks, e.g., for civil engineering [12] or contaminant transport [15]
### Table X
**INCLUDED RESULTS PER CONTENT TYPE**

<table>
<thead>
<tr>
<th>Class</th>
<th>Type of Content</th>
<th>Number of Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructive</td>
<td>Method</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Metrics</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Framework</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Tool/Prototype</td>
<td>2</td>
</tr>
<tr>
<td>Empirical</td>
<td>Review</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Study</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Evaluation</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Assessment</td>
<td>4</td>
</tr>
<tr>
<td>Discussion</td>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Challenges</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Analysis</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reflection</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Perspective</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Strategy</td>
<td>1</td>
</tr>
</tbody>
</table>

- Models, e.g., for software systems [14] or databases [16]
- Methods for specific application areas, e.g., security technology [13], green product design [17], or ecology knowledge [9]
- Metrics, e.g., for sustainability in eBusiness [18]

We chose just a few of the approaches for illustration and preferred the ones that are rather close to sustainability in software engineering. Furthermore, there are some empirical publications and rather few discussions.

**RQ5: Which methods are in use?**

There is a wide variety of methods in use for different purposes — we found traditional software engineering techniques as well as domain-specific techniques and methods from other disciplines.

Many approaches apply entity-relationship modeling, e.g., [9], as means to represent their data, knowledge, or information models. Neural networks are in use for dynamic environments and simulations, e.g., [19]. Methods adapted from other disciplines are, inter alia, cost calculations, e.g., [20], and life cycle analysis, e.g., [21].

**RQ6: Are there case studies available?**

We classified publications as case studies when they were explicitly named as such in the abstract and they were not, for example, only containing a small illustrative case study within a method proposal. The studies are listed in Tab. XI. Unfortunately, none of the studies contributes explicitly to an understanding of how to develop software for sustainable systems, but rather to analyses of specific application domains. Furthermore, publications that promote studies are often method proposals illustrated in a case study performed by the principal researcher.

**RQ7: Which domains are already considered?**

For an illustrated overview, please see the weighted domain cloud that visualizes the application domains in Fig. 3. We have derived a taxonomy for the domains that were used and described in the publications in Fig. 4. We used the same dimensions as for the research topics taxonomy in Fig. 2 and identified five coarse-grained domain clusters: *Systems & Knowledge* in the area of general purpose, analytical approaches, *Technologies & Methods* on the constructive side of the general purpose dimension, *Education* somewhere in the middle between these two, special *Disciplines* provide more domain-specific, analytical approaches, and the corresponding *Application & Implementation* cluster contributes the domain-specific, constructive approaches. These clusters are not overlap-free, but only a means to illustratively structure their diversity. The terms within the cluster clouds in Fig. 4 indicate the individual domains.

### IV. Discussion

This section provides a discussion of the results and of the threats to validity for this study.

**A. Conclusions on the State of the Art**

We started our search expecting to find more results to be classified as Sustainability in Software Engineering ($S$ in SE) in column Benefit in Tab. II-VI. As we found less than expected for a body of knowledge on $S$ in SE, we decided to extend the inclusion to publications that we classified as a research we could learn from when further investigating sustainability in software engineering. This lead to the other Benefit categories $S$ Concept, $S$ App Dom, $S$ SW Sol, and $S$ HW Sol as explained in Sec. III.

In our opinion, there is still a lot of research work to be done, especially to support the different dimensions of sustainability from within the software engineering discipline. This can either occur in form of domain-independent guidelines or domain-specific methods.

**B. Conclusions for a Body of Knowledge**

Due to these findings, our envisioned Body of Knowledge has areas that represent the core $S$ in SE publications, plus areas that represent application domains with software and hardware solutions as well as sustainability concepts from related disciplines that we can learn from. This is illustrated in Fig. 5.

![Figure 5. Areas of the Body of Knowledge for S in SE](image-url)
## Table XI
### CASE STUDIES

<table>
<thead>
<tr>
<th>Author and ref.</th>
<th>Domain</th>
<th>Context</th>
<th>Applied method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huang et al. [10]</td>
<td>web engineering</td>
<td>support green customers’ decision process on electronic commerce</td>
<td>questionnaire and experiment</td>
</tr>
<tr>
<td>Faith-Ell et al. [22]</td>
<td>energy</td>
<td>application of environmental requirements in Swedish road maintenance</td>
<td>index decomposition analysis</td>
</tr>
<tr>
<td>Liu et al. [23]</td>
<td>cleaner production</td>
<td>energy requirements and carbon dioxide emissions of tourism industry</td>
<td>semi-structured interviews</td>
</tr>
<tr>
<td>Tseng et al. [24]</td>
<td>business research</td>
<td>evaluating a firm’s green supply chain management</td>
<td>relational analysis, experiment</td>
</tr>
<tr>
<td>Yen et al. [25]</td>
<td>property development</td>
<td>management’s role in adopting green purchasing standards in industry</td>
<td>questionnaires</td>
</tr>
<tr>
<td>Zhang et al. [26], [27]</td>
<td>ecology</td>
<td>costs and barriers of green property development in China</td>
<td>cost analysis</td>
</tr>
<tr>
<td>Jin et al. [29]</td>
<td>hydrology</td>
<td>urban wetland planning in Beijing</td>
<td>ecological complexity research</td>
</tr>
<tr>
<td>Tseng et al. [24]</td>
<td>cleaner production</td>
<td>evaluating a firm’s green supply chain management</td>
<td>integrated calculation</td>
</tr>
<tr>
<td>Jia et al. [28]</td>
<td>ecology</td>
<td>urban wetland planning in Beijing</td>
<td></td>
</tr>
<tr>
<td>Jin et al. [29]</td>
<td>hydrology</td>
<td>urban wetland planning in Beijing</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Weighted domain cloud, created with http://www.wordle.net/

Figure 4. Taxonomy of application domains
C. Threats to Validity

There is a number of threats to validity that we are aware of and tried to minimize by different mitigation actions.

1) Researcher's bias: The semi-automatic part of the search was performed by five researchers. There could be a researcher's bias as the first selection was performed by only one researcher. We minimized the effects of such a bias by two measures:

- We explicitly stated the research questions, inclusion and exclusion criteria, and the rationale for performing the search.
- The first selection was reviewed and assessed by two expert reviewers from different institutions (TUM, UCLM). Differences were subsequently discussed, resolved and commonly agreed upon.
- The detailed analysis of the principle researcher was reviewed by all three expert reviewers (TUM, UCLM, UPC).

2) Search string validity: The search string validity can be questioned under two aspects: On one hand whether it filtered out too many publications that would have been relevant, and on the other hand whether it included too many irrelevant results and was, in either case, not the adequate search string.

Indicators for too many false positives are purely hardware papers, but as the automatically found Green IT publications all contained part of the second parenthesis of the search string, they were included in the results. Then, purely application in environmental domains, for example, agricultural support systems with no explicit relation to sustainability but relevant in case they exhibited an explicit link to sustainability in their content. Furthermore, “environment” used in the sense of system environment, not nature — these samples had to be excluded by hand as well as “ecosystem” used as population of interacting systems, for example, agents.

Indicators for too many relevant exclusions were that we found less relevant results than we would have expected. This can either be due to a search string that was too restrictive, to a search that was not extensive enough, or to the fact that there is rather little published yet on that specific topic. Not all publications we would have expected showed up early in the search results. For example, we missed Cabot et al. [30], as they treat goal modeling for supporting sustainability in the context of conference organisation. Mahaux et al. [7] were also missing in the results, with their work on exploring sustainability requirements.4

3) Database query evaluation: We did not have any information on which database performed which kind of search query evaluation, and a lazy versus an eager database query evaluation of the search string would probably have a significant impact on the search results, considering that we reviewed the first 100 most relevant results.

In case of a “lazy” search string evaluation, the results might have included more references matching early parts of the search string than compared to matching later parts. In that case, the results might be slightly biased in terms of favoring the terms “sustainab*” and “software engineering” and subordinating “green” and “software systems”.

As many of the results contained the term “software system” and not “software engineering”, we are confident that there was no bias introduced by database query evaluation.

4) Cross-validation of the search engines: We received hardly any double entries in the automatic search results. We would have expected some double entries in the more general databases ScienceDirect and WebOfScience. We decided not to use meta search engines in our first iteration of the SLR because relying on only one meta search engine would have made us completely dependent of the reliability of that engine, and using various meta search engines would have led to highly redundant results, as a pre-check showed.

Interestingly, Web of Science found Estrin [31] highly ranked, which originates from IEEEXplore but was not included in the IEEEXplore results (at least not within the first 100 results). This might be a hint towards different search query evaluation.

It would be one interesting step in future work to replicate the searches on more databases and meta search engines and explicitly compare the coverage.

V. Conclusions

In this paper, we presented the results of our SLR [4] on the research activity in sustainability in software engineering and related topics that allow for building up a body of knowledge. We considered 96 of 500 reviewed publications relevant with respect to our research questions and classified them according to content, topic, application domain, and potential benefit for further investigation. On that basis, we provided taxonomies for represented research topics and application domains. As there were not as many publications explicitly presenting work on sustainability in software engineering than expected, we propose an extended body of knowledge for S in SE that includes related application domains and sustainability concepts from related disciplines that we can learn from when further investigating S in SE.

Future work is to extend the study in two directions: on one hand by snowballing (following references) and on the other hand via meta search engines, book search engines, and dedicated journal searches. Probably even more important is the challenge of making SLRs themselves “sustainable” by providing yearly updates that not only repeat an SLR but adapt the iterations over the years according to lessons learned from previous iterations. Thereby, we can establish stable bodies of knowledge.

---

4These works were not included into the results manually because we wanted to strictly follow the SLR method. However, they will be included in the extended version and the envisioned body of knowledge.
REFERENCES


3 Systematic Mapping Study on Software Engineering for Sustainability
Systematic Mapping Study on Software Engineering for Sustainability (SE4S)

Birgit Penzenstadler, Ankita Raturi, Debra Richardson
University of California, Irvine
{bpenzens,araturi,djr}@uci.edu

Henning Femmer
Technische Universität München
Femmer@in.tum.de

Coral Calero
Universidad de Castilla - La Mancha, Spain
Coral.calero@uclm.es

Xavier Franch
Universitat Politècnica de Catalunya, Spain
franch@essi.upc.edu

Abstract

Background/Context: The objective of achieving higher sustainability in our lifestyles by information and communication technology has lead to a plethora of research activities in related fields. Consequently, Software Engineering for Sustainability (SE4S) has developed as an active area of research. Objective/Aim: Though SE4S gained much attention over the past few years and has resulted in a number of contributions, there is only one rigorous survey of the status of research, as most work has been conducted in the last 4 years. Method: The applied method is a systematic mapping study through which we investigate which contributions were made, which knowledge areas are most explored, and which research type facets have been used, to distill a common understanding of the state-of-the-art in SE4S. Results: We contribute an overview of current research topics and trends, and their distribution according to the research type facet and the application domains. Furthermore, we aggregate the topics into clusters and list proposed and used methods, frameworks, and tools. Conclusion: The research map shows that impact currently is limited to few knowledge areas and there is need for a future roadmap to fill the gaps.

Categories and Subject Descriptors
D.2.1 [Software Engineering]: [Requirements, Methodologies]

Keywords
Sustainability, Software Engineering, Systematic Mapping Study

1. MOTIVATION & BACKGROUND

Over the last decades, sustainability research has emerged as an interdisciplinary area; knowledge about how to achieve sustainable development has grown, while political action towards the goal is still in its infancy [1]. A sustainable world is broadly defined as “one in which humans can survive without jeopardizing the continued survival of future generations of humans in a healthy environment” [2]. This anthropocentric view of sustainability allows us to consider the implications of, and necessities for, human existence in the world. Sustainability can be also discussed with reference to a concrete system—such as an ecological system, a human network, or a specific software system. Here, global sustainability implies the capacity for endurance given the functioning of all these systems in concert. Software Engineering for Sustainability has developed as a current focus of research as a result of software engineers engaging in issues regarding the impact of software systems on global sustainability.

Definition

The term Sustainable Software can be interpreted in two ways: (1) the software code being sustainable, agnostic of purpose, or (2) the software purpose being to support sustainability goals, i.e. improving the sustainability of humankind on our planet. Ideally, both interpretations coincide in a software system that contributes to more sustainable living. Therefore, in our context, sustainable software is energy-efficient, minimizes the environmental impact of the processes it supports, and has a positive impact on social and/or economic sustainability (1 & 2). These impacts can occur direct (energy), indirect (mitigated by service) or as rebound effect [3]. The aim of Software Engineering for Sustainability (SE4S) is to make use of methods and tools in order to achieve this notion of sustainable software.

Motivation

There is a plethora of (new) journals, conferences and workshops where the topic pops up, so it is hard to get a comprehensive overview of the state of research. There is only one earlier systematic mapping study on sustainability in software engineering, namely the study performed by a subset of the authors of the work at hand from 2012. This first review [4] is now extended and analyzed in more depth and detail, as the first study did not differentiate research facets and knowledge areas. Furthermore, the first study revealed that only very few topics were in the actual area of...
software engineering, which is why the study then included related research on sustainable software systems outside of software engineering. As the topic has been researched very actively in the past few years, this second study leads to a larger set of data points that allow to draw more conclusions.

**Research Objective** Our aim is to provide an overview of the current state of research on software engineering for sustainability. The first step was our previous work with an earlier study on the available research [4], and now a related effort is made after only two years because the field has substantially evolved since then.

**Contribution** We contribute a systematic mapping study that follows the guidelines in [5]. It takes into account the lessons learned from the previous study [4] by defining more robust research questions, using an adapted search string, and including a number of publication channels (journals, conferences and workshops) on the topic that have either been just recently established or were not indexed yet in the earlier study.

2. STUDY DESIGN

We describe the study design in terms of research questions, set-up, and conducted procedures.

**Research Questions (Scope).**

The overall research objective of the study is to give an overview of the current state of the art in supporting sustainability in software engineering research and practice. This is detailed in the following research questions:

- **RQ1** What research topics are being addressed?
- **RQ2** How have these research topics evolved over time?
- **RQ3** How is sustainability support performed (e.g., models and methods)?
- **RQ4** Which of those models and methods are used in practice?
- **RQ5** Which research type facets have been considered in the contributions?
- **RQ6** Which application domains have been considered?
- **RQ7** Which research groups are most active and what is the distribution between academics and practitioners?

**Roles & Responsibilities.**

The roles and responsibilities for this project are defined in Tab. 1. We have two principal researchers (Birgit Penzenstadler and Ankita Raturi), three supporting researchers (Henning Femmer, Coral Calero, Xavier Franch), one internal reviewer (Debra Richardson) and two external reviewers (Daniel Méndez Fernández and Marcela Genero).

**Search Strategy.**

**Information Sources.** The search process for this study is based on an automated search of the following indexing systems and digital libraries: DBLP, Science Direct, Web Of Science, INSPEC, IEEE Xplore, Springer, ACM, JSTOR, arXiv, Wiley, and Citeseer. Furthermore, we added manual searches on conference and workshop proceedings of 2013, as pretests of the search string have revealed that they did not show up in the search results of the indexing systems: ICTIS'13, GREENS'13, and RE4SuSy'13. The reason for them not being indexed was that it was still too early after their publication, but as we knew of their existence and relevance, we decided to include them in order to have more up-to-date results.

**Search String.** The aim for our search string is to capture all results that relate sustainability or environmental issues with software engineering or requirements for software systems. Not only in software engineering, but especially during the early phase of requirements engineering sustainability issues should emerge and be discussed, which is the reason for specifically including requirement in the search string. The search string used on all databases is:

(sustainab* OR ecolog* OR green) AND (software engineering OR requirement* engineering OR requirement* specification OR software specification OR system specification)

We decided not to include “environment” as alternative for sustainab*, ecolog* or green in the first parenthesis because pretests showed only false positives as it is a term frequently used for denoting the system context, operational context, or business context. The second parenthesis contains the part making it relevant for software engineering and the first parenthesis contains the part that links it to sustainability including synonyms and alternative terms that we know are in use. Although we explicitly list keywords in our search string that point to environmental sustainability, we are interested in all dimensions of sustainability as they are strongly related to each other.

**Search Execution.** We execute the search on the databases specified earlier. The search string is used to perform the search including the meta data fields title, abstract, and keywords. In case the search returned more than 100 results ordered according to the relevance with regard to the search string, we use the first 100 search results of each database. We retrieve the meta information (full citation and abstract) as well as the full texts. We consolidate the results and clean from duplicates. We provide the primary sources as well as a separate voting sheet per classification assessor in a Dropbox folder.

**Study Selection Criteria.** We chose the following inclusion criteria to select the relevant publications to answer our research questions: Relevance with respect to research questions, scientific soundness (see quality assessment in Sec. 2), and coverage of a software system (as opposed to pure hardware systems).

**Exclusion Criteria** are Environment used in the sense of sys-

---

1 The search string used in the preceding study was (sustainab* OR environment* OR ecolog* OR green) AND (software engineering OR requirement* OR software system)
tem environment, not nature, and Ecosystem used as population of interacting systems, for example, agents.

**Study selection procedures.**

The process was conducted as follows: The five voters read all titles and abstracts and decide on the inclusion and exclusion for each entry according to the criteria given above. If unsure about an article, they read more of the paper until they are decided. Disagreements among voters are resolved by majority as we chose an uneven number of assessors. This also requires at least 3 out of 5 votes for decision taking. The internal reviewer reassesses the inclusion/exclusion of search results.

**Study quality assessment checklists and procedures.**

The following assessment checklist has been used to assess the quality of the studies under consideration: Peer-reviewed articles, reporting on background and context, description of research method, report on threats to validity.

For quality assessment, we performed internal and external reviews as also specified in Tab. 1. There were five internal reviews and three external ones. We conducted internal reviews of the protocol, of the voting, of the data extraction and classification, of the analysis and data synthesis, of the report. External reviews were performed of the protocol, of the analysis, of the report.

**Data extraction strategy.**

The principal researchers classify the studies according to the research type facets [7] and the knowledge area [8], as detailed in the list below. They extract information on topics, methods, frameworks, tools, case studies, and application domains.

The data extraction form captures the following data for each included primary resource:

- Metadata: Authors, Year of publication, Title, Source, Keywords, Research topic, Institution
- Research type facets [7]: Philosophical, Exploratory, Solution, Validation, Evaluation, Opinion, or Experience.
- Application domain (if applicable)
- Framework and/or Method (if applicable)
- Tool (if applicable)

**Synthesis of the extracted data.**

The principal researchers extract statistics and analyse the included results in further detail. They map out the current research. The internal reviewer assesses the analysis results and provides feedback. The external reviewers provide feedback. To conduct the data synthesis, we derived descriptive statistics for maps from the extracted data, performed semantic modeling of research topic clusters, mapped out current work, made timelines with amount of publications according to research topic, research type facet, and knowledge area.

### 3. RESULTS

An overview of the search result numbers is provided in Tab. 2. The publications that were voted in by the majority of reviewers are listed later in Tab. 4. The 83 resulting publications were published quite across a range of journals, conferences, and workshops and covered a variety of topics, knowledge areas and research types. The full report is available online [6].

<table>
<thead>
<tr>
<th>Total number of search results</th>
<th>1278</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number after duplicate removal</td>
<td>1039</td>
</tr>
<tr>
<td>Voted in by at least one reviewer</td>
<td>384</td>
</tr>
<tr>
<td>Voted in by majority</td>
<td>83</td>
</tr>
</tbody>
</table>

**RQ1: What research topics are being addressed?**

We used a variety of methods to structure and model the research topics of the 83 publications that were voted-in. Fig. 1 shows a simple weighted word cloud that was generated from the publication abstracts. It was created with Tagxedo², which used a stemming algorithm to filter the textual input. The goal of this image was to gain a first impression of the topical content of the publications.

![Weighted word cloud from the original abstracts of voted-in publications.](http://example.com/weightcloud.png)

Figure 1: Weighted word cloud from the original abstracts of voted-in publications.

The next, more in depth analysis method used is called Topic modeling. This is a method for analyzing large data sets to elicit commonalities, in this case topics, which are clusters of words that frequently occur together in the data [9]. It is a “probabilistic model for uncovering the underlying semantic structure of a document collection” [10]. We utilized the Machine Learning for Language Toolkit (MALLET)³, that is popularly used for machine learning applications to text, including classification, clustering, natural language

²[www.tagxedo.com](http://www.tagxedo.com)
³[http://mallet.cs.umass.edu](http://mallet.cs.umass.edu)
processing, and topic modeling. The purpose of performing topic modeling on the dataset (consisting of the abstracts of the voted-in publications) was to investigate what the ‘hot topics’ are in the domain of Software Engineering for Sustainability.

In order to be able to run the dataset through MALLET, we preprocessed the abstracts to be represented as a list of words associated with each publication. The dataset was also imported into MALLET using functionality that removed stop words and took into account basic word stemming. As our dataset of 83 documents was small, we only ran the trainer for 100 iterations. The goal was a qualitative corpus exploration [10], so we chose the top 10 topics for consideration. The modeling of the abstracts resulted in the topic clusters of the future of society, urban architecture and integration, energy efficiency, life cycle assessment, environmental management, smart grids, cloud services, carbon consumption, traffic strategies, and virtualization. Please refer to the complete protocol [6] for the full list of topic cluster keywords and further detailed analysis graphs.

Based on the word content of each abstract, and the output from the MALLET topic model, we were able to relate abstracts to the elicited topics. We pruned each topic down to 6 keywords that were most characteristic of the abstracts that belonged to each topic. Fig. 2 shows the resulting clusters of papers and the topics they belong to. The numbers in this graphic refer to the numbers in Tab. 4.

**Figure 4: Distribution of the publications over time.**

**RQ2: How have these research topics evolved over time?.**

The answer to RQ 2 needs a prelude on how the publications, and therefore our data points, are distributed over time. As depicted in Fig. 4, there were 40 new relevant publications in the last two years alone. A description of the evolution of the topics over time is somewhat limited, as this constitutes a majority of publications that are in domain of Software Engineering for Sustainability.

**RQ3: How is sustainability support performed?.**

There is a wide range of models, methods, frameworks, and tools that are proposed in the publications and used in research. They include standard software engineering support (like goal modeling and service modeling) as well as general purpose methods (like interviews and statistics) as well as more domain-specific methods from systems engineering (life cycle assessment), geosciences (global position system) and the energy domain (measuring devices).

- Software engineering methods & tools: goal modeling \([k1, c2, p3]\), stakeholder modeling \([e2, w3]\), agent modeling \([d1]\), service modeling \([g2]\), processes modeling \([d2, f2, l2]\), simulation \([c3, l1]\)
- General purpose methods & tools: interviews \([t1]\), statistics \([p2]\), surveys \([u1]\)
- Systems Engineering: life cycle assessment \([f1, l1]\)
- Geo Sciences: global position system, internet map services \([m1, w1]\)
- Earth Sciences: environmental information systems \([p1, q3]\)
- Urban Planning: simulation \([i1, d1]\)
- Energy Management: measuring devices [v2, k2, d3], traffic management systems [w1]

This plethora of used approaches only leads to the conclusion that there are many different roads being explored but there are no methods and models yet that can be considered as established for SE4S.

**RQ4: Which models and methods are used in practice?**

In order to report on which means are used in practice (as opposed to being only proposed as a solution in a publication), when considering Fig. 3 it is clear that there are not many publications of the research facet Evaluation or Experience. Evaluation papers are [v2, p2, a3, r3, d4], and Experience papers are [b1, c1, r2, s5]. Due to this low number, it does not make sense to draw further conclusions on the state of practice. It also leaves the question of whether the topic is not really triggering a state of practice at all or whether it is simply not published much on yet.

**RQ5: Which research methods have been considered in the contributions?**

In Fig. 5, we display the relation of knowledge areas [8] to research facets [7].

As represented in Fig. 5, there are many contributions of the type Exploratory and Solution, but on the other hand none of the type Opinion and very few in Experience and Evaluation. This indicates a young and still somewhat immature research area which needs to perform more evaluation and encourage practitioners to report on experiences.

**RQ6: Which application domains have been considered?**

As not all publications are considering an explicit application domain, but more than 50% have a generic approach across application domains, we classified papers either according to an application domain or a focus domain to be able to differentiate them in categories. As opposed to the topic clusters discussed in RQ 1, these domains were not extracted automatically, but assigned manually by the researchers. Furthermore, a subsequent mapping between the automatically extracted research topic clusters and the application domains did not lead to significant correlation, thereby undermining the fact that they are worth distinguishing. We found ten such domains:

- Software Engineering & Lifecycle: Publications that do not refer to a specific application domain but presented generic approaches related to software engineering and the software lifecycle.
- Energy Efficiency: Publications that dedicate their work specifically to energy efficiency topics.
- Services, Mobile & Cloud: Publications that research topics in a service-oriented paradigm, often including mobile aspects and/or cloud computing, including research that monitors and improves traffic in cloud computing.
Table 3: Number of Voted-in papers according to Application and Focus Domain

<table>
<thead>
<tr>
<th>Application / Focus Domain</th>
<th>Publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineering &amp; Lifecycle</td>
<td>22</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>5</td>
</tr>
<tr>
<td>Services, Mobile &amp; Cloud</td>
<td>10</td>
</tr>
<tr>
<td>Business &amp; Economics</td>
<td>5</td>
</tr>
<tr>
<td>Systems Engineering &amp; ICT</td>
<td>12</td>
</tr>
<tr>
<td>ULS Green Computing</td>
<td>7</td>
</tr>
<tr>
<td>Mechanics &amp; Manufacturing</td>
<td>3</td>
</tr>
<tr>
<td>Nature &amp; Agriculture</td>
<td>5</td>
</tr>
<tr>
<td>Metropolitan Areas &amp; Housing</td>
<td>9</td>
</tr>
<tr>
<td>Software Engineering Education</td>
<td>5</td>
</tr>
</tbody>
</table>

- Business & Economics: This focus domain includes publications on business processes and organizational issues as well as globalization.
- Systems Engineering & ICT: Many contributions go over the boundaries of software, but consider ICT and whole systems, leading to a broader application of the principles of sustainability.
- ULS Green Computing: (Ultra) Large-scale systems have become a focus in computing as optimization of software on that level can potentially have a big impact on the overall resource consumption of ICT.
- Mechanics & Manufacturing: Few contributions specifically address green (re-)manufacturing.
- Nature & Agriculture: This focus domain entails developing systems for supporting sustainability in agriculture as well as improving environmental modeling for monitoring nature and gaining insights on related data points and clusters (e.g., on climate change).
- Metropolitan Areas & Housing: A number of approaches targets urban management, including traffic, transportation, smart homes, and urban ecosystems.
- Software Engineering Education: Last but not least, five publications address how to incorporate the topic of sustainability into software engineering education.

The application domains and focus areas that have been considered in the publications are listed in Tab. 3. The publications are all referenced and clustered according to these domains in Tab. 4.

Figure 6 briefly summarizes the mapping of the manual classification of publications to Application Domains to the resultant topic cluster modeling classification of publications to the Topic Clusters. This figure shows the distribution of topics that occur in each of the application domains of the SE4S publications. Fig. 6 shows that the research topic clusters on the future of society, urban architecture and integration, energy efficiency, life cycle assessment, environmental management, smart grids, cloud services, carbon consumption, traffic strategies, and virtualization (as in Fig. 2) do not significantly correlate with the distribution of the application domains. However, alignments are perceivable for a small subset.

RQ7: Which research groups are most active in researching the topic and what is the distribution between academics and practitioners?

The network graph was constructed based on the authors of the 83 voted-in publications. It was generated using ManyEyes, an experimental Visualization web service by IBM.

Apart from that there are a 197 unique authors, but as was shown in Fig. 4, most have been active in the last 3 years. We found 56 connected subgraphs (some of which were single author nodes), three of which are major research clusters, where authorship spans more than one or two papers. These are also fairly globally distributed, with even some intercontinental collaborations. These three interesting subgraphs are shown in Fig. 7 (a larger version can be found in [6]). The distribution of publications between academia and industry is currently unbalanced with roughly 80% of reported evidence coming from academia, the rest being distributed between industry and mixed collaborations. This distribution was derived from the affiliation that the authors provided for the publication.

Figure 7: Three largest author network subgraphs in detail; a larger version can be found in [6]

4. DISCUSSION & THREATS

This section provides a discussion of the results and of the threats to validity for this study.

Completeness of Results.

During the voting period, there were suggestions by reviewers for other papers they knew of, which they had expected to show up in the results but did not, were carefully checked by the principal researchers.
The work was carried out by two principal researchers, three supporting researchers, an internal reviewer and two external reviewers over the course of 4 months. Please refer to the full report for additional information [6].

The topic of SE4S has received wide-spread attention in the software engineering community over the past few years. Due to the fact of being a relatively new area of research, there is rather little reported evidence of establishment in practice. At the same time, industry has recognized the topic and use the term sustainability all over, reminding of the Green IT hype, but now broadened to sustainability. As Green IT practices are by now further established in practice, hope remains that the same will come true for other sustainability practices. The low number of evaluation and experience papers in the reported evidence also suggest that the research area including its solutions are still somewhat immature. Furthermore, the fair distribution over a range of journals and venues indicates that the research community is still forming. However, the large number of topic clusters, focus areas and application domains indicates that research is being conducted in broad coverage of the area of SE4S. The following list sums up the major conclusions from the reported evidence.

Data Synthesis.

The data synthesis was performed partially automatic, partially manually. For the automatic part, we relied on topic cluster modeling and generated graphics to aggregate information in a form that is more easily perceived by human cognition that pure numbers. These tools have been used widely throughout this and other research communities and we trust they are reliable and produce valid results. For the manual part, i.e. the classification according to knowledge areas, research facets, and focus domains, we performed the data synthesis to the best of our knowledge. However, human judgement is always subjective to a certain degree, so other researchers might have chosen slightly different terms for application domains or keywords for methods and frameworks. This threat was mitigated by internal and external reviews.

Figure 6: Correlation of Application Domains to Topic classification

One reason for why some of the expected results had not shown up in the automatic search results was that they had not applied to the first part of the search string. The first part (sustainab* OR ecolog* OR green) required an explicit link of the research to sustainability concerns. This was not the case for many energy efficiency publications, therefore these may be underrepresented in the results of our study.

Another reason for missing expected results was that papers did not match the second part of the search query (software engineering OR requirement* engineering OR requirement* specification OR software specification OR system specification). We encountered a few papers, for example, from the GREENS workshop at ICSE 2013, that we consider relevant to the research area, but did not show up in the results because they used other terms like 'software quality' to classify their research. We conclude that some software engineering researchers who work in the analyzed area of investigation are missing from the results because they used more specific terms and did not include the more general terms 'software engineering', or 'software specification', for example in energy efficiency and software quality.

Search Engine Correctness.

Each of the information sources (i.e. the indexing systems and digital libraries listed in 2) evaluated boolean search queries according to their own mechanism. Therefore, when an information source did specify query rules, the search string was adapted accordingly. An issue that was prevalent in some information sources was that there were different search results for semantically equivalent queries based on the order of operations. To this extent, we can not guarantee for the quality of the automatically executed queries in those information sources. However, as we used a wide range of search engines, we hope we have mitigated that effect as far as possible.

Manual Additions.

We have manually added a small set of proceedings of venues that are very relevant to the research area to the set of automatically retrieved papers due to the fact that the newest (2013) edition of these conferences and workshops was not yet indexed by the search engines. We did this in order to make the selection pool for relevant papers as up-to-date as possible. In our understanding, this does not introduce a strong bias for the research but rather merely offers a potential qualitative improvement of the results.
RQ1. The research topic clusters that have been addressed include a variety of aspects ranging across the future of society, urban architecture and integration, energy efficiency, life cycle assessment, environmental management, smart grids, cloud services, carbon consumption, traffic strategies, and virtualization. The majority of publications are in the knowledge areas of Software Design, Engineering Management, Models and Methods, Process, Quality, and Requirements.

RQ2. The evolution of the research topics over time reveals a strong general development over the last four years, especially in the topic clusters of future of society, life cycle assessment, and energy efficiency.

RQ3. Sustainability support is performed by a variety of models and methods that include general purpose (interviews, statistics, surveys), software engineering (goals, stakeholders, services, processes), systems engineering (LCA), as well as methods from geo sciences, earth sciences, urban planning, and energy management.

RQ4. The usage of the approaches in practice is very limited in the reported evidence.

RQ5. The most prominent research type facets were Exploratory and Solution.

RQ6. The application domains that were predominantly considered are Software Engineering and Lifecycle, Systems Engineering and ICT, Energy Efficiency, Mobile Services and Cloud, Business and Economics, UL Computing, Mechanics and Manufacturing, Nature and Agriculture, Metropolitan Areas and Housing, and Software Engineering Education.

RQ7. There are three rather active research groups but research is performed all over the world and distribution between academia and industry is currently unbalanced with roughly 80% of reported evidence from academia, the rest distributed between industry and mixed collaborations.

The aggregation of results and overviews in graphics and tables as well as the compact table of included publications may be considered as a compact overview of the field of Software Engineering for Sustainability.

Due to the facts that SE4S has significantly gained importance over the past few years and that it has been intensely researched by a world-wide community, we conclude that there is need for a future roadmap that identifies the major research gaps and outlines promising options of how to fill these gaps.

Acknowledgements. We would like to thank Daniel Méndez Férandez and Marcela Genero for serving as external reviewers and for helpful feedback. This work is part of the DFG EnvironSSE project (grant number PE2044/1-1).

6. REFERENCES

Table 4: Voted-in papers according to Application Domain

<table>
<thead>
<tr>
<th>Ref</th>
<th>Year</th>
<th>Author</th>
<th>Title</th>
<th>Output Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>[a1]</td>
<td>1989</td>
<td>Cohill</td>
<td>The human factors design process in software development</td>
<td>3rd Intl. Conf. HCI</td>
</tr>
<tr>
<td>[a2]</td>
<td>2011</td>
<td>Johann et al.</td>
<td>Sustainable development, sustainable software, and sustainable software engineering: An integrated approach</td>
<td>IC Humanities, Science Engineering Research</td>
</tr>
<tr>
<td>[a3]</td>
<td>2011</td>
<td>Mahemeh, Saced</td>
<td>Application of a composite process framework for managing green ICT application development</td>
<td>Handbook of Research on Green ICT</td>
</tr>
<tr>
<td>[a5]</td>
<td>2011</td>
<td>Shenoy, Eeratta</td>
<td>Green software development model: An approach towards sustainable software development</td>
<td>IEEE India Conference</td>
</tr>
<tr>
<td>[a6]</td>
<td>2012</td>
<td>Agarwal et al.</td>
<td>Sustainable approaches and good practices in green software engineering</td>
<td>J of Research and Reviews in Computer Science</td>
</tr>
<tr>
<td>[a7]</td>
<td>2012</td>
<td>Hindle</td>
<td>Green mining: Investigating power consumption across versions</td>
<td></td>
</tr>
<tr>
<td>[a8]</td>
<td>2012</td>
<td>Johann et al.</td>
<td>How to measure energy-efficiency of software: Metrics and measurement results</td>
<td>W GREENS</td>
</tr>
<tr>
<td>[a9]</td>
<td>2012</td>
<td>Liani, Bugioni</td>
<td>Measuring Software Sustainability From a Process-Centric Perspective</td>
<td>W on Software Measurement</td>
</tr>
<tr>
<td>[b1]</td>
<td>2012</td>
<td>Penzenstadler et al.</td>
<td>Sustainability in software engineering: A systematic literature review</td>
<td>ICCE</td>
</tr>
<tr>
<td>[b3]</td>
<td>2013</td>
<td>Lago et al.</td>
<td>Exploring initial challenges for green software engineering</td>
<td>SIGSOFT SE Notes</td>
</tr>
<tr>
<td>[b7]</td>
<td>2013</td>
<td>Penzenstadler et al.</td>
<td>Who is the advocate? Stakeholders for sustainability</td>
<td>W GREENS</td>
</tr>
<tr>
<td>[b8]</td>
<td>2013</td>
<td>Rohr, Richardson</td>
<td>A proposed recommender system for eliciting software sustainability requirements</td>
<td>W USER</td>
</tr>
<tr>
<td>[b9]</td>
<td>2013</td>
<td>Kocak et al.</td>
<td>The Impact of Improving Software Functionality on Environmental Sustainability</td>
<td>ICT4S</td>
</tr>
<tr>
<td>[b10]</td>
<td>2013</td>
<td>Rohr, Richardson</td>
<td>Sustainability requirement patterns</td>
<td>W RePa</td>
</tr>
<tr>
<td>[b11]</td>
<td>2013</td>
<td>Sventinovic</td>
<td>Strategic requirements engineering for complex sustainable systems</td>
<td>J Systems Engineering</td>
</tr>
<tr>
<td>[b12]</td>
<td>2013</td>
<td>Penzenstadler</td>
<td>Towards a definition of sustainability in and for software engineering</td>
<td>SAC</td>
</tr>
</tbody>
</table>

Application / Focus Domain: General Software Engineering & Lifecycle

<table>
<thead>
<tr>
<th>Ref</th>
<th>Year</th>
<th>Author</th>
<th>Title</th>
<th>Output Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>[c1]</td>
<td>2010</td>
<td>Agarwal et al.</td>
<td>Internet map services: new portal for global ecological monitoring, or geodata junkyard?</td>
<td>IC on Computing for Geospatial Research &amp; Application</td>
</tr>
<tr>
<td>[c3]</td>
<td>2011</td>
<td>Ainsel et al.</td>
<td>Toward sustainable software engineering</td>
<td>IC on Software Engineering</td>
</tr>
<tr>
<td>[c7]</td>
<td>2012</td>
<td>Razavian et al.</td>
<td>Modeling to support communication and engineering of service-oriented software</td>
<td>W Software Services and Systems Research</td>
</tr>
<tr>
<td>[c8]</td>
<td>2013</td>
<td>Atkinson, Schulze</td>
<td>Towards application-specific impact specifications and GreenSLAs</td>
<td>W GREENS</td>
</tr>
</tbody>
</table>

Application / Focus Domain: Services, Mobile & Cloud

<table>
<thead>
<tr>
<th>Ref</th>
<th>Year</th>
<th>Author</th>
<th>Title</th>
<th>Output Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>[d1]</td>
<td>1994</td>
<td>Jones</td>
<td>Globalisation of software supply and demand</td>
<td>J Software Engineering</td>
</tr>
<tr>
<td>[d2]</td>
<td>2007</td>
<td>Heimakristian</td>
<td>Business process ontology and software service models for environmentally sustainable manufacturing enterprises</td>
<td>ICHIT</td>
</tr>
<tr>
<td>[d4]</td>
<td>2011</td>
<td>Harmon, Demirkan</td>
<td>The corporate sustainability dimensions of service-oriented information technology</td>
<td>SHI Global Conference</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Title</td>
<td>Conference/Journal</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td>Atallah</td>
<td>Systematic methodology for developing advanced complex systems</td>
<td>Conf. CE and CALS</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Rasmussen</td>
<td>From human-centred to human-context centred approach: looking back over ‘the hills’, what has been gained and lost?</td>
<td>AI &amp; Society</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Mitrea et al.</td>
<td>Sustainability ICT visions and their embedding in technology construction</td>
<td>Information Communication &amp; Society</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Sussa</td>
<td>Green Software</td>
<td>J UPGRADE</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Cunley</td>
<td>Towards sustainability: Harnessing computing and communications for a better future</td>
<td>Symp. on Computers and Communications</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Penzenstadler et al.</td>
<td>Green Performance Indicators Aggregation through Composed Weighting System</td>
<td>IOT-GLOW</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Chen &amp; Kaizman</td>
<td>Architecting ultra-large-scale green information systems</td>
<td>W GREENS</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Salon et al.</td>
<td>Initial explorations on design pattern energy usage</td>
<td>W GREENS</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Liang et al.</td>
<td>Virtualization Maturity Reference Model for Green Software</td>
<td>ICCEC-T</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Salin et al.</td>
<td>Toward power reduction through improved software design</td>
<td>C Energytech</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Boschoven, Bloem</td>
<td>Pilot Result Monitoring Energy Usage by Software Slides</td>
<td>ICT4S</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>Dong et al.</td>
<td>Research on the development of green product life cycle analysis tool</td>
<td>J China Mechanical Engineering</td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Zhou et al.</td>
<td>Green remanufacturing engineering in structural machinery based on reverse engineering</td>
<td>IC on Security Technology</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Johansson et al.</td>
<td>Discrete event simulation to generate requirements specification for sustainable manufacturing systems design</td>
<td>W on Performance Metrics for Intelligent Systems</td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td>Ramalho-filho et al.</td>
<td>Use of geographic information systems in (planning) sustainable land management in Brazil: potentials and user needs</td>
<td>Int. J C ICT Journal</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Argoli</td>
<td>An overview of model integration for environmental applications - components, frameworks and semantics</td>
<td>J Env. Modeling &amp; Software</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Bingshan &amp; Wang</td>
<td>Knowledge-based environmental information system for sustainable development of wetland area</td>
<td>IC on Software Engineering and Data Mining</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Easterbrook et al.</td>
<td>Second International Workshop on Software Research and Climate Change</td>
<td>W Software Research &amp; Climate Change</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Grunfeld, Houghton</td>
<td>Using ICT for Climate Adaptation and Mitigation through Agro-Ecology in the Developing World</td>
<td>ICT4S</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Robinson et al.</td>
<td>SUNtool - a new modelling paradigm for simulating and optimising urban sustainability</td>
<td>J Solar Energy</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Albertao et al.</td>
<td>Measuring the Sustainability Performance of Software Projects</td>
<td>IC on e-Business Engineering</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Shishkov et al.</td>
<td>On the application of autonomous and context-aware computing to support home energy management</td>
<td>IC on Enterprise Information Systems</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Kraines</td>
<td>Integrating distributed computational models as dynamic expressions of knowledge: the case for evaluating measures for urban ecosystem sustainability</td>
<td>W on Software Knowledge</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Schrammel et al.</td>
<td>FORE-Watch: The Clock That Tells You When to Use: Persuading Users to Align Their Energy Consumption with Green Power Availability</td>
<td>IC Ambient Intelligence</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Blumenthal</td>
<td>Building Sustainable Smart Homes</td>
<td>ICT4S</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Penzenstadler, Fleischmann</td>
<td>Teach sustainability in software engineering?</td>
<td>CSEE&amp;CT</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Ahmed, Shuaib</td>
<td>Incorporating Green IT concepts in undergraduate software requirements engineering course: An experience report</td>
<td>C Information Systems and Technologies</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Penzenstadler et al.</td>
<td>Jumpt start sustainability in seminars: hands-on experience in class</td>
<td>CSEE&amp;CT</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td>Penzenstadler et al.</td>
<td>University meets industry: Calling in real stakeholders</td>
<td>CSEE&amp;CT</td>
<td></td>
</tr>
</tbody>
</table>
4 Towards a Definition of Sustainability in & for Software Engineering
Towards a Definition of Sustainability
in and for Software Engineering

Birgit Penzenstadler
Institut für Informatik
Technische Universität München
Germany
penzenst@in.tum.de

ABSTRACT
Sustainability is not supported by traditional software engineering methods. This lack of support leads to inefficient efforts to address sustainability or complete omission of this important concept. Defining and developing adequate support requires a commonly accepted definition of what sustainability means in and for software engineering. We contribute a description of the aspects of sustainability in software engineering.

Categories and Subject Descriptors
D.2.1 [Software Engineering]: [sustainability, environment, requirements engineering, quality assurance, guidance]

1. MOTIVE: COMMON UNDERSTANDING
Although many people in software engineering (SE) are by now aware of the general definition of sustainability as “the capacity to endure” and sustainable development as “meeting the needs of the present without compromising the ability of future generations to meet their own needs” [10], there still is no concrete guidance for the different aspects of sustainability that are observable from the point of view of software engineering. This is also due to the fact that the concept of sustainability does not become adequately tangible from that definition. The consequence is that, as the topic of sustainability is now receiving attention in software engineering, our discipline tries to contribute. However, without a thorough and commonly accepted definition of what sustainability means applied to software engineering, our individual contributions remain somewhat insular and isolated. In order to realize green computing, we need such a definition.

Problem: Traditional software engineering has not fully supported sustainability as a relevant, first-class concern. We software engineers approach specific topics that have to do with sustainability in our discipline, for example, green IT, efficient algorithms, smart grids, agile practices and knowledge management, but we lack a common understanding of the concept of sustainability and if and how it can be encompassingly applied to software engineering.

Contribution: In this paper, we describe the different aspects of sustainability from a point of view of software engineering and exemplarily illustrate their realisation in activities. This provides software engineers with a preliminary guideline on how sustainability can be incorporated into their daily practice, for example during requirements engineering and quality assurance.

Outline: After providing background on frameworks for sustainability, the relation between ICT and sustainability, and earlier approaches to definitions in the context of SE in Sec. 2, we contrast the aspects of our definition of sustainability in SE and relate it to the referenced frameworks in Sec. 3. We conclude with future work in Sec. 4.

2. BACKGROUND & FOUNDATIONS
The background for the presented definition approach is structured into frameworks for defining sustainability, the relation between ICT and sustainability, and earlier approaches to defining sustainability in the context of software engineering.

2.1 Frameworks for Defining Sustainability
The following frameworks are designed to serve for defining sustainability in general, without a specific reference to software systems. Dobson [3] suggests a framework for the comparison of sustainability notions that poses conceptual questions any conception of sustainability must answer: What to sustain? Why to sustain? Who/what concerned? Substitutability allowed? Burger and Christen [1] propose a capability approach of sustainability. Their methodological approach is to formulate adequacy conditions for concepts of sustainability, illustrate a categorial framework with the required general concepts, and propose a conception of sustainability based on the capability approach. They consider the idea of sustainable development as a problem-solving strategy (for the “developmental dilemma” [10]and their adequacy conditions are future-orientation, normative power, justice, universality, limited natural conditions, and high-level strategic actions [1, p. 788]. The developed approach is supposed to serve as basis for empirical research in development studies. Roberts et al. [8] present the Framework for Strategic Sustainable Development that is now mainly promoted by the Swedish NGO ‘The Natural Step’. Their objective is to show how the increasing number of tools and approaches to develop sustainability (e.g., the ISO14001, Life Cycle Assess-
ment, and Ecological Footprinting) relate to each other and build on each other when used for planning for sustainability. Following a strictly analytic approach in a theoretic contribution to the discourse in sustainable development, Christen and Schmidt [2] propose a Formal Framework for Conceptions of Sustainability that intends to solve the problem of arbitrariness. The framework consists of modules that serve to help to elaborate the elements answering the ‘what’ and ‘how’ questions.

2.2 Relation between ICT & Sustainability
Hilty, Lohmann, and Huang [4] provide an overview of the fields of ICT in the service of sustainability: Environmental Informatics, Green IT, and Sustainable Human-Computer Interaction. As technological efficiency alone will not produce sustainability (cf. Jevon’s paradoxon), sustainable development requires a combination of efficiency and sufficiency strategies, inter alia by decoupling economic growth from environmental impacts and from the use of natural resources.

Furthermore, Hilty, Arnfalk, Erdmann et al. [5] analyse the relevance of information and communication technologies for environmental sustainability. They present the impacts of ICT on environmental sustainability on different levels: first order effects like increasing electronic waste streams, second order effects such as improved energy-efficiency of production, and third order effects like a product-to-service shift or rebound effects in transport.

2.3 Sustainability in the Context of SE
Penzenstadler et al. [7] provide a systematic literature review that points to two definitions of sustainability in the context of software engineering, i.e. by Mahaux et al. [6] and by Naumann et al. [9].

Mahaux et al. simply use the Brundtland definition [10] plus the statement that IT changes behavior and therefore has considerable effect on society and environment, which is supported by green IT concepts and analysis of the usage processes of a software system [6].

Naumann et al. define “green and sustainable software” such that “direct and indirect negative impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software (...) ha(ve) a positive effect on sustainable development”, and “Green and Sustainable Software Engineering” such that the “negative and positive impacts on sustainable development (...) are continuously assessed, documented, and used for a further optimization of the software product” [9, p. 296].

These two works show that the term sustainability is strongly dependent on the taken perspective, and even the second definition cannot be operationalised. However, this is necessary if SE researchers want to contribute. Consequently, scope and context have to be clearly defined to be able to make any statements.

3. ASPECTS OF SUSTAINABILITY IN SOFTWARE ENGINEERING
Applying the principles of system thinking, sustainability can be defined as preserving the function of a system over a defined time span. Consequently, there are three variables, namely system, function, and time, that need to be defined for setting the scope for a discussion on sustainability. An absolute definition has fixed variables, while a relative definition chooses the variables according to the context. For sustainability aspects in software engineering, the author’s aim is to relate a relative definition of sustainability in the context of software engineering to an absolute one. This relation allows to discuss the definition in comparison to the definitions by other disciplines.

3.1 Absolute and relative definition
An absolute definition of sustainability has fixed variables: the time span is various generations, the function is a satisfaction of needs, the system is humanity in its ecosystem. For a relative definition, the variables of system, function and scope still have to be chosen. This choice for the variables can be made independent of a “higher value”, and just signify plain capacity to endure. For example, I can have a manufacturing site for cars running in a way that all orders are satisfied (function “develop cars in time for orders”) but still be economically, socially, and ecologically unsustainable in various ways. If I want to avoid the latter, the function has to refer to an absolute definition, e.g., “develop cars in time for orders respecting the sustainability guidelines defined by our company”.

Applied to software engineering, the differentiation can be made by distinguishing software (engineering) for sustainability, which is related to the absolute definition, and sustainable software (or sustainability in software engineering), which is related to the relative definition.

3.2 Defining aspects in software engineering
Apart from this differentiation of sustainability in software and software for sustainability, we distinguish four aspects. The first two focus rather on the developing company and its processes, while the latter two have the system under development in scope. These aspects are derived from iterated discussions with various experts on sustainability and they shall serve as structuring means for discussing and supporting sustainability activities rather than as an apodictic differentiation.

Development Process Aspect.
Sustainability during the initial software development process means development with responsible use of ecological, human, and financial resources.

System: a software system development company (including its staff as well as its operational environment with equipment)

Function: to perform software development with minimized environmental impact, a sufficient economic balance, and good working conditions

Time: the planning period of the long-term business plan of a software development company; depends on the company’s size and the general duration of development projects, may vary from months to years

Maintenance Process Aspect.
Sustainability of the software system during its maintenance period until replacement by a new system includes continuous monitoring of quality and knowledge management.

System: the maintenance department of the software system development company
Function: to maintain and evolve a software system with minimized environmental impact, a sufficient economic balance, and well-managed knowledge

Time: planning period of the lifetime of the system; depends on the contracted maintenance period for projects, may vary from a couple of years to decades

System Production Aspect.
Sustainability of the software system as product with respect to its use of resources for production is achieved, for example, by using green IT principles, sustainably produced hardware components, and optimizing the required logistics for assembly, etc.

System: a conjoint of developing and supplier companies that collaborates for assembling the specific product

Function: to produce a system with minimized environmental impact and a sufficient economic balance

Time: according to the project plan for production; depends on the system’s size and complexity, may vary from a couple of months to years

System Usage Aspect.
Sustainability in the usage processes within the application domain triggered by the software system as product takes into account responsibility for the environmental impact and designing green business processes.

System: the system under analysis with operational environment & business context in its application domain

Function: to maintain and evolve a software system with minimized environmental impact, a sufficient economic balance, and social responsibility

Time: estimated lifetime of the system; depends on system’s size and complexity; varies from years to decades

The lifecycle of a system of course continues through to replacement and disposal, but for software the activities of reengineering and refactoring are already considered within the Maintenance Process Aspect.

3.3 Relation to frameworks
In order to give a justification for our definition, we compare it definition to the others from background section and set them in context within the frameworks.

The definition by Mahaux et al. [6] is basically included in our’s as they rely on Brundtland [10] which we consider as one possible version of an absolute definition of sustainability. The definition by Naumann et al. [9] is expressed within the four aspects, only expressed differently. By this differentiation, our definition makes it easier to assign actions to supporting the various notions of sustainability.

Considering Burger and Christen’s [1] adequacy conditions (future-orientation, normative power, justice, universality, limited natural conditions, and high-level strategic actions), the definition needs to be extended to satisfy all of the required information. However, this can only be done within a specific context, for example, a specific company of for a specific system type, most likely only within a specific country. Consequently, also the categories named in [1] can only be fulfilled by the referenced absolute definition.

Christen and Schmidt’s [2] framework modules (problem, justice, integration, criteria, and transformation) are equally given by the absolute definition. The definition may be integrated into the framework by Robèrt et al. [8] and, finally, Dobson’s [3] questions (what? why? who? in what respect?) are all satisfied by the aspect definitions above due to the description of the variables of system, function, and time.

Certainly, this short justification is not a sufficient discussion of the adequacy of our definition, but it is an indicator that it complies to the requirements for definitions in general sustainability research. Thus, future SE sustainability research may build upon this definition.

4. CONCLUSION AND FUTURE WORK
This paper presents a definition of the aspects of sustainability and for software engineering. Thereby, for software engineering is how to make SE itself more sustainable and in software engineering is how we improve the sustainability of the systems we develop. The aspects have, in a next step, to be extended with a catalogue of actions to realize and/or improve them. We are currently elaborating a case study with a software development company where we align the process and a system under development with a set of sustainability principles in a pilot project.

As future work, we envision an encompassing approach for software engineering that supports building sustainability into software products from early requirements engineering on. This will also include a corresponding assessment model.

Acknowledgement
The author would like to thank Lorenz Hilty for inspiring discussions as well as Bill Tomlinson and Debra Richardson for feedback on early ideas.

5. REFERENCES


5 ICT4S 2029: The Systems Supporting Sustainability in 15 Years?
ICT4S 2029: What will be the Systems Supporting Sustainability in 15 Years?

Birgit Penzenstadler (bpenzens@uci.edu), Bill Tomlinson (wmt@uci.edu), Eric Baumer (ericpsb@cornell.edu), Marcel Pufal (mpufal@uci.edu), Anika Raturi (araturi@uci.edu), Debra Richardson (djr@uci.edu), Baki Cakici (cakici@sics.se), Ruzanna Chitchyan (rrc256@leicester.ac.uk), Georges Da Costa (dacosta@irit.fr), Lynn Dombrowski (lynn.dombrowski@uci.edu), Malin Picha Edwardsson (picha@kth.se), Elina Eriksson (elina@kth.se), Xavier Franch (franch@essi.upc.edu), Gillian R. Hayes (gillianrh@ics.uci.edu), Christina Herzog (herzog@irit.fr), Wolfgang Lohmann (wolfgang.lohmann@gmail.com), Martin Mahaux (martin.mahaux@fundp.ac.be), Alistair Mavin (Alistair.Mavin@Rolls-Royce.com), Melissa Mazmanian (mmazmani@uci.edu), Sahand Nayebaziz (snayebaz@uci.edu), Juliet Norton (jnnorton@uci.edu), Daniel Pargman (pargman@kth.se), Donald J. Patterson (dpj3@ics.uci.edu), Jean-Marc Pierson (pierson@irit.fr), Kristin Roher (kroher@uci.edu), M. Six Silberman (silberman.six@gmail.com), Kevin Simonson (simonsok@uci.edu), Andrew W. Torrance (torrance@ku.edu), André van der Hoek (andre@ics.uci.edu)

Communication Department & Information Science Department, Department of Informatics
Cornell University, USA
KTH/CSC/Media Technology & Interaction Design
Royal Inst. of Techn., Stockholm, SE
Department of Computer Science
University of California-Irvine, USA
University of California-Irvine, USA

Index Terms—sustainability, software systems, vision, fictional, requirements, research challenges

I. INTRODUCTION

At ICT4S 2013, the developing ICT4S research community developed a set of recommendations and list of challenges [1] that was derived from the research results presented and discussed at the conference. Statements that were provided by the speakers before the conference were compiled in a draft document that was then discussed in plenary sessions. These recommendations led to questions on topics including, but certainly not limited to, closure of material cycles to avoid hardware obsolescence, incentives for sustainable behavior, evolution of education for sustainability, and systems for sustainability assessment.

In addition to following recommendations and principles, research is often inspired by the work of others in related domains who have faced similar challenges, and by people who provide visions of the future, for example the SMART 2020 report [2] and the Vision 2050 [3]. The authors of such visions do not necessarily have to be visionaries in the sense of a domain expert in future studies, but instead may well be found within our growing ICT4S research community. Most of the visions in our research evolve somewhat organically, if not implicitly, and therefore a collection of future scenarios or visions to start such an organic evolution may be a good way to kickstart the discussion.

Abstract—Research is often inspired by visions of the future. These visions can take on various narrative forms, and can fall anywhere along the spectrum from utopian to dystopian.

Even though we recognize the importance of such visions to help us shape research questions and inspire rich design spaces to be explored, the opportunity to discuss them is rarely given in a research context.

Imagine how civilization will have changed in 15 years. What is your vision for systems that will be supporting sustainability in that time? Which transformational changes will have occurred in the mean time that allow for these systems? Is ICT even the right tool or does it contradict sustainability by making our world ever more complex? How can we make systems and our societies more sustainable and resilient by ICT4S?

This paper presents a compilation of fictional abstracts for inspiration and discussion, and provides means to stimulate discussion on future research and contributes to ICT4S community building.
Inspired by 1) a visit of Joseph Tainter [4], 2) a contribution to the alt.chi track in the ACM CHI Conference on Human Factors in Computing Systems [5] and 3) the research questions posed by Kates et al. [6], we sent out a call for fictional abstracts for papers on ICT4S systems that might appear in the ICT4S conference in the year 2029. This call also makes the process explicit that the abstracts would be compiled into a paper with the above title and submitted to the regular review process of this year’s edition of the ICT4S conference. A time span of 15 years (the fictional context of ICT4S 2029) seemed appropriate to allow for creativity and significant paradigm changes in the future scenarios but not too far in the future to result in abstracts that are totally disconnected from the present. The abstracts were required to be 150 words long, to provide a title for the fictional future paper, and to optionally include an image as well. Abstracts were then selected for inclusion based on their ability to represent a diversity of guiding research visions, their excitatory or provocative potential, and the likelihood of engendering conversations about the future of ICT4S. After the selection, authors of voted-in abstracts were invited to be co-authors of the paper and we performed cross reviews of an earlier version of the paper that were discussed via email within the whole group.

The contributions of this paper are threefold:

1) We present 19 fictional projects that may inspire future research.
2) We offer a way to stimulate discussion at the conference.
3) We contribute to the community building of ICT4S by showing links between topics and by establishing new research collaborations.

The rest of this paper is organized as follows: In Sec. II, we provide underlying definitions and related work. In Sec. III, we provide the compilation of fictional abstracts. And in Sec. V, we conclude the paper with a summary and an appeal for discussion.

II. BACKGROUND

This section provides the foundations for the concepts presented in the fictional abstracts. It includes definitions as well as related work on visions and challenges.

A. Definitions

This subsection provides definitions of the most important terms that are used throughout the abstracts and require a common understanding for the discussion of concepts for future research: sustainability, collapse, and resilience.

1) Sustainability: In general, sustainability is the “capacity to endure,” but interpreting this concept requires context. A popular definition of sustainable development was given by the UN as “meeting the needs of the present without compromising the ability of future generations to meet their own needs” [7]. Although it is not actionable, it is the most cited definition currently in use. Ultimately, sustainability depends on the population at large, so common conceptions of sustainability must be acknowledged. People sustain what they value, which can only derive from what they know [8]. For that purpose, Joseph Tainter has suggested that it is useful to pose the four questions with regard to sustainability:

1) Sustain what?
2) For whom?
3) How long?
4) At what cost? [4]

To understand the sustainability implications of ICT systems as well as other systems, it is useful to consider not only first order effects but also second and third order effects [9]. First order effects are direct effects of a system on its environment; in the case of a software system, first order effects would include energy usage, e-waste production, emissions caused by required infrastructure, etc. Second order effects are indirect effects or induction effects; in the case of software systems, second order effects could include changes in users’ resource consumption or consumer behavior. Third order effects are broader scale societal changes including rebound effects; a rebound effect would occur if the increased efficiency of systems tends to make us use even more systems which, in total, consume even more energy. All of these types of effect should be considered in an encompassing approach to supporting sustainability [10].

Tainter [8] draws seven lessons from his various studies linking social complexity to sustainability:

1) Sustainability is an active condition of problem solving, not a passive consequence of consuming less.
2) Complexity is a primary problem-solving tool, including problems of sustainability.
3) Complexity in problem solving is an economic function, and can reach diminishing returns and become ineffective.
4) Complexity in problem solving does its damage subtly, unpredictably, and cumulatively over the long term. Sustainability must therefore be a historical science.
5) Sustainability may require greater consumption of resources rather than less. One must be able to afford sustainability.
6) The members of an institution may resort to resilience as a strategy of continuity only when the option of sustainability is foreclosed.
7) A society or other institution can be destroyed by the cost of sustaining itself. [8]

Ostrom points out that we need a general common framework to be able to organize findings and knowledge with regard to analyzing the sustainability of social-ecological systems, because isolated knowledge does not accumulate, and proposes such a framework in [11].

2) Collapse: Collapse has been defined as “a lower degree of stratification and social differentiation; less economic and occupational specialization, of individuals, groups, and territories; less centralized control; that is, less regulation and integration of diverse economic and political groups by elites; less behavioral control and regimentation; less investment in the epiphenomena of complexity, those elements that define the concept of ‘civilization’: monumental architecture, artistic and literary achievements, and the like; less flow of information
between individuals, between political and economic groups, and between a center and its periphery; less sharing, trading, and redistribution of resources; less overall coordination and organization of individuals and groups; a smaller territory integrated within a single political unit” by [12, p. 4]. Rapid and powerful events such as nuclear attacks may cause events to unfold in a way that merits the term “apocalypse”; however, while various “tipping point” phenomena (according to [13], [14]) may cause non-linear changes to occur very rapidly (as is common in ecosystems), sustainability scientist Tainter [12], geographer Jared Diamond [13], and others note that collapse frequently occurs more gradually.

The concept of “collapse informatics” has been proposed as encompassing the “the study, design, and development of sociotechnical systems in the abundant present for use in a future of scarcity” [15]. If collapse is a potential outcome of industrial civilization’s “business as usual,” then collapse informatics could become part of the purview of ICT4S.

3) Resilience: While sustainability is the capacity to continue a desired condition or process, social or ecological, resilience can be defined as “the ability of a system to adjust its configuration and function under disturbance” [8, p. 92]. In social systems, resilience can mean abandoning sustainability goals and the values that underlie them. Consequently, sustainability and resilience can conflict [16].

Stokols et al. [17] suggest that taking into account the different types of capital — the material resources of economic, natural, human-made environmental, and technological capital as well as the human resources of social, human and moral capital — might be the solution to enhancing the resilience of human-environment systems.

B. Visions

There are a few vision papers worth mentioning that may be considered as related work, although they fundamentally differ in their approach to presenting visions and are more in-depth reports. However, they also have some commonalities: all visions require the research community as well as the overall population of the planet to substantially change their life styles. They demonstrate that a lot of effort is required to solve our current challenges for climate and resources. Furthermore, they outline a constructive approach and a feasible solution roadmap (as opposed to dystopian visions).

1) SMART 2020: The SMART 2020 report [2] outlines the transition to the low carbon economy in the information age. Half-way into the teens of this millennium, we should already be substantially further in implementing this roadmap, but it may still serve as reminder of what needs to be accomplished.

2) Vision 2050: In their Vision 2050 [3], the World Business Council for Sustainable Development (WBCSD) envisions a pathway with nine key areas of action: peoples’ values, human development, economy, agriculture, forests, energy and power, buildings, mobility, and materials. After the ‘turbulent teens’ (2010-20), each of these key areas has to go through a transformation in order to achieve the vision of a sustainable world in which 9 billion people can live well, and within the limits of the planet, by 2050.

3) Transition Engineering: More visions for the future are brought by an emerging discipline called Transition Engineering [18] that seeks to enable change from existing unsustainable systems to more sustainable ones by adaptation and filtering of demand to declining supply.

4) Resilient Citizens in the Information Society: Cameron [19] makes the case that sustainability presupposes the requirement for sustainability and, more concrete, that the “concept of a ‘sustainable’ Information Society incorporates the requirement for ICT (Information and Communication Technology) to support and promote viable communities of citizens” [19, Sec. 1]. She provides a framework with best practices for governments and for organizations aiming to support resilient citizens, but also for the citizens themselves who need to take over responsibility, including a failure survival strategy.

C. Challenges

The challenges presented in this section are intended to serve as background information and related work. Not all of them are addressed by the fictional abstracts, but we consider it important to provide them as context for the discussion on where research should pick its topics for investigation.

1) Limits to Growth: The book “Limits to Growth” by Donella Meadows et al. [20] and the Club of Rome¹ has been around since 1972, and presents the computer modeling of exponential economic and population growth with finite resource supplies. The book was updated 20 and 30 years later. Most scenarios resulted in an ongoing growth of population and of the economy until to a turning point around 2030. 40 years later, the Club of Rome noted that “Only drastic measures for environmental protection proved to be suitable to change this systems behaviour, and only under these circumstances, scenarios could be calculated in which both world population and wealth could remain at a constant level. However, so far the necessary political measures were not taken.” [21]

2) Central Research Questions for Sustainability Science: Kates et al. [6] offers seven central research questions for sustainability science that in part inspired the fictional abstracts presented in the paper at hand:

1) How can the dynamic interactions between nature and society — including lags and inertia — be better incorporated into emerging models and conceptualizations that integrate the Earth system, human development, and sustainability?

2) How are long-term trends in environment and development, including consumption and population, reshaping nature-society interactions in ways relevant to sustainability?

3) What determines the vulnerability or resilience of the nature-society system in particular kinds of places and for particular types of ecosystems and human livelihoods?

¹www.clubofrome.org
4) Can scientifically meaningful “limits” or “boundaries” be defined that would provide effective warning of conditions beyond which the nature-society systems incur a significantly increased risk of serious degradation?

5) What systems of incentive structures — including markets, rules, norms, and scientific information — can most effectively improve social capacity to guide interactions between nature and society toward more sustainable trajectories?

6) How can today’s operational systems for monitoring and reporting on environmental and social conditions be integrated or extended to provide more useful guidance for efforts to navigate a transition toward sustainability?

7) How can today’s relatively independent activities of research planning, monitoring, assessment, and decision support be better integrated into systems for adaptive management and societal learning? [6]

Like other use-inspired sciences, sustainability science includes “significant fundamental and applied knowledge components, and commitment to moving such knowledge into societal action” [22].

3) EROI: Energy continues to be one of our most important resources, while recent history seems to indicate that we have at least reached declining returns from our reliance on fossil fuels [12]. Due to increases in energy demands, we are now facing increasing energy costs as a result of the increased costs of producing energy.

The EROI (Energy Returned on Energy Invested) formula illustrates this phenomenon [23]:

\[
\text{EROI} = \frac{\text{Energy gained}}{\text{Energy required to gain that energy}}
\]

Put into simpler terms, as oil becomes more difficult to access, it costs more to retrieve it for energy production. Preliminary investigations of EROI for oil and gas have been reported on by [24] and a long-term assessment of oil and gas in the US has been provided in [25].

III. VISIONS OF THE FUTURE: ICT4S’ 2029

Here we present 19 fictional abstracts selected from those submitted in response to the open call. The abstracts were selected for inclusion based on their ability to represent a diversity of guiding research visions, their excitatory or provocative potential, and the likelihood of engendering conversations about the future of ICT4S. The selection was performed via a vote held by the first five authors of this paper, who initiated the call for abstracts. Some of the systems outlined in the abstracts may not even be achievable by 2029, and some might appear significantly before then. We present them clustered according to four major aspects that revealed themselves as themes: infrastructure, society, food, and waste. This is not the only possible categorization but it provides a simple classification to allow for an overview, as depicted in Fig. 1.

**INFRASTRUCTURE ABSTRACTS**

A. Let’s get into space!

*Georges Da Costa, Christina Herzog, Jean-Marc Pierson*

Earth is surrounded by the cold and (almost) unused outer space, while on earth we face global warming and limited space. We believe that space would be an alternative for activities currently done on earth and causing problems to our environment. The cooling of data centers is getting more problematic and even “green” solutions for cooling like using water or constructing data centers in cool regions of the earth are not acceptable for the environment.

Having storage for data in outer space might be a solution for the future as the expansion of our living space into outer space has already begun and is not causing huge difficulties any more. The aim of our paper is to show that cooling in outer space is an alternative to the polluting current way on earth. Additionally, a solution for energy problems will be investigated as running data centers in outer space may include the usage of solar energy to power them.

**B. Open (re)Source: An anonymous clearinghouse for sustainability-related corporate information**

*Bill Tomlinson, Debra Richardson, Ankita Raturi*

The flows of materials and energy through industrial processes have profound environmental implications. Trade secrets have typically prevented disclosure of information on these flows, and prevented knowledge of them from being taken into account in consumer decision making, thereby hampering consumer efforts to make environmentally-informed decisions. Over the past decade, though, the Open (re)Source system has enabled broad scale disclosure of this information.

Created and maintained by the online collective 6Anonymous, Open (re)Source is an online database of corporate information anonymously leaked by employees. This paper presents an ethnographic analysis of the Open (re)Source community, revealing two types of contributors: first, a group that are concerned about privacy and lawfulness, but believe that environmental impact data needs to be transparent; and second, a group that is disgruntled about corporate structures, and post information as retaliation.
C. Computing for all: A sustainable infrastructure in a time of need

Daniel Pargman

With unemployment numbers exploding after the Potemkin-Aramco scandal of 2018, the 2020 Hindsight (great oil reserves) Writedown and the global flash crash of 2023, the emergence of a “lost generation” has profoundly shaken all Western countries. These developments have had a profound effect on the ICT sector both in terms of usage patterns and R&D orientation (and budgets) in the 2020s. This paper outlines a broad research agenda for the development of a low-tech, low-cost and ultra low-energy computing infrastructure that can meet the computing needs of the swelling ranks of unemployed consumers, while simultaneously decreasing the energy/CO2 footprint of our computing infrastructure by an estimated 68%.

We draw on previous studies of marginalized groups’ use of ICT in affluent societies (immigrants, the young, the poor, the unemployed) as well as studies of reverse technology transfer from underdeveloped to formerly-affluent societies (the “counter-ICT4D” movement). We conclude by suggesting an offensive strategy for the radical simplification of hardware, software and networking and propose the Freeternet—a “future-proofed” resilient low-cost, low-energy, limited-bandwidth Internet infrastructure.

D. ICT Development in light of looming civilization-level collapse

Debra Richardson, Bill Tomlinson, Birgit Penzenstadler, André van der Hoek, Ankita Raturi

ICT development is changing rapidly as the global economic crisis deepens and the potential for civilization-level collapse is upon us. This paper describes a qualitative study of how civilizational contraction has affected ICT development over the past decade.

We find that: (a) ICT system development has changed dramatically in balance from a focus on functionality first (assuming resources are available) to a focus on key system properties such as sustainability and resilience first followed by the development of functionality in that context, and (b) the methods by which ICT systems are developed have correspondingly changed, to explicitly guide developers in making choices related to sustainability and resilience in usage process modeling, human computer interaction design, and system architecture design. We find serious shortcomings, however, in current processes and methods should the world experience further deepening of the crisis foreboding collapse, and recommend transformational changes to account for these issues.

E. A policy perspective on emission reductions through heating, lighting and electricity quotas

Daniel Pargman, Baki Cakici

In this age of great hardship, there is a great need for 1) protecting the integrity of national borders in the face of mounting immigration pressure from “flipped” climate zones and failed states (c.f. Garrett Hardin’s (1974) Lifeboat ethics) and 2) to strongly incentivize citizens to do their utmost in husbanding energy and other scarce resources. The first challenge has essentially been met through the development of third-gen drone-mounted search & purge technologies (e.g. OctoSurv). While Swedish CO2 emissions have decreased by 56% compared to the 2010 level of 10 tons CO2e/capita, much is left to do to before reaching the 1 ton/capita goal by 2050.

In the face of inertia in citizen compliance with previous emission reduction plans, we propose a radical three-pronged plan for further emission reductions: 1) the introduction of a strict quota system for subsistence-level heating outside of city centers, 2) a general prohibition of lighting in both private and public spaces during non-productive hours and 3) a strict smartgrid-enforced 200 Watt ceiling on electricity usage per household during said hours.

Society Abstracts

F. ECOin

Ankita Raturi, Bill Tomlinson, Donald J. Patterson

The early 2010s saw a rise in the popularity of peer-to-peer digital currencies with the cryptocurrency Bitcoin leading the charge. Bitcoin was instrumental in revolutionizing the nature of e-commerce after it stabilized at around 42 US Dollars to 1 Bitcoin in 2020. This led to Bitcoin outlasting copycat currencies like Coinye (peaked at 69US$ to 1CYe) and Dogecoin (0.01USD to 1DeC).

The ECOin (see Fig. 2) was the first currency to function as a hedge against continued environmental damage. The ECOin stored a person’s environmental value by capturing the externalities of their actions. Leaps in resource tracking and the quantified self movement, have meant that we now know the environmental impact of each person. The better they are at being sustainable the higher the value of their ECOin. This paper is a retrospective on the role of the ECOin in the extreme rise in global sustainable behaviours, its effects on the economy, and rate of collapse.

Fig. 2. ECOin: cryptocurrency against environmental damage

G. Together now — a study on participatory information diffusion and the media industry

Malin Picha Edwardsson, Elina Eriksson
With heavy rainfall and flooding, drought and new records of cold weather, we are living in an era where climate change no longer is a possibility — it is here. In this study we are evaluating the new collaboration portal on ‘Task Force on Mediated Climate Information’, where governmental agencies, the media industry and the public can meet and collaborate in order to make information on mitigation and adaptation plans easier to grasp and diffuse.

The portal is a multimodal, virtual ICT-platform, where different stakeholders can work together in a participatory process to make local solutions from global knowledge. In particular, this study will look at the media industry’s role in transmogrifying complex issues (climate data, regulations, research findings) into understandable presentations for distribution in different ICT channels. The results from this study show that the participatory aspect is of utmost importance for success, as citizens need to engage in this process.

**H. Social aspects of autonomous and electric cars in Germany**

**Wolfgang Lohmann**

The mantra of electric cars to stabilize smart grids and to reduce the CO2 emissions is still in vogue. Repeatedly, ICT is declared as key to logistics optimization. During the last decade the share of electric cars has increased dramatically, as well as the number of autonomous cars in the last five years.

First, we show the increase of life quality during the last decade. The positive effects to health of both, the improved air quality and the decreased noise level, are now statistically evident. The amount of city space regained from removed parking space due to the success of concepts like ‘car from the moving cloud’ is amazing.

Second, we show that due to the optimized logistics, too few electric cars stand idle and, therefore, cannot provide their battery as buffer for the smart grid. We suggest an explanation why the energy for the cars is still coming from coal. More importantly, we argue that the trend to replace truck drivers by autonomous cars will likely continue. In Germany alone the number of truck drivers has been decreased from about 803 000 to 227 000 within a decade. Together with similar effects of ICT in areas such as automated shops this potentially imposes dramatic societal instability.

**I. Green lifestyle lessons: learning from green lead users**

**Daniel Pargman**

With increasing acceptance of the assertion that we live in an age of decreasing returns of increasing societal complexity (Taftter 1988), we urgently need to look for examples that can help us transition to simpler, more sustainable, low-energy “green” models of consumption. This paper summarizes lessons learned from the decade-long research project “Green lifestyles for reduced energy consumption” (Pargman, Eriksson & Katzeff 2026).

More specifically, we discuss 1) the results of early studies of “lead users” (primarily members of eight Transition Town initiatives in three different countries) who voluntarily and proactively chose to simplify their lifestyles (with an emphasis on attitudes, actions, computing habits and everyday energy consumption), 2) the design and development of concepts, prototypes and products that embody lead users’ best-of-breed computing and energy-saving behaviors and 3) the resulting services and products that were developed and marketed by project partner and global retail chain IKEA. We conclude the paper by enumerating the five most promising areas for wide scale energy and computing lifestyle changes.

**J. Participatory simulations: sustainability and the legal rules of corporate governance**

**Bill Tomlinson, M. Six Silberman, Andrew W. Torrance**

The structures of many existing institutions predate sustainability discourse, and may constrain people to engage in activity that impedes progress toward sustainability goals. Corporations are one instance of this class of institutions.

We conducted a series of interactive computational simulations to investigate possible effects of different rules to govern corporate behavior on sustainability outcomes. In particular we examined the potential role of the benefit corporation, a legal entity similar to a for-profit corporation, but with an explicit goal of have a positive societal and/or environmental impact.

In addition, we envisioned and implemented several other models for corporate activity, including the zero-sum corporation (which must document how its growth is offset by the shrinkage of some other corporation) and the anti-growth corporation (which is funded via a subscription model, and uses its resources to cause particular sectors of industry to shrink). These models of corporate behavior present a range of potential outcomes that do not perpetuate “business as usual.”

**K. The Open Revolution: Lessons learned from the era in which technology united humankind**

**Martin Mahaux, Alistair Mavin, with contributions from the Open Interdisciplinary Welfare Research Group — The Wiki-University**

Breakthroughs in communication technologies have always been central to major historical changes. The Open Revolution followed this pattern, but could easily have failed. The power of capitalism could have destroyed the planet and seemed impossible to break down. However, Goliath ignored David: thousands of skilled citizens, spread over the world, who believed in open, participative and sustainable innovation. They influenced millions, empowered them, allowing them to change from dumb consumers into smart prosumers.

People naturally switched from competition to collaboration, enabled by frictionless knowledge sharing infrastructures. This eliminated corruption, private life violations, technology lock-in, lack of interoperability, obsolescence and many other evils. Collaboration made us feel part of a single world, which raised levels of respect for both people and the planet.

We analyse how those ingredients were crucial in finally leading the Open Revolution to destroy and replace the growth-based capitalist economy and representative democracy (see Fig. 3).
WASTE ABSTRACTS

I. EverGreen: An MDD-based approach for generating self-recycling cyber-physical systems

Xavier Franch

The automated support to the produce - use - retire cycle in cyber-physical system has been an object of attention in the last decade. Approaches including those by James Disposal and Henry Waste at 50th ICSE have advocated for the use of MDD techniques but are still subject to limitations, remarkably the inability to automatically retire all the components that conform the system. This paper presents EverGreen, a model-based development (MDD) approach that, starting from a UML6.5 specification, is able to generate adequate software and hardware components that apply the correct technique to every type of component in order to retire them.

In the case of physical components, we use the TotalPrint last generation 3D printer to generate mechanical artefacts. Total print also offers chemical compounds able to destroy those components with minimal environmental impact according to the ISO/IEC 84320 Sustainability Quality Model. The approach has been validated through a multi-case study conducted at the premises of the Dharma Initiative.

M. A general theory for implementing interactive systems on legacy hardware

Bill Tomlinson, Donald J. Patterson, Debra Richardson

The recent convergence of developing nations and collapsing nations has led to profound shifts in resource availability around the globe. For example, the non-renewable resources needed to manufacture post-post-modern ICT systems (e.g., metals such as rhenium, ytterbium, and manganese) are largely located in landfills in developing nations, as a result of decades of e-waste exportation from nations that are now collapsing. Shortages of these resources in collapsing nations, coupled with continuing demand for ICTs to support communication, commerce, coordination and other human activities, has led to a new trend in which modern functionality is adapted to operate on hardware that significantly predates it.

We introduce a general theory for layering new interactions on legacy platforms. This theory accounts for the development of IVOR (Interactive Video Over Radio) and TVSL (Twitter Via Signal Lamp) and presents other transformational changes in technologies that may support human wellbeing in the context of growing resource scarcity.

N. Mother Svea Vigilant: Lessons learned from a nation-wide anti-waste initiative

Baki Cakici, Daniel Pargman

In this paper, we analyze the widely acclaimed “Mother Svea Vigilant” initiative aimed at eliminating wasteful consumption in Sweden. The initiative was funded by the Swedish state between 2021 and 2026 to recognize and classify consumption acts by automatically monitoring commercial transaction logs from all Swedish households and combining them with data submitted by citizens’ smart-ID implants. From a technical perspective, we argue that automatic advisory methods such as scheduled comparisons of recycled mass versus the total mass of purchases in a given time period have created new possibilities of ensuring enthusiastic public commitment to monthly recycling quotas.

We also analyze the success of social aspects of the Mother Svea initiative such as the “See some waste, tell with haste!” program and the community-enhancing “tell (on) your neighbor” campaign. We conclude that Mother Svea and other comparable neo-Benthamite national ICT initiatives this far provide the only scientifically proven methods to stem CO2 emission through the combination of powerful technical and social motivators.

O. Carbon Glass: Reducing your carbon footprint

Kristin Roher, Debra Richardson

Over the past decade, researchers have begun realizing the full potential of image processing and computer vision advances and also the ability of carbon tracking algorithms to accurately report environmental impact. These advances have led to our development of Carbon Glass, an application running on Google Glass that tracks the wearer’s resource usage (see Fig. 4). Carbon Glass performs carbon footprint analysis, and based on observed behaviors, provides users with recommendations for ways to cut resource usage and reduce one’s carbon footprint — all uploaded to a personalized webpage. Together with each recommendation is an explanation and further background as to how the recommended behavior will affect one’s carbon footprint. Further, if an impactful, immediate change in behavior is recommended, Carbon Glass suggests it via the Google Glass voice response. This paper discusses the requirements elicitation for this product, particularly concerning the following features and the design tensions between them: feedback, control, privacy, information access, and avoiding interruptions. The design tension between privacy concerns and the need for information access to make the product functional is discussed in detail.

FOOD ABSTRACTS

P. Food Packets: Network graphs for food distribution

Donald J. Patterson, Bill Tomlinson, Gillian R. Hayes, Melissa Mazmanian, Lynn Dombrowski

Various food distribution mechanisms exhibit different network structures. For example, food systems may use peer-to-peer structures (where individuals engage in many small exchanges across different locations), hub-and-spoke structures (where individuals engage in many small exchanges across different locations), hub-and-spoke structures
Carbon Glass: Reducing your carbon footprint

(where many individuals collect from a single distribution center), and hub-and-tree structures (where there is a centralized distribution point from which one or more distribution trees arise). Each of these structures is appropriate for different kinds of exchange, and would benefit from different kinds of ICT support.

We have developed an ICT application, informed by ethnographic analysis, that supports the process of local food distribution by matching the characteristics of the system to an appropriate network topology. Our research describes how ICT can most effectively support the distribution of ‘food packets’ in a local food system. Given the different stakeholders and their food distribution structures, such an approach creates new understandings of local food systems (Fig. 5). For example, it may expose food distribution bottlenecks and opportunities, and help local governments and communities understand the resiliency of their food system.

![Big Agriculture Tree Network Topology](image)

**Fig. 5. Local Food Production**

**Q. Evolving optimal ICT systems for local food production**

**Bill Tomlinson, Juliet Norton, Kevin Simonson, Debra Richardson**

Over the past half century, ICT systems have gradually become integral to most local and industrial food production processes (e.g., computer-controlled watering systems, chemical sensors for aquaponics, online harvesting reminders). Until now, the systems that supported local food production have relied on the expertise of the teams creating and deploying them to be successful (e.g., a gardener programs the seasonal water schedule of a watering system).

In the last decade, a considerable amount of data has been generated around key performance indicators (KPIs) of agricultural systems (e.g., species specific water requirement per unit of yield). Such data can be used to automate menial tasks and aid in the execution of complex tasks. This paper describes a new model for agricultural ICT support systems that takes KPIs as inputs to evolve ICT-based control systems that maximize utility of the food system.

**R. Enabling sustainable farming through FarmWell**

**Ruzanna Chitchyan**

The need to ensure continuity of secure food production in the face of mounting environmental change and population growth is one of the biggest challenges of our time. This paper demonstrates how the widespread use of FarmWell systems helps to drastically reduce the negative environmental impacts of food production, as well as raise yields and increase the efficiency of input use.

A number of studies of the system use are discussed, showing that the best results are obtained where all (sensors, autonomous learning, and social networking) components of the system are deployed. Yet, we observe that the use of the integrated knowledge base on sustainable farming techniques, location-specific climate-motivated farming transition pathways, and economic return guidance built into the social networking module has the largest return from the FarmWell adaption. Thus, we aim at even closer integration of agricultural, climate, and economics research into future releases of the system.

**S. Sustainable food production in space**

**Juliet Norton, Bill Tomlinson, Marcel Pafal, Donald J. Patterson**

Since the advent of NASA’s first space garden in early 2016, ICT has played an important role of monitoring and informing researchers and the public of the potential to “off-planet” some of our food production. Although they serve as preparation for distant future emergencies or significant population growth scenarios, most of the experimental lunar food production efforts thus far have been of an industrial mindset.

This paper describes the need for “off-planet” food production to originate from a sustainable model and suggests how “lunar” adjustments can be made to existing ICTs that support sustainable food production on Earth in effort to make such technology also applicable to the moon. In particular, this paper addresses issues relating to the moon’s low gravity environment, its different circadian and annual cycles, and the chemical composition of lunar soil.
Several ways to structure the discussion of this collection of fictional abstracts arose while drafting this paper:

- Synthesis of the abstracts in a summarizing vision, for example in a story of the future
  - Utopian story: Mary wakes up, puts on her Google Glass, stocks up online on her fair trade coffee with 2 ECOIns, and walks out into her farmwell-supported garden to harvest breakfast from her aquaponic food forest.
  - Dystopian story: Mary is woken up by the Mother Svea Vigilant alarm after being “told on” by her neighbor but was able to avoid arrest due to collapse of the police telecommunication infrastructure.

- Further classification of the abstracts
  - Further or not so far into the future (assumes many vs. few scientific breakthroughs in the next 15 years)
  - Assumes continued economic growth vs steady-state/shrinking economy
  - Utopian vs dystopian
  - According to the challenges listed in Sec. II

- Solution elements: How exactly does ICT contribute to solving sustainability challenges in the future?
- Success factors: What assumptions are made for the scenarios to become true?
- Analysis of change and continuity via three perspectives: technological, economic, and social.

While all of these possibilities are worth exploring, we limit ourselves to the last three for a brief discussion and leave further consideration to the conference participants.

A. Solution Elements

The elements that ICT needs to provide in order to contribute to solving sustainability challenges in the future can be classified into:

- Monitoring (e.g., of resource requirements, food/plant growth)
- Controlling (e.g., energy consumption, irrigation of crops)
- Information sharing (e.g., on research, climate change, techniques of farming)
- Consulting (e.g., on adjusting individual behavior, optimizing business processes)

This categorization can be used to gather general requirements on these types of systems and provide a framework for more efficient elicitation for domain-specific instances.

B. Success Factors and Assumptions

For the scenarios behind the fictional abstracts to become true by 2029, several assumptions need to be fulfilled for each scenario. For example, many abstracts suggest that individuals are the driving force with ICT supporting them in collaboration. Other abstracts suggest that the priorities are to be changed towards working “bottom up” from local communities and thereby strengthening resilience.

We can see a spectrum of actions aimed at:

- Individual level (optimization of resource consumption, self management)
- Community level, either in local communities (collaborative consumption, carpooling, etc.) or by building up topic-specific communities (open collaboration, open research)
- Global level (social movements, paradigm changes)

Our projection is that parts of the utopian scenarios as well as parts of the dystopian scenarios will manifest themselves on several of these levels.

C. Analysis of Change and Continuity

An important component of the genius behind the writings of Jules Verne and H. G. Wells emerge from their ability to envision futures not only in which submarines take us beneath the sea and we travel through time but also in which men still wear neckties [26], [27], [28]. Similarly, in thinking through these abstracts, one fruitful approach may be considering both what the authors suggest will have changed and what will implicitly have remained the same. This can be done along at least three dimensions: technological, economic, and social.

Several of the abstracts, including those dealing with utilizing outer space, augmented reality, electric cars, and others, present distinct visions of what will become technologically possible within 15 years. Others make interesting implications about the persistence of particular technologies. For example, the need for implementing TVSL (Twitter via signal lamp) suggests that Twitter still exists 15 years from now. Still others, such as Mother Svea Vigilant, could, technologically speaking, potentially be implemented today. Of importance here is considering how these differing technological visions influence what counts as an important technological advancement in today’s research.

A number of economic shifts are also proposed. For example, ECOin implies some (potentially radical) changes to world economies. Similarly, alternative food production and distribution systems imply potential concomitant shifts in economic practices. On the other hand, entities such as IKEA are suggested to continue to play central roles in consumer products. Also, Open (re)Source suggests not only that corporations still exist as powerful and important entities, but it also perpetuates a mode of environmental activism through capitalist consumption.

This point draws attention to broader social implications. The Open Revolution, as well as the “Task Force on Mediated Climate Information,” both suggest some fundamental changes to the organization of society, labor, and information. Mother Svea Vigilant, with its “tell (on) your neighbor” campaign, presents a powerful, if potentially somewhat dark, vision of how societies might adapt to handle problems of waste reduction. The civilization-level collapse occurring implicitly or explicitly in many of the abstracts certainly suggests major societal-level changes. On the other hand, lead “green” users have existed for some time now [29], and locavores have long advocated the value of locally-based food production and distribution systems.
This section is meant neither as praise nor as critique for any particular abstract. Rather, we seek to point out patterns that may be informative with respect to thinking through how these future visions may shape current research.

V. CONCLUSION

“Think globally, act locally!” goes back to 1915, when the town planner and social activist Patrick Geddes wrote that “Local character is thus no mere accidental old-world quaintness, as its mimics think and say. It is attained only in course of adequate grasp and treatment of the whole environment, and in active sympathy with the essential and characteristic life of the place concerned” [30, p. 397]. Yet, it is still unsolved whether and how we will answer the need for changing our lifestyles towards more sustainability and resilience, and thereby avoid superfluous complexity and the potential for collapse.

This paper presented a set of fictional abstracts that could appear at ICT4S in 2029 around the concepts of sustainability, complexity, collapse, and resilience of ICT systems. We hope that by providing this collection together with the set of referenced definitions and background from the various related disciplines we are able to (1) inspire future research, (2) stimulate discussion at the conference, and (3) contribute to building the ICT4S research community.

ACKNOWLEDGMENTS

This material is based in part upon work supported by the DFG (project ENViroSISE, grant number PE2044/1-1) and the National Science Foundation under Grant No. 0644415. We would like to thank Joseph Tainter and Lorenz Hilty for valuable feedback and discussions as well as Reviewer 2 for helpful comments that made the rooting of the paper more explicit.

REFERENCES


6 Sustainability Design and Software: The Karlskrona Manifesto
Sustainability Design and Software: The Karlskrona Manifesto

Christoph Becker
Faculty of Information
University of Toronto
Toronto, ON, Canada
c Christoph.becker@utoronto.ca

Ruzanna Chitchyan
Dept of Computer Science
University of Leicester
Leicester, UK
rc256@leicester.ac.uk

Leticia Duboc
Dept of Computer Science
State Univ Rio de Janeiro
Rio de Janeiro, Brazil
leticia@ime.uerj.br

Steve Easterbrook
Dept of Computer Science
University of Toronto
Toronto, ON, Canada
sme@cs.utoronto.ca

Birgit Penzenstadler
Institute for Software Research
University of California, Irvine
Irvine, California, US
bpzenzens@uci.edu

Norbert Seyff
Dept of Informatics
University of Zurich
Zurich, Switzerland
seyff@ifi.uzh.ch

Colin C. Venters
School of Computing & Engineering
University of Huddersfield
Huddersfield, UK
c.venters@hud.ac.uk

Abstract—Sustainability has emerged as a central question in society and increasingly as a design concern for computing and software. Multiple engineering communities have started to tackle challenges that are related to sustainability concerns. However, persistent misperceptions in practitioners and research communities are mirrored in a lack of a coherent perspective. There is a severe lack of common understanding of the fundamental concepts of sustainability and how they relate to software systems research and practice.

This article describes a cross-disciplinary initiative to create a common ground and develop a focal point of reference for the global community of research and practice in software and sustainability that can be used for effectively communicating key issues, goals, values and principles of sustainability design.

The key result of this effort comes in the form of the Karlskrona manifesto for sustainability design, a vehicle for a much needed conversation about sustainability in and beyond the software community and the fundamental principles underpinning the design choices in software systems. We outline the motivation for developing this manifesto and discuss the genre of the manifesto and the dynamics of its creation. We illustrate the collaborative reflective writing process and present the current edition of the manifesto itself. We assess immediate implications and applications of the articulated principles, compare these to current practice, and suggest future steps.

I. INTRODUCTION

It is clear that society is facing major challenges that require long-term, joined-up thinking. Sustainability has emerged as a central question of this type, and we consider the discipline of Software Engineering (SE) to become a key player for sustainability. However, software practice has a tendency to focus on the immediate effects and tangible benefits of software products and platforms. Correspondingly, SE research has, for the most part, focused on increasing the reliability, efficiency and cost-benefit relation of software products for their owners by analyzing, measuring, standardizing, reinventing, or continuously improving the efficiency and effectiveness of processes, methods, models and techniques to create, verify and validate software systems and keep them operational.

This focus, and the associated lack of long-term thinking, has been diagnosed multiple times from different angles, articulated in multiple ways. Maintenance and evolution have been topics present ever since the very first software engineering conference [1]. Since then, efforts to increase the maintainability of software products themselves and facilitate their evolution have often focused on improving architecture, decreasing lifecycle costs and managing technical debt [2]. Neumann has criticized the lack of long-term thinking and in particular security considerations in SE [3]. Throughout the past two decades, digital information assets have largely replaced their analog counterparts in the information society. The recognition that they often lack long-term survivability, driven by a number of factors including the rapid lifecycles of software technology, has caused some to speak of a “digital dark age” [4].

As a vocal advocate for long-term thinking, Neumann has on many occasions highlighted the fatal effects of short-sighted design decisions, and points out that “there is much to be gained from farsighted thinking that also enables short-term achievements.” [3]. However, to date, considerations that extend beyond immediate software product qualities and user benefits are generally treated as secondary considerations, optional qualities to be addressed in late design stages if the system under design has progressed satisfyingly far to include such concerns. The larger effect of software artefacts on society and its natural environment is not routinely analyzed.

This has manifested in very diverse ways, which include failures to assure desired system qualities, loss of digital records, and excessive costs accruing over the lifecycle of software systems, but extends far beyond that. Other manifestations of the same phenomenon include systems that have negative far-ranging effects on their environment: In the environmental dimension, these may arise as excessive resource
consumption, electronic waste, and high energy usage; in the technical dimension, the effects can surface as a lack of durability, longevity, resilience, or system longevity; in the social dimension, they manifest in privacy breaches, decreased agency of end users, systems biased towards reinforcing stereotypes, favouring specific populations, or exhibiting severe security and privacy flaws.

As a key driver in the continued automation and dematerialization [5] in almost any domain, software has the potential to play a central role in enabling our society’s transition towards a state that meets broadly formulated and inclusive expectations. The design of the complex software systems at the heart of our society hence comes with a special set of responsibilities. Increasing attention is being paid to the broader effects of software on society and the need to embody an understanding of sustainability into the design of software systems, but a common perspective on what this entails is missing.

This article describes a cross-disciplinary initiative that emerged with the aim to create a common ground and develop a focal point of reference for the global community of research and practice in software and sustainability. The process was initiated when a paper for the Third International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy), held at RE’14, Karlskrona, Sweden, proposed that “[a]n open manifesto for forward-thinking sustainable software design, drafted collaboratively in an open and sustainable process, could set a milestone and provide the necessary focal point for joint future efforts” [6]. The main result of our work is the Karlskrona manifesto [7], a document to be used for effectively communicating key issues, goals, values and principles of sustainability design.

The next section sets a basis for the discussion by revisiting the history of ideas in the area of sustainability in software. Section III traces the history of the manifesto as a genre and draws observations from a study of manifests. It summarizes the lessons learned and the principles guiding the collaborative writing process that we have initiated. The current version of the manifesto is reproduced in Section IV. Section V discusses the implications of these principles on SE research and practice and emphasizes open questions.

II. Software Systems and Sustainability

The concept of sustainability is used by many different communities, often in ambiguous ways. When applied to a software system, the term usually refers to a product’s potential for longevity within a given field [8]. There have been discussions in Software Engineering about sustainability-related topics for a long time. These discussions date back to as early as 1968, where software maintenance and evolution where brought up at the NATO Software Engineering conference [1]. Laws of software evolution were defined shortly thereafter [9], [10]. Over the following decades, the software evolution community has accomplished significant advances in the areas of software maintenance, program understanding, reverse engineering, reengineering, mining software repositories, software migration, and software processes. More recent relevant publications for example include [11], [12], [13]. Each of these research areas has brought forward insights on how to improve software engineering practice and how to improve the quality of the systems we build [14]. This is one important dimension of sustainability that we call the technical sustainability dimension.

When applied to human society more broadly, sustainability often refers to some variant of “the ability of the current generation to meet its needs without compromising the needs of future generations” [15].

More abstractly, it expresses the ability of some system to continue at some desired level of operation. But these and other definitions merely raise further questions [16]. Tainter [17] points out we need to ask: (i) Sustain what? (ii) For whom? (iii) How long? (iv) At what cost? It is perhaps much easier to recognize what is not sustainable. Any system that consistently consumes more value (e.g., money, energy, effort) than it produces cannot be sustained indefinitely if its environment, from which resources are drawn, is finite. By all accounts, human society has been in such a state since the 1970’s, consistently drawing on more ecological resources than the planet can produce [18].

In developing a manifesto, the question of how we define sustainability has proved challenging. Our intent has been to develop a broad set of principles that would motivate deeper thinking on sustainability and software, while avoiding terminological disputes. In this, we have struggled to find a balance between abstraction and precision. We collected a number of definitions of sustainability from the literature (see Table I) and conducted a straw poll where each participant could distribute scores. The results favoured fundamental definitions such as “the capacity to endure” [19], but emphasized the importance of more specific expressions and showed considerable support for most of the entries in Table I.

Our approach has therefore been to select as simple a definition as possible (e.g., “the capacity to endure”), and focus instead on a conceptual framework for thinking about sustainability and a set of dimensions by which to approach it. Even this approach has proved difficult. Early feedback on drafts of the manifesto questioned whether it is even appropriate to discuss sustainability in terms of a set of dimensions. Discussion of this feedback revealed some sharply different interpretations of sustainability.

The core issue can be illustrated in terms of a preference over one or other of the diagrams shown in Figure 1. Fig. 1(a) shows a common visualization of sustainability in terms of three separate concerns. The key idea is that human society is only sustainable if it can be sustained in all three dimensions: social, economic and environmental. This view is incorporated into the triple bottom line approach, where companies account not just for financial returns, but also for benefits and impacts in the social and environmental spheres [30].

This approach is rejected as weak sustainability by those who argue that it’s an error to seek to balance the three concerns. According to this view, Figure 1(a) is misleading, as the economy is really only a subsystem of society, which
Table I: Sustainability-related definitions

<table>
<thead>
<tr>
<th>Source</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SustainAbility [19]</td>
<td>Sustainability as ‘capacity to endure’</td>
</tr>
<tr>
<td>Carlowitz, H. C. [20]</td>
<td>“Nachhaltigkeit” (Sustainability) as sustained-yield forestry</td>
</tr>
<tr>
<td>The Oxford Dictionary of English [21]</td>
<td>“Sustainability” as “capacity to endure”</td>
</tr>
<tr>
<td>UN WC on Environ. &amp; Development [22]</td>
<td>Sustainable development as a development that “meets the needs of the present generation without compromising the ability of future generations to meet their own needs”</td>
</tr>
<tr>
<td>The Natural Step [23]</td>
<td>In a sustainable society, nature is not subject to systematically increasing: 1. concentrations of substances extracted from the earth’s crust. 2. concentrations of substances produced by society. 3. degradation by physical means. And, in that society: 4. people are not subject to conditions that systematically undermine their capacity to meet their needs.</td>
</tr>
<tr>
<td>Rees W. and Wackernagel M. [18]</td>
<td>Ecological footprint as the amount of land and water area a human population would hypothetically need to provide the resources required to support itself and to absorb its wastes, given prevailing technology</td>
</tr>
<tr>
<td>Hemberg R. [24]</td>
<td>Five axioms to define sustainability: 1) Any society that continues to use critical resources unsustainably will collapse. 2) Population growth and/or growth in rates of consumption of resources cannot be sustained. 3) The use of renewable resources must proceed at a rate that is less than or equal to the rate of natural replenishment. 4) The use of nonrenewable resources must proceed at a rate that is declining, and the rate of decline must be greater than or equal to the rate of depletion. 5) Substances introduced into the environment from human activities must be minimized and rendered harmless to biosphere functions. Where pollution from extraction and consumption of nonrenewable resources has proceeded at expanding rates for some time and threatens the viability of ecosystems, reduction in the rates of extraction and consumption of those resources may need to occur at a rate greater than the rate of depletion.</td>
</tr>
<tr>
<td>Tainter J. [17]</td>
<td>“(i) Sustainability is an active condition of problem solving, not a passive consequence of consuming less. 2) Complexity is a primary problem-solving tool, including problems of sustainability. 3) Complexity is problem solving as an economic function, and can reach diminishing returns and become ineffective. 4) Complexity in problem solving does its damage subtly, unpredictably, and cumulatively over the long term. Sustainability must therefore be a historical science. 5) Sustainability may require greater consumption of resources rather than less. One must be able to afford sustainability. 6) The members of an institution may resort to resilience as a strategy of continuity only when the option of sustainability is foreclosed. 7) A society or other institution can be destroyed by the cost of sustaining itself. To define sustainability in a specific context, the questions should be (i) Sustain what? (ii) For whom? (iii) How long? (iv) At what cost?”</td>
</tr>
<tr>
<td>Hillard J., Dijon V. and King M. [25]</td>
<td>Sustainability is often thought of as composed of three overlapping, mutually dependent goals: a) to live in a way that is environmentally sustainable, or viable over the very long-term, b) to live in a way that is economically sustainable, maintaining living standards over the long-term, and c) to live in a way that is socially sustainable, now and in the future. The social dimension of sustainability should be understood as both 1) the processes that generate social health and well-being now and in the future, and 2) those social institutions that facilitate environmental and economic sustainability now and for the future.</td>
</tr>
<tr>
<td>Polese M. and Stren R. [26]</td>
<td>Social sustainability as “policies and institutions that have the overall effect of integrating diverse groups and cultural practices in a just and equitable fashion.”</td>
</tr>
<tr>
<td>Harris J. M. and Goodwin N. R. [27]</td>
<td>“A socially sustainable system must achieve fairness in distribution and opportunity, adequate provision of social services, including health and education, gender equity, and political accountability and participation.”</td>
</tr>
<tr>
<td>Hilty L. M. et al. [28]</td>
<td>For evaluating sustainability of ICT systems, three orders of effect need to be considered. ‘First-order’ or ‘primary’ effects: effects of the physical existence of ICT (environmental impacts of the production, use, recycling and disposal of ICT hardware). ‘Second order’ or ‘secondary’ effects: indirect environmental effects of ICT due to its power to change processes (such as production or transport processes), resulting in a modification (decrease or increase) of their environmental impacts. ‘Third order’ or ‘tertiary’ effects: environmental effects of the medium- or long-term adaptation of behaviour (e.g., consumption patterns) or economic structures due to the stable availability of ICT and the services it provides.</td>
</tr>
<tr>
<td>Mahaux M. (unpublished)</td>
<td>“For me, what matters is naively to make the world a better place, for this and all coming generations, and for all populations. That means keeping an environment in which it is great to live in and that can provide the resources to live well. That means people who live in harmony with each other, with their environment, who are free to think and able to do what will make them happy. Beyond fulfilling their basic needs, it’s about realizing the human potential on earth.”</td>
</tr>
</tbody>
</table>

in turn is a subsystem of the environment (Figure 1(b)). To achieve strong sustainability, we have to acknowledge that there are fundamental biophysical limits that constrain the flows of natural resources on planet earth, and no arrangement of society can be considered sustainable unless it lives within these limits [31]. In this view, it is wrong to talk about sustainability in terms of a set of ‘dimensions’, as the concerns are strictly hierarchical.

The conflict between these two views plays out differently in different disciplines. In economics, it rests on the question of substitutability. Many economists assume that natural capital (the stock of natural resources) are infinitely substitutable with human capital (e.g., human ingenuity). If they are, then economic growth need not be constrained by biophysical limits. However, ecological economists dispute this, and argue that there are firm limits to substitutability, which implies there are limits on economic growth [32]. For social issues, the dispute centres on whether all aspects of social sustainability trace even to questions of distributability justice over access to (natural) resources, or whether there are other aspects of social sustainability (e.g., human rights) that arise independently from the question of how we allocate resources.

While we believe these questions are important, we do not believe they offer a useful starting point for software practitioners and researchers struggling with the question of what sustainability means for them. A more pragmatic view is shown in Figure 1(c), where sustainability is depicted as a learning process by which we move towards integrated thinking. Software practitioners tend to treat techno-centric concerns (e.g., software qualities and the economic value they create) separately from socio-centric concerns (how software can make people’s lives better) and eco-centric concerns (protecting the environment). 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
In the past few decades, production and use of information technologies (IT) have had a dramatic effect on society, giving us new tools and new capabilities, but also generating a massive growth in demand for energy and other resources. Software systems, in particular, play a transformative role, as they enable dematerialization [33], drive consumption patterns for products, services, materials, and energy, and facilitate structural changes from consuming material goods towards consuming immaterial services, such as the shift from listening to music online instead to purchasing and discarding hard copies. They also collect, manage and distribute information needed to understand long-running complex phenomena ranging from climate data to personal health records and statistics on global equity and capital. As such, the software industry increasingly represents a central driver for innovation and economic prosperity, but simultaneously increases social inequity, as people without access and technical skills are left behind [34], and causes environmental damage, as consumption of technology grows [35].

The approach we have adopted is to focus on how we understand and take responsibility for the multiple interacting opportunities and impacts of software technology, including first, second and third order effects [36]. First order effects are impacts and opportunities created by the immediate existence of a software system, arising from its design features and flaws. Second order effects are those created by the ongoing use and application of the software, such as how it changes what we do and what we’re capable of. Third order effects are the changes that occur through the aggregated behaviours of very large numbers of people using the technology over the medium to long term (e.g., energy demand, mass surveillance, etc.). These effects play out across many domains. Following Goodland [37] and Penzenstadler and Femmer [38], we identify five sustainability dimensions:\footnote{We observe that these dimensions are interdependent. Cumulative effects from the individual dimension will bleed into the social one; effects from the environmental dimension into the individual, social, and economic, and so on. Yet, these dimensions provide a useful tool for dis-aggregating and analyzing relevant issues.}

- **Environmental**: seeks to improve human welfare by protecting natural capital. The dimension includes ecosystems, raw resources, climate change, food production, water, pollution, waste, etc.
- **Social**: aims at preserving the societal communities (groups of people, organizations) in their solidarity and services. This includes social equity, justice, employment, democracy, etc.
- **Economic**: aims at maintaining assets in terms of capital and added value. This includes wealth creation, prosperity, profitability, capital investment, income, etc.
- **Technical**: refers to longevity of information, systems, and infrastructure and their adequate evolution with changing surrounding conditions. It includes maintenance, innovation, obsolescence, data integrity, etc.
- **Individual**: refers to maintaining individual human capital. This includes mental and physical well-being, education, self-respect, skills, mobility, etc.

An understanding of these short and long term effects of software technology and how they play out over multiple dimensions can then lead to a consideration of leverage points [39]: where are the most effective places to intervene to achieve sustainability? Such interventions might be changes in the way we analyze and design software systems, or they might change in how we seek to apply software solutions to societal problems.

A growing concern among software researchers about these impacts, and a desire to find good leverage points, has inspired a number of workshops at software-related conferences dedicated to focusing on how we integrate and long-term thinking.

![Competing Visualizations of Sustainability](image_url)

---

Figure 1: Competing Visualizations of Sustainability.
cated to software and sustainability. For example, at ICSE’09, a special conference session explored the relationship between Software Engineering and climate change, which led to a series of workshops on software research and climate change (WSRCC), at Oopsia/Ondward! in 2009, at ICSE in 2010, and at ECOOP in 2011. These led to a special issue of IEEE Software [40]. As interest grew and broadened, the community was brought together under the umbrella of the GREENS (Green & Sustainable Software) workshop series at ICSE through a workshop merger, and held in 2012, 2013, and 2014, and another series of workshops on RE for Sustainable Systems (RE4SuSy) at REFSQ in 2012 and at the RE Conference in 2013 and 2014.

Other research communities have followed a similar path [5]. The Human-Computer Interaction community began series of workshops on HCI and sustainability at CHI’2009, which have been held every year since, and the Artificial Intelligence community runs a series of workshops on “Computational Sustainability”, which began with separate workshops at Cornell in 2009 and MIT in 2010, and a conference track at AAAI from 2011 onwards. Meanwhile, the scientific computation community have started a series of workshops on “Sustaining Software for Science: Practice and Experience” (WSSSPE), at SC’13 and SC’14, focussing more specifically on issues of technical sustainability. Finally, a new annual conference series on ICT for Sustainability (ICT4S) launched in 2013. At the same time, other communities with a long-term view on socio-technical systems, such as digital curation and preservation, have attempted to identify what sustainability concerns in software technology mean for them [6].

These various efforts tackle a wide range of different research questions, often with very little overlap. But they all share a sense that the design of software is critically important for sustainability. It is through design that we engage in a process of understanding the world and articulating an alternative conception on how it should be shaped, according to the designer’s intentions. We take the view that all design has an impact on sustainability and all software has an impact on the world. Therefore, it is the responsibility of those who are involved in the creation of software to consider this impact carefully, so that we might make wise decisions for the future.

Multiple software research communities have recognized the need to tackle the issue of sustainability. They are developing conceptual frameworks, techniques, and systems to understand different aspects of the problem [5]. Some seek to encourage reductions in consumption of energy and material goods, or to support changes in purchasing behavior. Others seek to use software capabilities to build smarter (lower impact) infrastructure. However, there is a lack of common understanding of the fundamental concepts of sustainability and how they apply and a need for common ground and consistent terminology. As such, persistent misperceptions occur, as researchers and practitioners disagree over whether we’re even asking the right questions (see, for example, Streger’s essay on “Designing for Resource Man” [41]). We lack a coherent framework with sound theoretical basis that can provide a well-understood trans-disciplinary basis for sustainability design [16], [6].

III. TOWARDS A MANIFESTO

Historically, a common vehicle for catalyzing communities and providing such a focus in comparable situations has been the genre of the manifesto. Communities have relied upon it to articulate their viewpoint, often as an alternative one opposed to a prevailing paradigm. Some manifestos very effectively captured key messages with an appeal beyond the originating community and hence provided a platform for subsequent thoughts and initiatives to develop. Examples in the software world include the GNU

\[\text{https://www.gnu.org/gnu/manifesto.html}\]

and the Agile Manifesto\[\text{http://agilemanifesto.org/}\]. However, these manifestos have little in common in terms of their structure and content, and the manifesto is a delicate genre. What makes a manifesto a successful ‘point of reference’?

As a preparation for RE4SuSy, the first author conducted an informal study to reflect on the nature and history of this genre, based on a review of about two dozen manifestos in the areas of SE, computer science and broader fields of product design, and secondary sources discussing the genre and its practical aspects. The purpose of this study was to understand the nature of the writing process and the possible implications of its product; enable a conscious choice as to whether the creation of a manifesto is a desirable mechanism with a positive impact; identify key elements to address and possible pitfalls to avoid; and derive a set of principles to guide the process. This section reflects on the findings, describes the principles, and outlines the writing process initiated at the workshop.

The origin of the manifesto as a distinct genre can be traced to documents such as Luther’s theses and the Communist manifesto. A fascinating account of the early history of manifestos in politics and art is offered by Puchner [42] who traces the distinct nature, rhetorics and effects of manifestos across the evolution of the genre up to the art and politics manifestos of the 20th century. Fundamentally, the manifesto is a speech act [42]. While originally, it manifested the will of an authority, Luther and Marx morphed this act into one that assumes such authority. As such, the act becomes inherently one in future perfect tense [42]: A successful manifesto will have been effective in capturing a more commonly understood message and articulating it clearly enough to enable others to self-identify with the message. Here, the distinction between declaring ‘manifesto’ in a title line and the rhetorical nature of the manifesto becomes visible. However, we also need to distinguish between the intentionally polarizing nature of earlier manifestos in politics, art, and design, and recent interpretations of manifestos which implicitly assume a much more conversational standpoint, aiming to initiate broader reconsideration rather than aiming to create a revolution.

What is a successful manifesto, then? An ideal manifesto can provide a focal point of reference and catalyze communities by phrasing key questions in accessible language appealing to a broad audience. It enables others to see connections and
synergies and self-identify with the concerns articulated in the manifesto. It contributes to unifying the language about a subject and facilitates visible community building. It also facilitates the action of reaching out to related communities with a clear value proposition and provides arguments for the relevance of the topic, especially encouraging new community members to engage. As such, it can enable a clear communication of the benefits of engaging in the subject.

However, the manifesto has also been called a ‘defunct format’ that ‘belongs to the early twentieth century’4. It arises from the nature of the genre that the creation of a manifesto brings with it the potential for pitfalls and negative consequences. The compact, shortened form of communication often assumed in a manifesto can appear dogmatic, and catchy language designed to be broadly appealing can ultimately hide the real complexity of underlying issues. A polarizing perspective can result in splinter groups rather than a unification, and the questioning of commonly accepted assumptions might alienate rather than unite the audience. Hence, an intended focal point of reference can make others feel excluded rather than invited.

The acknowledgement of these risks was discussed in the early stages when initiating the collaborative process and has led us to articulate a set of principles, meant to guard the emerging group from the above mistakes, avoid groupthink, and foster a sustainable process. These have guided our work:

Principles, not techniques. The manifesto should focus on principles and values of sustainability, not on current techniques, specific models, and suggested approaches.

Scope. The intended scope is broad and inclusive, but clearly delimited. This is inherently difficult to define and achieve, but the principle has repeatedly been brought in when shaping the manifesto to provide a balance and a focus consistent with what the authors are confident to address.

Emerging structure. We believe that the content and the structure of the manifesto needed to emerge from a common set of elements arising from the discussion, initially at the RE4SuSy workshop. This has prompted the process to be very bottom-up and the structure strongly grounded in what emerged from the contributions of the initial group of workshop participants. Facilitation within and beyond the workshop was strongly focused on providing a stable structure, documenting outcomes, and facilitating the discussion.

Participation and transparency. The discussion was initiated within the workshop, and all participants of the workshop were invited to the subsequent process. No conditions are set for entering the discussion process. The initial document was released publicly for comments5 and presented at the RE closing session, and direct discussions with a number of experts in the fields of sustainability have been initiated. Broader engagement with the community includes a discussion panel held at the SPLC’14 conference [44], a workshop proposed for the iConference 20156, a discussion at WSSSPE27, and a special discussion session planned for RE4SuSy’2015.

Conversation over consensus. This acknowledges that while internal consensus is critical, universal consensus is an elusive, and maybe undesirable, goal. The intention for external engagement is to initiate a dialogue rather than aim for full consensus within an extremely broad community.

Minimal and adaptive process. In line with the focus on emergent content and structure, we designed only a minimal process and created the required support structure only as necessary. This eventually included an email list, regular Google hangouts, and a shared folder.

Synchronous collaboration. The elements of the manifesto, at all times, were written and edited in fully synchronous collaboration, first in person during the RE conference, then virtually on Google Drive8.

Iterative evolution. A vision was formulated early on, but no specific milestones or objectives were set and the process was intended to be incremental, iterative, and open-ended.

As preparation for the workshop discussions in parallel to the study of manifestos, we elicited initial responses, gathered a sense of the common ground through a voting on sustainability definitions, and collected initial thoughts and principles of sustainability for a possible manifesto.

At the workshop, all participants supported the collaborative writing of a manifesto and worked through a number of brainstorming sessions to collect starting points for central statements in the categories context, purpose, scope, principles and values, best practices, and prescriptions. After the workshop, the core set of interested collaborators continued to work on the manifesto throughout the conference via a number of intense face-to-face writing sessions, each between two and five hours. As a result, by the end of the third day of the conference, an initial version of the manifesto was released publicly and presented at the Workshop Highlights Session of RE’14. A combination of weekly synchronous collaborative writing sessions and individual contributions and email discussions continued over several months.

IV. THE KARLSKRONA MANIFESTO

4See [43] for a cultural perspective on manifestos in art.


6http://ischools.org/the-iconference/

7http://wssspe.researchcomputing.org.uk/wssspe2

8This paper was developed in similar fashion on overleaf.com.
THE KARLSKRONA MANIFESTO
FOR SUSTAINABILITY DESIGN
Version 0.5, January 2015

Introduction
As software practitioners and researchers, we are part of the group of people who design the software systems that run our world. Our work has made us increasingly aware of the impact of these systems and the responsibility that comes with our role, at a time when information and communication technologies are shaping the future. We struggle to reconcile our concern for planet Earth and society with the work that we do. Through this work we have come to understand that we need to redefine the narrative on sustainability and the role it plays in our profession. What is sustainability, really? We often define it too narrowly. Sustainability is at its heart a systemic concept and has to be understood on a set of dimensions, including social, environmental, economic, individual, and technical. Sustainability is fundamental to our society. The current state of our world is unsustainable in more ways that we often recognize. Technology is part of the dilemma and part of possible responses. We often talk about the immediate impact of technology, but rarely acknowledge its indirect and systemic effects. These effects play out across all dimensions of sustainability over the short, medium and long term. Software in particular plays a central role in sustainability. It can push us towards growing consumption of resources, growing inequality in society, and lack of individual self-worth. But it can also create communities and enable thriving economic growth and resource conservation. As designers of software technology, we are responsible for the long-term consequences of our designs. Design is the process of understanding the world and articulating an alternative conception on how it should be shaped, according to the designer’s intentions. Through design, we cause change and shape our environment. If we don’t take sustainability into account when designing, no matter in which domain and for what purpose, we miss the opportunity to cause positive change.

We recognize that there is a rapidly increasing awareness of the fundamental need and desire for a more sustainable world, and there is a lot of genuine desire and goodwill - but this alone can be ineffective unless we come to understand that...

There is a narrow perception of sustainability that frames it as protecting the environment or being able to maintain a business activity. Whereas as a systemic property, sustainability does not apply simply to the system we are designing, but most importantly to the environmental, economic, individual, technical and social contexts of that system, and the relationships between them.

There is a perception that sustainability is a distinct discipline of research and practice with a few defined connections to software. Whereas sustainability is a pervasive concern that translates into discipline-specific questions in each area it applies.

There is a perception that sustainability is a problem that can be solved, and that our aim is to find the “one thing” that will save the world. Whereas it is a “wicked problem” - a dilemma to respond to intelligently and learn in the process of doing so; a challenge to be addressed, not a problem to be solved.

There is a perception that there is a tradeoff to be made between present needs and future needs, reinforced by a common definition of sustainable development, and hence that sustainability requires sacrifices in the present for the sake of future generations. Whereas it is possible to prosper on this planet while simultaneously improving the prospects for prosperity of future generations.

There is a tendency to focus on the immediate impacts of any new technology, in terms of its functionality and how it is used. Whereas following orders of effects have to be distinguished: Direct, first order effects are the immediate opportunities and effects created by the physical existence of software technology and the processes involved in its design and production. Indirect, second order effects are the opportunities and effects arising from the application and usage of software. Systemic, third order effects, finally, are the effects and opportunities that are caused by large numbers of people using software over time.

There is a tendency to overly discount the future - in fact, the far future is discounted so much that it is considered for free (or worthless). Discount rates mean that long-term impacts matter far less than current costs and benefits. Whereas the consequences of our actions play out over multiple timescales, and the cumulative impacts may be irreversible.

There is a tendency to think that taking small steps towards sustainability is sufficient, appropriate, and acceptable. Whereas incremental approaches can end up reinforcing existing behaviours and lure us into a false sense of security. However, current society is on a path that is so far from sustainability that deeper transformative changes are needed.

There is a tendency to treat sustainability as a desirable quality of the system that should be considered once other priorities have been established. Whereas sustainability is not in competition with a specific set of quality attributes against which it has to be balanced – it is a fundamental precondition for the continued existence of the system and influences many of the goals to be considered in systems design.

There is a desire to identify a distinct completion point to a given project, so that success can be measured at that point, with respect to a pre-ordained set of criteria. Whereas measuring success at one point in time fails to capture the effects that play out over multiple timescales, and so tells us nothing about long-term success. Criteria for success change over time as we experience those impacts.

There is a narrow conception of the roles of system designers, developers, users, owners, and regulators and their responsibilities, and there is a lack of agency of these actors in how they can fulfill these responsibilities. Whereas sustainability imposes a distinct responsibility on each one of us, and that responsibility comes with a right to know the system design and its status, so that each participant is able to influence the outcome of the technology application in both design and use.

There is a tendency to interpret the codes of ethics for software professionals narrowly to refer to avoiding immediate harm to individuals and property. Whereas it is our responsibility to address the potential harm from the 2nd and 3rd-order effects of the systems we design as part of our design process, even if these are not readily quantifiable.

As a result, even though the importance of sustainability is increasingly understood, the majority of software systems are created unsustainably and often decrease sustainability instead of increasing it.

Thus, we propose the following initial set of principles and commitments:

Sustainability is systemic. Sustainability is never an isolated property. Systems thinking has to be the starting point for the transdisciplinary common ground of sustainability.

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.

Sustainability transcends multiple disciplines. Working in sustainability means working with people from across many disciplines, addressing the challenges from multiple perspectives.

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.

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.

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.

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).

It is possible to meet the needs of future generations without sacrificing 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.

Sustainability requires long-term thinking. Consider multiple timescales, including longer-term indicators in assessment and decisions.

Signed,
What implications do these principles and commitments have on Software Engineering? The present section focuses on the implications and questions that the principles advocated in the manifesto raise for SE research and practice.

While some software systems have very explicit goals related to sustainability, for other cases the role of sustainability is more subtle. In practice, the opportunities and risks raised through such interventions have to be understood from multiple perspectives. This requires conceptual frameworks, but also a culture that welcomes, encourages and rewards this understanding and enables these perspectives to be adopted in the professional practice of system analysts, designers and developers.

In this practice, sustainability can not simply be seen as a quality of the systems we design. Crucially, we must distinguish between a (solution-oriented) system quality and a (problem-oriented) concern, i.e. an ‘interest in a system relevant to one or more of its stakeholders’ [45]. Considering only the system under design from a technical and economic perspective, the (technical) sustainability of a system architecture, as defined in [46], is clearly a system quality and can be measured and improved by techniques such as evolution scenario analysis, architecture compliance checks, and tracking of architecture-level code metrics. However, in the overall design of the complex socio-technical system that contains this system architecture, sustainability needs to be treated as a design concern of interest to multiple stakeholders that will drive specific capabilities and qualities in the system [6]. As such, it will interact in different ways with technical features and system qualities. Understanding these interactions and designing the system accordingly is a challenge that current methods, techniques and tools do not fully address, and needs a broader, more holistic perspective than product quality models and architectural metrics can capture.

Consider an imaginary software company called CodeIT. Their next project is developing a community car sharing application that specifically intends to satisfy the needs of a suburban community in a western country such as the US. Kodi, the project manager, is sensitive to the impact of car traffic on the environment and painfully aware of the technical debt carried by the project she just completed. As such, she is determined to make her best effort to design responsibly and effectively this time. Is she also aware of the complex dependencies that will surface in this system, and will she be able to contribute both to the sustainability of this system and to the envisioned positive impact it should have?

Kodi could initiate an investigation into the various concerns for economic, environmental, individual, social, and technical sustainability in this project by asking initial guiding questions similar to those suggested in [47]:

1) Does the system have an explicit sustainability purpose? Can we analyze it in depth using sustainable development scenario techniques [48]?
2) Which direct impact does the system have on its operational environment? Which indirect effects can we identify?

3) Who are the stakeholders for sustainability? Who are the domain sustainability experts, policy makers, and legal representatives?

These questions highlight that Requirements Engineering is a key area where systems level thinking can be applied to identify sustainability concerns, as it translates the domain-dependent goals and concerns into technical requirements that can be realized in the implementation of a software system. Requirements engineers can question user needs, shape people’s expectations and use sustainability concerns and apparent conflicts creatively as drivers for innovation.

Currently, however, Kodi is unlikely to even begin asking these questions. Corporate culture will often not encourage and reward them, and the company’s incentive structure may instead favour short-term thinking based on the economic paradigms that corporations are operating in.

Using the manifesto as a guide, how does a commitment to the principles affect the car sharing scenario?

**Sustainability is systemic.** This raises fundamental questions about the relationship between the proposed software and the problem it is attempting to solve. Often, software engineers fall into the trap of solutionism [49]. Kodi’s questions are based on the assumption that the car sharing app will solve a real problem. However, sometimes, the system the customer wants and the system that should be built are quite different. Choosing appropriate system boundaries and actively critiquing assumptions about these boundaries is important, as any given choice will privilege some stakeholders and their concerns, and marginalize others [50].

**Sustainability has multiple dimensions.** Sustainability transcends multiple disciplines. Sustainability design in this scenario will require cross-disciplinary expertise covering transportation systems, carbon emissions, social network effects, effects on family structures, but most importantly, the interaction between these and additional aspects. The problem of transportation in suburban communities is often a dilemma to be addressed rather than a problem to be solved, and success may be a moving target.

Conceptual models, techniques and tools are needed to communicate, represent, and visualize relationships between software, systems, and particular aspects of sustainability in their social, economic, technical, and natural environment.

**Sustainability applies to both a system and its wider contexts.** Sustainability requires action on multiple levels.

The car sharing application could focus on action on multiple levels. For example, the design of the system could focus on making sharing effortless - an obvious choice. If car booking becomes as efficient and effortless as using one’s own car, though, car traffic could ultimately even increase, countering a key argument originally brought forward in support of the car sharing project. The application design, however, could also attempt to facilitate joint usage to support a reduction in total traffic.

**System visibility is a necessary precondition and enabler**
for sustainability design. Each participant carries a distinct responsibility, and this comes with the need to be informed about the status and structure of the socio-technical system under design and the right to influence the outcomes.

It is possible to meet the needs of future generations without sacrificing the prosperity of the current generation. Sustainability requires long term thinking. The effects of this project will need to be studied over time rather than at the completion of the initial product release. A long-term requirement that may surface in this context is the availability of authentic, reliable records about the system usage and its trends beyond the system life span so that the phenomena associated with such an initiative can be better understood. As a typical case involving the interest of stakeholders that are not commonly involved in such scenarios, this example also illustrates that often, there does not need to be a tradeoff between these future needs and current needs: Well-defined data models and records management principles benefit current stakeholders as well, and ignoring them increases the technical debt of the system design.

If she would ask questions like these, could Kodi convince the stakeholders to see sustainability as the first and foremost goal, to formulate a project vision with sustainability as a precondition rather than an additional requirement? Instead of asking how to build a “system that does car sharing, and is sustainable”, she would ask whether we can build a sustainable system, and if so, whether car sharing is a valuable function within that.

Kodi will need clear guidelines for assessing sustainability on multiple dimensions and multiple timescales. If she is to convince the stakeholders with robust arguments and evidence, she will demand empirically evaluated methods and metrics, measures, and tools for evaluating effects and their interactions. Showcases that demonstrate how sustainability concerns can be integrated and balanced with existing quality attributes and business constraints can support him in understanding and communicating the benefits and opportunities.

Incentive systems are a way to enable Kodi to ask these questions. Can we build reward structures that foster the production of sustainable systems? Is Kodi encouraged to pursue sustainable practice? Maybe, one of the best leverage points would be to redesign the reward structure in the company to encourage those who attempt to apply sustainability design principles - for example, by focus interventions on strong leverage points rather than weak ones through fostering responsible consumption instead of improving energy efficiency, or by giving people information that empowers them to take action themselves (system visibility).

The SE curricula are an opportunity to provide software engineers with the skills to take into account other disciplines, promoting awareness and ensuring they have basic understanding about the implications of software systems on different dimensions of sustainability. Software engineers need systems thinking skills [49] and sufficient understanding of sustainability to ask relevant questions, involve the right stakeholders, and help to perform the required analysis analyses.

However, current standard references and textbooks for SE do not address such topics. For instance, the term ‘sustainability’ features only once in the latest edition of SWEBOK [51], under the section finance.

Finally, the codes of ethics of professional associations such as ACM and IEEE may need to be revisited. For example, the current ACM code of ethics [52] acknowledges that actions with good intentions ‘may lead to harm unexpectedly’, it defines harm as ‘injury or negative consequences, such as undesirable loss of information, loss of property, property damage, or unwanted environmental impacts’. This is not sufficient to cover the potential harmful impacts of technology over multiple timescales and across all sustainability dimensions. The code does not consider second and third order effects adequately in stating: ‘To minimize the possibility of indirectly harming others, computing professionals must minimize malfunctions by following generally accepted standards for system design and testing’.

VI. CONCLUSIONS AND OUTLOOK

Increasing attention is being paid to the broader effects of software on society and the need to embody longer-term thinking, ethical responsibility, and an understanding of sustainability into the design of software systems. However, the software profession lacks a common ground that articulates its role in sustainability design, and a persistent set of misperceptions persist in research, theory, and practice. To truly make progress on understanding the role software plays in the choices we take as designers of the systems at the backbone of our society, we need to understand the nature of sustainability and find a common ground for a conceptual framework.

This article presented a collaborative cross-disciplinary effort to foster the establishment of such common ground. We highlighted a number of perspectives on concerns of sustainability and emphasized the need for a common terminology. We addressed a set of persistent conceptions related to sustainability that we believe are misleading, and proposed a set of counterpoints in the attempt to show how the narrative on sustainability can be rewritten in particular in the context of software systems and the crucial role they play in our society.

The manifesto is meant to undergo future iterations and stay a living, publicly accessible document, and we envision specific extensions articulating the concrete impact that the core principles should have in specific areas. It is put forward as a contribution to a broader conversation and proposes a set of fundamental principles that we see as the seed of a continued conversation about the potential role of our profession both in undermining and in enabling a sustainable future for our planet.

VII. ACKNOWLEDGMENTS

The authors wish to thank all signatories of the manifesto, and in particular Sedef Akınıl Kocak and Coral Calero, for comments and contributions, Emily Maemura for contributions to the study of manifestos, and the reviewers for valuable comments and suggestions. Part of this work was sup-
ported by the Vienna Science and Technology Fund (WWTF) through the project BenchmarkDP (ICT12-046), and by the Deutsche Forschungsgemeinschaft under project EnviroSISE (grant number PE 2044/1-1).

REFERENCES


7 Who is the advocate? Stakeholders for Sustainability
Who Is the Advocate?

Stakeholders for Sustainability

Abstract—While the research community has started working on sustainable software engineering recently, one question that is often asked still remains unanswered: who are the stakeholders? Who are the people who actually have an interest in improving the sustainability of a specific software system or of the discipline of software engineering itself? And who are the devil’s advocates?

Having no explicit stakeholders is a problem as improvement of sustainability is challenging without a driving force. An objective that has no stakeholder is not likely to receive sufficient attention to be realized and will eventually disappear.

In this paper, we present four approaches of identifying stakeholders for sustainability in a given context: top-down by sustainability dimensions (individual, social, environmental, economic, and technical), by instantiation of a generic list, bottom-up by an organigram, and iteratively by an activity model according to the generic sustainability model. We furthermore analyze the feasibility by a small case study for each approach.

As the stakeholders are the key persons determining whether or not any objective is achieved, identifying the stakeholders for sustainability is crucial for successfully implementing sustainability support in a given context.

Index Terms—stakeholders; sustainability; requirements engineering; case study.

I. INTRODUCTION

The Swiss Parliament asked a panel of philosophers, lawyers, geneticists and theologians to establish the meaning of flora’s dignity, which led to a treatise stating that vegetation has an inherent value and that it is immoral to arbitrarily harm plants [18]. This may sound a little far-fetched, but it leads to the question of where responsibility actually starts and where we have to begin looking for stakeholders when it comes to sustainability.

This is one of the challenges we need to answer for a green software industry. For decision-making and incentives to invest in greener software, we need to identify sustainability stakeholders. This will also allow us to investigate the return on investments and economic aspects of green software development.

Problem: Supporting sustainability requires knowing the stakeholders, as the key challenge and success factor for all projects, hence also for sustainable or green software, is the support of stakeholders. Currently, there is a lack of identification methods of stakeholders for sustainability in software engineering.

Contribution: This paper presents four possible approaches to identifying stakeholders in software engineering who have sustainability as one of their objectives. The first approach is top-down by identifying stakeholders according to the five dimensions of sustainability (individual, social, environmental, economic, and technical) [13], [21]. The second approach is the instantiation of a generic list of sustainability stakeholders. The third approach is bottom-up by using the organigram (or organization chart) of the company in the given context. The fourth approach is performed iteratively by deducing the corresponding stakeholders for the activities of an instantiation of a generic sustainability model as described in [21]. We furthermore present a small case study for each approach.

Impact: If we can identify stakeholders for sustainability, it will be easier for software engineers to find incentives to invest in greener software, as they can be coupled to the objectives that these stakeholders already have. By finding the synergies, stakeholders can be convinced that sustainability can be achieved without sacrificing their other objectives. Additionally, together with these stakeholders we can check instantiated sustainability models with regard to completeness and correctness.

Outline: The remainder of the paper gives an overview of the related work (Sec. II), then presents the four approaches for the identification of stakeholders for sustainability (Sec. III), describes the conducted case studies (Sec. IV), discusses the approaches (Sec. V), and concludes with an outlook on future work (Sec. VI).

II. RELATED WORK

The related work for the paper at hand is composed of two areas: stakeholder identification & management in sustainability research and stakeholder identification & management in requirements engineering.

A. Stakeholder Identification in Sustainability Research

In sustainability research, we found work on stakeholder frameworks and stakeholder processes as well as case studies on stakeholder management.

1) Stakeholder Identification: Carroll and Buchholtz [8] investigate the social and political environment of business and explore the role of the corporation in current society. They provide an extensive analysis of stakeholders in business, their

1 i.e. longevity of systems and infrastructure.
stakeholders, and how to consider them during business development and in corporate social responsibility. Their models serve as a basis for the work at hand. Wheeler et al. [29] analyze how corporate social responsibility and sustainable development relate to the creation of business value and how respective stakeholders need to be taken into account. Perrini and Tencati [23] propose a sustainability evaluation and reporting system, which monitors the overall corporate performance according to a stakeholder framework. Hemmati et al. [14] detail how to design multi-stakeholder processes. All these works provide helpful insights on stakeholder identification in other domains but none of them mentions stakeholders for sustainability in the area of software engineering.


B. Stakeholder Identification in Requirements Engineering

The concept of stakeholders is central in requirements engineering and has been investigated by a number of researchers: Glinz and Wieringa [12] introduce the topic and define stakeholders in requirements engineering. Sharp et al. [25] propose an approach to identifying relevant stakeholders for a specific system. Decker et al. [10] explore the use of wikis for stakeholder collaboration. Damian [9] highlights the problem of dealing with stakeholders in globally distributed settings. Woolridge et al. [30] present an outcome-based model for assessing stakeholder risks that identifies deficits between expected and desired stakeholder impact and perceptions. Alexander propagates the “Onion Model” [1] as simple means to structure the stakeholders of a software system. All these works provide insights on stakeholder identification for software systems but none of them regards sustainability.

Mahaux et al. [17] applied Alexander’s Onion Model to a case study that investigated sustainability requirements, but does not detail any further on the stakeholders or the identification process. In summary, various authors have recognized the need for stakeholder identification; however, to all our knowledge there is no approach to systematically identify sustainability stakeholders in IT. Therefore, we take the knowledge from Sec. II-A and apply it to requirements engineering.

III. Identification of Stakeholders for Sustainability

Consider the situation of an analyst working on analyzing or improving the sustainability for a context (i.e. the concrete company or project under analysis). To ensure success of this undertaking, he needs to identify the involved stakeholders. To identify these stakeholders for sustainability there are four potential information sources that imply different, but simple approaches, which we describe in the following sections (see Fig. 1):

1) Analyzing the dimensions to find responsible roles, and matching them top-down to the context.
2) Instantiating generic lists of sustainability stakeholders for the concrete context.
3) Inspecting the context, understanding which concrete roles are involved, and matching them bottom-up to the dimensions.
4) Iteratively analyzing and refining a generic sustainability model.

For applying the method, we expect that there is always one predominant (most suitable) information source that determines which of the approaches shall be used.
A. Approach 1: Top-Down using Sustainability Dimensions

Sustainability can be decomposed into five different dimensions. These dimensions specify different focus points and are connected with different roles. The dimensions are based on [13] and [27] and further specified and extended for software systems in [19].

Source: The sustainability dimensions. Following the definitions given in [19], sustainability can be seen from five dimensions:

- **Individual** sustainability refers to maintaining human capital (e.g., health, education, skills, knowledge, leadership, and access to services).
- **Social** sustainability aims at preserving the societal communities in their solidarity and services.
- **Economic** sustainability aims at maintaining capital and added value.
- **Environmental** sustainability refers to improving human welfare by protecting the natural resources: water, land, air, minerals and ecosystem services.
- **Technical** sustainability refers to longevity of systems and infrastructure and their adequate evolution with changing surrounding conditions.

Task description: The analyst inspects each dimension based on the above definition, and identifies roles that are directly or indirectly connected to this dimension. Afterwards, these roles are related to the concrete project or company in a top-down manner. The dimensions are mapped on the roles within the company. In case no roles can be found, the company needs to analyze if the missing dimension is important to the company and consequently consider creating a dedicated stakeholder role. Thereby, the analyst identifies who is affected by which dimension in this concrete company or for this concrete product. Typical roles that have stakes in social sustainability are managers, in economic sustainability the budget responsible, in technical dimension the administrators, and in environmental sustainability there is either a designated CSR\(^2\) role or a gap.

B. Approach 2: Instantiation of Generic Lists

Within a preliminary series of case studies we performed, we found that the outcomes of Approach 1 tend to repeat stakeholder roles and show potential for deriving a generic list of stakeholders. Thus, we decided to create a reusable, generic list of stakeholders that repeatedly appear in sustainability modeling. A similar approach of having generic lists of stakeholders is also very common in requirements engineering [1], where stakeholders are often identified based on reference models and professional experience. Such reference models in the form of generic check lists help to enable reuse.

Source: A generic stakeholder list. Based on the dimensions explained above and our experience with creating sustainability models, we created a generic list of stakeholder roles, which are independent from a concrete company [20]. The list is by no means complete and not intended to be so; instead,

\(^2\)corporate social responsibility

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Stakeholder Description/Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual</td>
<td>User</td>
</tr>
<tr>
<td>Developer</td>
<td>The developer is heavily involved in creating the system. Aspects like sustainable pace and growth of the developer must be considered.</td>
</tr>
<tr>
<td>Employee represent.</td>
<td>The mental and physical safety of individuals needs to be maintained. Employee representatives watch rights of employees involved.</td>
</tr>
<tr>
<td>Legislation (indiv. rights)</td>
<td>Systems must respect the rights of their users. A legislation representative is a proxy for privacy and data protection laws.</td>
</tr>
<tr>
<td>Social</td>
<td>Legislation (state authority)</td>
</tr>
<tr>
<td>Community represent.</td>
<td>In addition to the state authority, other communities such as the local government (e.g. the mayor) or non-government clubs might be affected by a software system. A complete analysis must take their views into account.</td>
</tr>
<tr>
<td>CRM</td>
<td>The Customer Relationship Manager (CRM) is in charge of establishing long-term relationships with their customers and creating a positive image of the company.</td>
</tr>
<tr>
<td>Economic</td>
<td>CEO</td>
</tr>
<tr>
<td>Project manager</td>
<td>It is very important to have the project manager agree in what ways the project should support sustainable aspects as he decides on prioritization with conflicting interests.</td>
</tr>
<tr>
<td>Finance responsible</td>
<td>As sustainable software engineering often also affects the budget, many financial decisions have to be made to implement a sustainable software engineering model in a company.</td>
</tr>
<tr>
<td>Environm. Legislation (state authority)</td>
<td>Environment protection laws are in place to ensure sustainability goals. These laws must be reflected in the model.</td>
</tr>
<tr>
<td>CSR manager</td>
<td>The CSR manager is often also responsible for environmental aspects.</td>
</tr>
<tr>
<td>Activists/Lobbyists</td>
<td>Nature conservation activists and lobbyists (e.g., WWF, Greenpeace, BUND)</td>
</tr>
<tr>
<td>Technical Admin</td>
<td>The administrator of a software system has a strong motivation for long-running, low-maintenance systems, making this work easier.</td>
</tr>
<tr>
<td>Maintenance/IT</td>
<td>Hardware maintenance is interested in a stable, long-term strategy for installation of hardware items.</td>
</tr>
<tr>
<td>Customer</td>
<td>Users are interested in certain longevity of the systems they are using. This refers to user interface and required soft- and hardware.</td>
</tr>
</tbody>
</table>
we plan to extend it over time and add stakeholders based on new expertise. The current state of the generic list is described in Table I. We would like to offer it as reference checklist for other researchers and persons responsible for sustainability.

Task description: The analyst checks the generic list and instantiates the roles where appropriate. Obviously, some of the roles need to be adapted to the context; yet, they give a reasonably good initial idea.

C. Approach 3: Bottom-Up Analysis of Organigrams

In Approach 1 and 2, we analyzed dimensions, found generic roles and mapped them to concrete roles in the company or product in a top-down manner. In contrast, one could also take the company’s or project’s role model and identify which of these roles are related to the sustainability dimensions. We consider this a bottom-up approach.

Source: A company’s or project’s organizational diagram. Many companies create structured representations of the company’s or projects role model—e.g. as an organigram—in order to visualize the involved institutions and individuals. Even though we will furthermore only consider organigrams, this approach also works with other representations of the role model, e.g. lists of roles in spread sheets.

Task description: The analyst identifies the existing roles in the present context—for example, through an organigram. He then inspects each role and maps it on the dimensions of sustainability.

D. Approach 4: Iterative Analysis of the Sustainability Model

In addition to the bottom-up and top-down approaches from Approach 1-3, another approach is iterative based on a sustainability model that was introduced in [21] and makes the goal “sustainability” tangible in a concrete context. In [21] and [20], we furthermore explain how to derive such a concrete sustainability model from a generic reference model. Moreover, after having an initial sustainability model, the analyst can also use it to identify sustainability stakeholders. These new stakeholders will lead to new elements in the sustainability model, which might lead to new stakeholders and so on.

Source: An instantiated sustainability model as defined in [21] (and depicted exemplarily within Fig. 5). The sustainability dimensions can be derived into a generic sustainability model in order to make sustainability explicit [21]. For each context, the generic sustainability model is instantiated into a product- or company-specific model that describes the values, activities and indicators that specify the exact definition of sustainability for this context.

In short, the model comprises the generic sustainability model (M1 level), the respective meta model behind it (M2), and the instances (M0) derived for specific processes and systems. The generic sustainability model adheres to the following metamodel: Dimensions are represented by a set of values. Values are approximated by assessable indicators, influenced by activities and affected by regulations.

The generic model (M1) is a library that can be structured in three levels: the top level contains the five dimensions; the middle level consists of (currently) 51 values, 5 generic indicators, and 6 regulations; and the lower level is formed by 38 activities. For example, for the dimension social sustainability, the spirit of the community is an important value that can be decomposed in different values such as trust or education. The education value is regulated, amongst others, by human rights. This value can only be assessed roughly and individually by indicators, where one indicator contributing to that assessment is the level of graduation of a person. Education is fostered by different activities, such as knowledge management, education programs or mentoring [22] (see excerpts from model instances in Fig. 5). A process or requirements engineer instantiates this generic model into a context-specific sustainability model by refining the rather abstract activities into concrete actions and defining specific indicators to measure the success in the concrete domain.

For the work at hand, the generic model is one input for the identification of stakeholders and instantiating the model for a specific context is a means to structure the objectives and interests of the identified stakeholders.

Task description: Assuming the analyst has an existing initial sustainability model at hand, he can iteratively analyze the model for lacking stakeholders. By going through the model items, such as values, activities and indicators, he can check a set of questions: Who is actively involved in this item? Who is (passively) affected? Who is interested? When a new stakeholder is identified, the analyst can interview this new stakeholder and extend the model based on this new information. This will lead to a more detailed sustainability model, which will raise new questions of stakeholders and so forth.

IV. Case Studies

To understand the benefits and limitations of the approaches, we conducted a small case study for each of the four approaches.

A. Top-Down Approach: Munich Software Company

For the top-down approach, one of our master’s students (Susanne Klein) created a sustainability model for a Munich-based software company with about 100 employees. To understand the company’s needs and habits, the student conducted a series of interviews. To ensure that all domains of the company are covered, she analyzed which stakeholders could be available for each dimension (see Fig. 2). After she identified various different roles, she asked for feedback from the organization, which led to addition of four more roles and rejection of one role (sales). These decisions were based on the company specifics.

These stakeholders were the basis for a set of interviews and discussions, in which it turned out that the role of a person does not necessarily reflect the subjective perception of importance of the sustainability dimensions. For example, it turned out that for many developers the dimension of technical
sustainability was less important than social and individual aspects. Also remarkable is the role of the innovation consultant, who was supporting the company in long-term development, including the environmental dimension.

B. Instantiation Approach: Car Sharing Platform

In a previous study we instantiated the sustainability model for understanding its applicability to the car sharing platform [21]. For this study we instantiated the generic stakeholder list from Section III-B for the DriveNow car sharing program, which is a mobility service offered by German car manufacturer BMW. The result is displayed in Fig. 3.

The instantiation is straightforward. Roles like CEO and CRM can be directly mapped to the company, as nearly all companies have representatives with this role. Some generic roles have several instances. For example, we have two roles for the generic role maintenance: software maintenance for the implemented solutions, as well as car maintenance that is responsible that all cars are checked and repaired regularly. Also interesting is that the mayor of Munich is represented as a stakeholder, as the German Car Sharing Association quotes the mayor to be a strong supporter of car sharing platforms [16]. This is a good example for making the important stakeholders (here: politics) explicit.

C. Bottom-up Approach: Project Organigram

In contrast to going from dimensions to roles, one can also take existing organigrams of the company or project and understand which roles can be mapped to which dimensions.

The organigram in Fig. 4 is taken from a project management course at TUM and depicts a typical medium to large-sized software project structure (here named Code & Talk). Some roles are straightforward to map: Architecture, quality assurance (QA) and release management have a particular interest in creating a sustainable technical architecture, management tasks are mostly dedicated to keep an overview over finance, and all involved persons have interest in individual sustainability. For social sustainability, the only role that can roughly be connected is the project lead. More alarming, no role could be identified for environmental sustainability.

This highlights several issues in this project: Having the
project lead as the single role being involved in creating a professional team atmosphere is a problem, that is often targeted by creating a dedicated role for team management. The same holds for individual sustainability, as there is no single person responsible. However, this might be taken care of outside of the project. Lastly, there is a lack of responsibility for environmental sustainability.

The analysis gives hints to where too many or too few stakeholders are involved with certain dimensions. This must lead to follow-up questions regarding the goals of this project and company. If project or company goals are under-represented, the company should consider creating dedicated roles for these stakeholders.

D. Sustainability Model Iteration Approach: RE Conf 2013

The fourth option to identify stakeholders for sustainability is to start with an instantiation of the generic sustainability model that includes actions and activities developed to support sustainability in its different dimensions throughout a given context [21]. Such an instance has been elaborated for the upcoming 21st RE conference (RE’13). Given the envisioned activities, they can be assigned to stakeholders that are identified as capable for taking care of the actions. The sustainability model, as well as the identification and assignment of stakeholders for the activities at the 21st RE conference are illustrated in Fig. 5.

For the RE’13 we can derive various stakeholders from the sustainability model. One can see that there is a general stakeholder, the sustainability chair, who takes care of the overall coordination of the sustainability activities; yet, it is important to understand that there are responsible stakeholders for other individual tasks. For example, triggering more interaction as part of the individual sustainability measure for development and growth, i.e., how people can make the most of the conference is a task for the interaction chair.

Naturally, there are a considerable number of assignments for local chairs, general chairs, and other roles, but it is important to note that also the attendees have a crucial responsibility for achieving the objective of sustainability at the conference. Without their support, the undertaking will simply not lead to the desired results.

V. DISCUSSION

This section discusses the advantages and drawbacks of the method as well as of the individual approaches used for the stakeholder identification.

A. Comparison of Approaches

Each of the approaches for identifying stakeholders has its advantages and disadvantages. Yet, an industrial evaluation with more formal criteria is subject of future work. We performed the case studies ourselves, which leads to a subjective, informal comparison of the methods. The most complete set of stakeholders will be found by using their combination.

**Top-Down** identification allows for a rather general reflection on the sustainability dimensions. However, it may be less efficient when compared with instantiation of generic lists, as there is no further input than the definition of the sustainability dimensions. Consequently, the success relies completely on the creativity and cleverness of the person performing the identification.

**Instantiation** of generic lists is probably the most efficient approach (which is probably the reason why these generic lists are mostly used for stakeholder identification in professional RE [25]). On the downside, there is no thinking “outside of the box” involved even though creativity is probably important in order to involve all relevant roles.

**Bottom-up** analysis is very practical and down-to-earth, i.e., with the interviews and lots of input this approach actively includes the employees. At the same time, it is the most time-consuming approach.

**Iteratively** analysis the sustainability model for the identification might be especially beneficial as a closure at the end of the analysis, but it is not as straightforward as the other approaches.

B. Benefits and Limitations of the Stakeholder Analysis

For assessing how effective and how efficient the approach is, we can only refer to our experiences from the first case studies. For each of the case studies the approach identified more stakeholders for sustainability than we had considered originally. This may indicate that the approach help to discover stakeholders that might be missed otherwise.

Consequently, when all stakeholders are involved, who actually do have an interest in sustainability, it is possible to sketch a more complete picture of the subject under analysis through decreasing the risk of forgetting any sustainability issues that should be considered. This is the case for stakeholder identification for sustainability, just as it is during any requirements elicitation activity where stakeholders have not been identified correctly [1].

However, this paper presents only a small set of case studies that were performed so far and, currently, there is no tool support and only the guidance provided in this paper. For further evaluation, we will apply all four methods on one set of case studies. Furthermore, for industrial application, we need to extend that guidance so that industrial collaborators can perform the approach by themselves.

VI. CONCLUSION

This paper proposes method consisting of four approaches to identify stakeholders for sustainability in different contexts. We offer four approaches for this identification: top-down from the sustainability dimensions, instantiated from a generic sustainability stakeholder list, bottom-up from the organigram of a company, or iteratively in parallel with the development of an instance of a sustainability model.

**Results:** We present case studies to illustrate the different approaches for our method. Two of them are performed in collaboration with industry—i.e., a large automotive company and a medium-sized software development company. The other two stem from an academic context, one of them being...
Fig. 5. Sustainability model and stakeholders assigned for the activities to support sustainability at the 21st requirements engineering conference
the conference management of the international requirements engineering conference, and the other one a hypothetical case study on IT project management that is used throughout our IT processes and management course.

The case studies revealed various stakeholders for sustainability that are already present, but not yet seen in this context. They have to be taken into active consideration by an analyst for sustainability. The studies also revealed missing responsibilities that may have to be filled by new roles or new responsibility assignments. For example, if a company decides they want to improve their environmental sustainability, a stakeholder needs to be identified or a specific role created, because otherwise the objective of improving environmental sustainability will not be pursued in an effective way.

**Future Work:** The next step after the presented validation, which was performed by the developers of the approach, is an evaluation performed by external subjects in an industrial context. Furthermore, we envision tool support that guides the identification process using the different methods as options.

**Advocate for Sustainability:** After all, who is the advocate for sustainability? We have identified a number of stakeholders for the dimensions of sustainability in different context. Roles that reoccurred across case studies are captured in the generic list in Tab. I, which serves as a first reference checklist for further research and practice. We are positive that successfully identifying the stakeholders for sustainability will help ensure that this objective receives the deserved attention.

**ACKNOWLEDGMENT**

The authors would like to thank Manfred Broy for insisting on the question about stakeholders until we finally gave it the deserved attention. Furthermore, we would like to thank Susanne Klein for providing us with necessary background on jambit so we could perform that specific case study.

**REFERENCES**


8 Towards Incorporating Sustainability while Taking SPM Decisions
Towards Incorporating Sustainability while Taking Software Product Management Decisions

Birgit Penzenstadler\textsuperscript{1}, Mahvish Khurum\textsuperscript{2}, and Kai Petersen\textsuperscript{2}

\textsuperscript{1} Technische Universität München, Germany
penzenst@in.tum.de
\textsuperscript{2} Blekinge Institute of Technology, Sweden
{Mahvish.Khurum|Kai.Petersen}@bth.se

Abstract. Software product managers are missing guidelines on how to incorporate different dimensions of sustainability in software product management and requirements selection decision-making. This is a challenge because considering sustainability perspective while selecting requirements has become a major objective for software product development companies; however, it is unclear how to support it during complex product management decision-making. In this paper, we identify the value aspects related to sustainability for software requirements selection. An exemplary dialogue between a consultant and a product manager illustrates how the proposed approach can be used while taking product management and requirements selection decisions. Our contribution provides software product managers with guidance on how to incorporate value aspects related to sustainability while taking software product management and requirements selection decisions.

Keywords: Sustainability, value-based software engineering, decision-making, software product management

1 Introduction

In today’s world, software has become the main competitive advantage, enabling faster and cheaper innovation as well as product differentiation, and at the same time hardware is becoming standardized [10,35]. Simultaneously, the size and complexity of software in products are increasing, and so is the impact of software development decisions on the overall product offering [15]. That is, any decision taken regarding software, e.g. what features to realize, what quality to offer, or what technology to choose, will impact the entire product’s life cycle and value, not to mention that it limits future possibilities and direction of the product and business (economic sustainability) [1,19]. Along with this, due to increased awareness about environmental, social and human sustainability, a challenging question is how a company can build innovative products that not only meet the needs of its customers, but are also built in a socially responsible and sustainable way? While it is required to build special software for the
customers to measure, monitor and act on various sustainability indices, it is equally important that the products are developed and managed in an adequate way with an in-depth understanding of the environment they are applied in. This situation gives rise to many decision-making challenges for industry practitioners, for example, which value aspects with respect to sustainability need to be considered while taking software management decisions? How does the realization of one functional or quality feature influence the sustainability value of the product offering, where short-term potential sales and revenues are almost always premiered over sustainability aspects? Answering these questions can help to innovate and develop products that do not only deliver value to the customers, but also enable development of products keeping in view the sustainability perspective. This perspective spans all levels of decision making: on the project, the product, and the portfolio level. Our research question is: *How can we incorporate sustainability as a primary objective with the conventional goals in software product management decisions?*

Value-based software engineering (VBSE) can help answering these questions as it emphasizes that every decision and/or feature of a product does not have an equal value like in a value-neutral setting [2]. This requires making decisions that are better for overall value creation, according to Kontio et al. [21] and Rönkkö et al. [29], and balancing short-term and long-term value creation.

**Contribution** The primary contribution of this paper is a list of value aspects that need to be considered from the perspective of sustainability while taking product management and development decisions. The Software Value Map is used as the basis for identification of these value aspects. The Software Value Map [20] provides a consolidated view on value aspects relevant for taking software product management and development decisions based on the Balanced Score Card approach. In addition to its application for sustainability concerns, a set of value aspects not yet covered by the Software Value Map has also been included, where each aspect is described and given a rationale. The identified value aspects can be used as criteria for taking requirements selection decisions. The application of the approach is illustrated in a fictitious dialogue between a product manager and a consultant.

## 2 Foundations and Related Work

The following sub-sections give a brief introduction to the concepts of sustainability and the Software Value Map as well as an overview of related work.

**What is Sustainability?** The four main dimensions of sustainability that we consider important are human, social, economic, and environmental, see Goodland [9]. The three latter ones are the dimensions known from the most cited definition of sustainable development by Brundtland et al. [4]: “…meets the needs of the present without compromising the ability of future generations to satisfy their own needs.” The first dimension, human, is not present in the public discussion, but we argue that it should be included because it is the basis for the others.
**Human sustainability:** Human sustainability refers to the maintenance of the private good of individual human capital. The health, education, skills, knowledge, leadership and access to services constitute human capital. [9]

**Social sustainability:** Social sustainability means maintaining social capital and preserving the societal communities in their solidarity. Social capital is investments and services that create the basic framework for society. [9]

**Economic sustainability:** Economic capital should be maintained. The definition of income as the amount one can consume during a period and still be as well off at the end of the period can define economic sustainability, as it devolves on consuming value-added (interest), rather than capital. [9]

**Environmental sustainability:** Although environmental sustainability is needed by humans, it itself seeks to improve human welfare by protecting natural resources. These are water, land, air, minerals and ecosystem services; hence much is converted to manufactured or economic capital. Environment includes the sources of raw materials used for human needs, and ensuring that sink capacities recycling human wastes are not exceeded. [9]

Our analysis of how to incorporate sustainability into software product management decisions is based on these definitions as our understanding of sustainability. The foundation we use for guidance in taking software product management decisions is the Software Value Map, described in the following section.

**The Software Value Map.** The Software Value Map [20] provides a consolidated view of the software value concept utilizing four major perspectives: the financial, the customer, the internal business process, and the innovation and learning. The value aspects and value components contained in the map are collected through extensive review of economics, management and value-based software engineering literature.

The value map offers a unified view of value, which can be used by professionals to develop a common understanding of value, as well as acting as decision support to assure no value perspective is unintentionally overlooked when taking product management decisions. For example, during requirements selection in addition to short term increases in customer value and company revenue, a company’s and product’s long-term sustainability view can also be considered. While evaluating the effects of a requirement on the maintainability value of the product’s architecture, human capital value of the company and innovation value would enable a comprehensive (long-term) impact analysis of a certain decision. Thus, by having a value focus, the overall trade-off between positive and negative impact on the present product offering can be estimated. This is central from many perspectives. For example, from a business perspective, the selection and realization of a feature might be good idea, but simultaneously the long-term effects pertaining to, e.g., sustainability of system architecture, might be very negative.

The taxonomy used to categorize the perspectives for measuring value was inspired by the balanced scorecard (BSC) approach, see Kaplan et al. [16,17]. BSC can be defined as a set of measures that gives managers a fast but comprehensive view of the business using four main perspectives, namely the financial,
customer, internal business process, and innovation and learning [16,17], each described below.

The financial perspective contains aspects that address the company’s implementation and execution of its strategy which are contributing to the bottom-line improvement of the company. It represents the long-term strategic objectives of the organization and thus incorporates the tangible outcomes of the strategy in traditional financial terms [32,29]. Some of the most common financial measures that are incorporated in the financial perspective are earned value analysis and profit margins.

The customer perspective defines the value proposition that the company will apply to satisfy customers and thus generate more sales to the most desired (i.e. the most profitable) customer groups, see, e.g., Steven [32]. Measures that are selected for the customer perspective should measure both the value that is delivered to the customer (value proposition) with respect to the perceived value, which may involve time, quality, performance and service, and cost, and the outcomes that come as a result of this value proposition (e.g., customer satisfaction and market share).

The internal process perspective is concerned with the processes that create and deliver the customer value proposition. It focuses on all the activities and key processes required in order for the company to excel at providing the value expected by the customers both productively and efficiently. Quality, cycle time, productivity and cost are some aspects where performance value can be measured [16].

The innovation and learning perspective is the basis of any strategy and focuses on the intangible assets of an organization, mainly on the internal skills and capabilities. The innovation and learning perspective is the intellectual capital categorized as human capital, structural capital, and the organization capital of a company [16,23].

Related Work. The ISPMA provides a first body of knowledge, which does not explicitly consider sustainability yet [15]. Penzenstadler et al. [27] conducted a systematic literature review on sustainability in software engineering. The review revealed no work specifically related to sustainability in the context of software product management decision making. Cabot et al. [7] performed a case study for sustainability as goal for the ICSE organization with $i^*$ goal models to support decision making for future conference chairs, but don’t discuss decision support for potential measures and do not provide methodical guidance or decision support. Naumann et al. [26] investigate how web pages can be developed with little environmental impact, i.e., energy-efficiently, but do not discuss implications on product management. Mahaux et al. [24] performed a case study on a business information system for an event management agency that advertises environment-friendly events but do not address the decision making challenge. While all of these works refer to sustainability goals, none of them discusses values related to sustainability and how to consider them while taking product management and requirements selection decisions. Moreover, within research on value-based software engineering, to the best of our knowledge, no
work yet explicitly discusses sustainability as one of the major consideration in software product management decision-making.

3 Identification of Value Aspects for Sustainability

The software value map can be used as a basis for identification of value aspects to be considered from different sustainability perspectives, while taking product management and requirements selection decisions. We discuss four sustainability dimensions and value aspects relevant to each of the dimensions are given with the rationale for their relevance to the discussed sustainability dimension along with the references for further readings. Tables 1-4 list the values (column Value Name) related to the sustainability dimensions, identify the balanced score card perspective they belong to (column Perspective), describe the value itself (column Value Description), why it matters for sustainability (column Rationale), and what can be done in order to improve it with references to further reading (column Actions & Further Reading). Please note that the list of identified value aspects is not complete rather it is the first attempt towards theoretical foundations for incorporating sustainability perspective while taking requirements selection decisions.

**Human Sustainability.** Software is developed and managed by people. Therefore, it is fundamental to consider value aspects related to human capital while taking product management decisions. The human capital value is described in Table 1. Although value aspects related to human sustainability are not necessarily related to all product management decisions, they actually influence the effects of other value aspects related to sustainability. For example, satisfied developers will be more focused to build high quality products with efficient use of resources.

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Perspective</th>
<th>Value Description</th>
<th>Rationale</th>
<th>Actions &amp; Further Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human capital value</td>
<td>Innovation and learning</td>
<td>Human capital value refers to the stock of skills and knowledge embodied in the ability to perform labor so as to produce economic value. It is the value of skills and knowledge gained by a worker through experience.</td>
<td>For human sustainability, human capital value should be increased by enhancing the skills and knowledge of the developers since they are the work force that develops the system.</td>
<td>Offering training and continued education improves skills and knowledge of an employee. Fitz-enz [8] provides measures for the economic value of employee performance.</td>
</tr>
</tbody>
</table>

**Social Sustainability.** For supporting social sustainability, a software development company may want to consider customer capital value while taking product management decisions. Furthermore, network externalities play a role in binding the customer. In addition to the values in Table 2 that are identified from the Software Value Map, there are other values that can be discussed in this context. The reason for not listing them is that they are related less directly to the software product under development, rather to the surrounding environment,
Table 2. Value aspects for social sustainability

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Perspective</th>
<th>Value Description</th>
<th>Rationale</th>
<th>Actions &amp; Further Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer capital value</td>
<td>Internal Business perspective</td>
<td>Value of relationships that a firm builds with its customers, and which is reflected in their loyalty to the firm and/or its products. It is not reflected in a balance sheet in monetary terms.</td>
<td>Loyalty of customers is a stable basis for continuous bonding and customer retention [14].</td>
<td>Hennig-Thurau et al. [14] suggests improving bonding with customer with the use of motivation theory. Storbacka et al. [33] links service quality with customer satisfaction and profitability.</td>
</tr>
<tr>
<td>Network externalities</td>
<td>Customer perspective</td>
<td>The amount of other users of the software product that are relevant to the focal user, e.g., who might be motivated to use a service due to incentives for the user</td>
<td>If there are incentives for a user to motivate other people to use a service, the user might keep using it for two reasons: the incentive (e.g. lower costs), and the network of users that share the service [18].</td>
<td>Katz et al. [18] provide an analysis of the options to improve availability of complementary goods and services.</td>
</tr>
</tbody>
</table>

...for example, values related to labor practices, human rights, society, and ethical behavior (see Silvius and Schipper [31]). Activities for good labor practices are to ensure employment, to work on labor-management relations, to provide training and education as well as organizational learning, and to offer diversity and equal opportunity. Human rights support is, e.g., to prevent discrimination. For society, potential measures are to improve community support, to perform adequate market communication, and to guard customer privacy [31]. Ethical behavior includes checking investment and procurement practices [31].

**Economic Sustainability** Within a software development context, value aspects related to economic sustainability need to be considered while taking product management decisions. The values detailed in Table 3 are maintainability value, innovation value, differential value, and physical value w.r.t cost. Economic sustainability is the aspect that is already most present in today’s software business. One sub aspect of this is software sustainability, a term used interchangeably with software maintenance, and an innovation infrastructure are fundamental inputs to continuously maintain and evolve software products such that they sustain economically throughout their entire planned lifecycle. Moreover, competitive advantage has to be maintained to ensure economic sustainability of the company.

**Environmental Sustainability** Environmental sustainability may be improved by improving the market requirements value, the physical value w.r.t. cost, the sustainability value of technology, and the product’s intrinsic value. Each of the values is detailed in Table 4. Selling software as services enables a higher rate of innovation and also reduces the number of expensive hardware upgrades that needs to be done. This, in turn, means an increased environmental sustainability. While it can go hand in hand with economic efficiency, this can also have cost-increasing effects, e.g., through additional activities.

**Interrelationships** Interrelationships in the value aspects identified from different sustainability dimensions are possible. The interrelationships can exist as the following effects:
Table 3. Value aspects for economic sustainability

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Perspective</th>
<th>Value Description</th>
<th>Rationale</th>
<th>Actions &amp; Further Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability value</td>
<td>Internal business perspective</td>
<td>The capability of the software product to be modified. Modifications include</td>
<td>Maintainability of a software product is a foundation for sustainability [30] in a</td>
<td>An approach for achieving software sustainability and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>improvements or adaptation of the software to changes in environment and in</td>
<td>broader understanding, as evolution balances the factors to be accounted for when aiming at</td>
<td>how to measure it is given by Seacord [30].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>requirements and specifications.</td>
<td>sustainability.</td>
<td></td>
</tr>
<tr>
<td>Innovation value</td>
<td>Innovation and learning</td>
<td>The practical value of subject technology that is materialized in market (as a</td>
<td>According to Hansen et al. [13], sustainability is a key driver of innovation. If they go</td>
<td>Hansen et al.’s framework [13] allows also for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>product or service) or in business process (as process innovation)</td>
<td>hand in hand, innovation has to be supported.</td>
<td>conclusions with respect to the market.</td>
</tr>
<tr>
<td>Differential value</td>
<td>Internal business perspective</td>
<td>Differentiation is the process of distinguishing the differences of a product or</td>
<td>In order to have sustainable competitive advantage, it is fundamental to strive product</td>
<td>Hall [11] gives a framework for linking capabili-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>offering from others, to make it more attractive to a particular target market.</td>
<td>features/capabilities that enable economies of development and/or lower profit margins</td>
<td>ties to sustainable competitive advantage.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This involves differentiating it from competitors' products as well as one's own</td>
<td>(see Lado et al. [22]).</td>
<td></td>
</tr>
<tr>
<td>Physical value w.r.t. cost</td>
<td>Internal business perspective</td>
<td>A product being developed and marketed with lower development cost will have</td>
<td>For economic sustainability, it is fundamental to keep the development costs as low as</td>
<td>Hyggeth [6] proposes a procedure for sustainability-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>higher Physical value w.r.t. to cost. [20]</td>
<td>possible [6].</td>
<td>driven design optimization illustrated with a case</td>
</tr>
</tbody>
</table>

- A positive impact on one value aspect might have positive impact on one or more sustainability dimension
- A negative impact on one value aspect might have negative impact on one or more sustainability dimension
- A positive impact on one value aspect might have a negative impact on one or more sustainability dimension and vice versa

For example, if quality features, not even demanded by the customers, are provided; the intrinsic value of the product might be very high, however, this will negatively impact the environmental sustainability perspective as Physical value w.r.t. cost would decrease due to extra features produced (a waste). On the other hand, if generic products (which are demanded by majority of the customers) are developed and sold; Market requirements value would be high and Physical value w.r.t. cost would be high which doubles the positive impact on the environmental sustainability. Moreover, by developing maintainable products, while Maintainability value is increased which positively impacts the economic sustainability; however, this could have a negative impact on Human capital value because the developers feel that by maintaining the existing code no new skills are being learnt. Consequently, human sustainability is negatively impacted. Resolving trade-offs between conflicting dimensions (often between economic and environmental, as environmentally sustainable involves adequate supplies) can only be solved by goal prioritisation—the economic side or the environment.
Table 4. Value aspects for environmental sustainability

<table>
<thead>
<tr>
<th>Value Name</th>
<th>Perspective Value Description</th>
<th>Rationale</th>
<th>Actions &amp; Further Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Market requirements value</td>
<td>Internal business perspective Represents the production value with respect to a given market requirement (Time &amp; effort to implement a feature vs requirement's market demand &amp; value)</td>
<td>Producing generic products can save resources (such as computers, electricity), as customized solutions may require additional environmental resources [12].</td>
<td>Murugesan [25] proposes principles and practices for green IT.</td>
</tr>
<tr>
<td>Physical value w.r.t. cost (PVc)</td>
<td>Internal business perspective Represents the production value w.r.t. cost. A product being developed and marketed with lower development cost will have higher PVc</td>
<td>From environmental sustainability perspective, it should be ensured the resources are not wasted (adding to cost) during product development. However, efficiency can only contribute to, but not achieve sustainability by itself, see, e.g., Tomlinson et al. [34].</td>
<td>Silvius [31] takes transport, energy, waste, and materials into the resource balance. Poppendeick [28] proposes to eliminate “waste” in terms of partially done work, relearning, task switching, delays, defects etc.</td>
</tr>
<tr>
<td>Sustainability value of technology</td>
<td>Innovation and learning perspective The sustainability value of technology means how good or bad a technology is rated with respect to environmental impact due to its own production as well as usage during lifetime and later on for disposal.</td>
<td>If a new technology is being implemented, what is the environmental sustainability value of technology? Can it, e.g., increase interoperability possibilities to design more generic products/solutions?</td>
<td>Brown [3] provides insights and rationale to evaluate sustainability in technology for environmentally sound innovation.</td>
</tr>
<tr>
<td>Product intrinsic value</td>
<td>Customer perspective This includes functionality and quality attributes e.g. usability, reliability etc. of the product.</td>
<td>From environmental sustainability perspective, features and quality provided in the product has to be balanced w.r.t. resources used.</td>
<td>Byggeth [5] proposes a set of guiding questions for sustainable product development.</td>
</tr>
</tbody>
</table>

4 Illustrative Usage Scenario

The following dialogue is a fictitious discussion between Daniel and Mick. It illustrates the first steps of a usage scenario for a sustainability-driven application as depicted in Fig. 1. Mick is a product manager in a big car manufacturing company that wants to develop a car-sharing platform. Daniel is a method consultant from a well-established IT consulting company. They have already worked together in the past and generally get along well. The rich picture in Fig. 2 shows the most important elements of the car sharing platform. There is a community of users who can rent and share cars, there is a backend data base and there is a business infrastructure with maintenance, administration, and management. The speech bubbles indicate first starting points for the different aspects of sustainability. They first name the sustainability aspect and then, in parentheses, an exemplary respective value that can be considered for the car sharing platform.

**Sustainability as Objective** “The vice-president tells me that we need to focus on sustainability for the development project of that new platform—so, how do I do that?” Mick opens the discussion. “Well, that depends on the goals you want to achieve with respect to sustainability.” Daniel is a consultant well trained in first analyzing the problem and then developing a solution step by step with his customers, Fig. 1 Step 1. Mick sighs internally: “That’s not much
help—I don’t know what goals I can set for sustainability. I don’t even know of a precise definition what they refer to when using the term. To me, it seems like everybody has a different understanding of sustainability, so how can I come up with concrete goals for such a diffuse concept?”

Daniel is not surprised about this statement and provides a starting point: “Okay, I do agree that many people might have a different understanding of what sustainability is, but luckily there are concrete definitions out there that we can use to make the concept more tangible in your context, for example the one given by Robert Goodland …” and he quickly sketches the four dimensions of sustainability (given in Sec. 2): HUMAN, SOCIAL, ECONOMIC, and ENVIRONMENTAL, Fig. 1 Step 2.

“I see …” Mick acknowledges that it might be more than an abstract concept. Daniel takes that as an offer to further guide his customer: “Sure, that is still quite abstract, but it is the most general goal you can start with for that particular dimension of sustainability. From here on, we can refine the goal like any abstract business goal your vice-president might come up with.”

Mick is still skeptical: “I’m curious how you want to turn that into goals applicable to software-intensive systems development, but go ahead, we’ll give it a try.”

Daniel jumps up again to make more sketches on the whiteboard: “We can look
at that from the four different dimensions of sustainability.” He starts questioning: “What are current issues that need improvement? What are the values that are important here? What do your users want?” Mick is a little surprised by that question: “I guess they would want a good car sharing solution.” “That’s a start. However, think about what values lie behind the need for a car sharing solution, for example, the wish to save the costs for a car and the wish to do something for the environment by saving energy?” “Good point”, says Mick, “How can I structure my thoughts when trying to tackle the sustainability aspects for my product decisions and requirements elicitation?”

“I have that Software Value Map that gives a consolidated overview of common values for software-intensive product development. I have used it with a number of customers and it has proven sufficiently encompassing to be particularly helpful during analysis.” And he hands him a one-page introduction to the SVM as provided in Sec. 2. Mick scans it quickly but then returns to the discussion at hand: “Such a map is definitely useful for a start, but I need applicable guidance.”

**Human Sustainability** “Let’s first take a look at the human sustainability dimension”, Daniel continues (Fig. 1, Step 3). “According to the value map, the most important value that is relevant to be considered for human sustainability is HUMAN CAPITAL VALUE. Some of the metrics we can use to measure human capital are the general satisfaction of people and their impression of how their skills and knowledge develop over time. That would preserve human capital and therefore support human sustainability. So, how can we improve these two?” “Okay, I see. Let me think”, Mick picks up the thought, “User satisfaction depends on various factors, for example, the service costs and a good feeling when using the service. That’s an issue for both our interface designers, who optimize the user interaction with the system and its services, and our economics guys, who calculate the service prices.” “And service level agreements like a high availability of cars et cetera.” Daniel adds.

“Sure, and if we want to support the improvement of their set of skills and knowledge, we could offer, for example, an education program at the time of registration. Could that be a start?” Mick asks. “Of course!” Daniel replies, “Most of your users will be aware of the basics but you can still provide them with more information on the specifics of your service and the impact on the environment.” “Continuing that line of thought with knowledge and transparency,” Mick extends the idea, “we can perform an online evaluation of statistics and how much energy was saved in total, plus questionnaires that track even more, for example, which other means of transport they use apart from car sharing, and every user can optionally take part in that and gain knowledge on their individual statistics.” “... and, thereby, offer to provide them with additional information, yes, good idea.” Daniel agrees.

Mick starts taking notes on his To-Do template and scribbles facts here and there frowning at the piece of paper that is turning quite illegible. Daniel watches for a while and then proposes: “We have developed a template for this purpose that we call Impact Evaluation Pattern—maybe you would like to make use of it?” “Does that cost me extra?” grumbles Mick. “No”, shrugs Daniel, “it’s part of
the service.”

“Simply put the concept of Impact evaluation patterns was inspired by software design patterns. In software engineering, a design pattern is a general reusable solution to a commonly occurring problem software design. The same philosophy can be used to identify Impact Evaluation Patterns in different decision-making scenarios. An Impact evaluation pattern can be described as a generally reusable solution for a commonly occurring decision-making challenge in a particular scenario. For example, a product manager can use an Impact Evaluation Pattern for initial screening from sustainability perspective to decide if a set of new requirements should be selected for implementation in the product or not.”

On the second whiteboard in the room, Daniel sketches the template (Table 5, Fig. 1 Step 4): “Here you can see the basic structure for documenting the impact evaluation pattern—which is part of what we are discussing right now. I’ll keep explaining it while we continue.”

**Social Sustainability** Mick settles for that for the moment: “Okay, let’s continue with the social dimension. What have you got on your value map for that?” “There are customer capital values and network externalities. How is the relationship that you build with your customers? Are they loyal?”

“They like our cars, and once they had one their likelihood to buy the same brand again is about 80%. I’m not sure though whether that applies to car sharing as well. We could establish a bonus system for frequent users.” “Good! How about network externalities?” “What is that supposed to be?”

“It means the amount of other users of the software product that are relevant to the focal user. Applied to a car sharing service platform, we have to think about incentives we can offer a user for spreading the word about our service and making other people use it.” “Phew, you mean like family discount and stuff? The problem is that we diminish our revenue, so the business analysts are always reluctant to give such bonuses. However, we will find something.” “Alright, I’ll put it on the list.”

**Economic Sustainability** “Then let’s talk about the economic dimension”, says Daniel, “That’s the one you might already have sorted out the most. The values are maintainability, innovation value of technology and innovation value for market, differential advantage, and business agility.” “Yes, I guess we have elaborated on that for many hours.”

“Just what I thought, then why don’t we directly move on to the environmental dimension? That was one selling point of your campaign draft, right?” “Yes, of course”, agrees Mick.

**Environmental Sustainability** “Let’s see what your value list says”, Mick continues, “Market requirements value, physical value with w.r.t. cost, and so on. These first two are about resource saving, right? That would increase the respective values.”

“Yes”, Daniel agrees, “one of your market requirements would be to have an environmentally sustainable service, and you can consider various aspects for that—transport, i.e., local procurement, digital communication, traveling, and transport; then energy, i.e., energy used, and emission / CO2 from energy used;
waste, i.e., recycling and disposal; and materials, i.e., reusability, incorporated energy, and resources. The other part of resources and potential ‘waste’ to be considered are the working hours—on one hand spent in design and development and on the other hand as ‘waste’ in terms of partially done work, extra features, relearning, handoffs, task switching, delays, defects, etc.”

“That list is even longer than the one we calculated from—I will double check that with our business analysts. For a start, the transport in our service refers mainly to the vehicle availability. If there aren’t enough vehicles in a hot spot area, the service personnel have to move the vehicles accordingly. We definitely want to avoid that because it is costly in terms of emission and money. What is it with the sustainability value of technology?” Mick inquires.

Daniel replies: “You are potentially decreasing emissions by decreasing traffic—and I’m sure you will be able to find a lot more within an analysis of environmental optimization potential. You also have to evaluate how the desired system quality might affect the environment in a negative way, for example, by putting a lot of cars out there to ensure availability, you make a considerable impact again on the environment.” “True, we’ll have to perform an analysis on how they contradict.”

Table 5. Impact evaluation pattern for sustainability aspects

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Impact evaluation pattern for a product manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intent and Motivation</td>
<td>To perform a detailed value analysis with respect to sustainability.</td>
</tr>
<tr>
<td>Applicability</td>
<td>Analyze features to include in a product w.r.t. sustainability</td>
</tr>
<tr>
<td>Value aspects</td>
<td>Value aspects listed in Table 1-4</td>
</tr>
<tr>
<td>Impact evaluation criteria</td>
<td>See: <a href="http://www.bth.se/tek/aps/mkm.nsf/pages/software-value-map">http://www.bth.se/tek/aps/mkm.nsf/pages/software-value-map</a></td>
</tr>
<tr>
<td>Consequences</td>
<td>To be added when the pattern is put into actual use</td>
</tr>
<tr>
<td>Involved stakeholders</td>
<td>Product manager, domain expert, sustainability expert, process engineer, project manager</td>
</tr>
</tbody>
</table>

Impact Evaluation Pattern “Now we have a lot of scribbled notes on the whiteboard—what you called Impact Evaluation Pattern earlier on.” Mick gets back to the sketch on the whiteboard. “Yes.” Daniel emphasizes, “With the help of such impact evaluation patterns, the company can have tremendous benefits. An impact evaluation pattern presents a consolidated view of value aspects to be considered while deciding with respect to sustainability. Furthermore, it provides a common understanding and vocabulary, as well as acting as decision support to assure no value aspect is unintentionally overlooked. And finally, it enables a conscious impact evaluation (positive and/or negative) of relevant value aspects.” Mick likes decision support: “That’s good. And I can teach my staff to use the guideline instead of trying to mentally infuse my experience into them, as genuine experience cannot really be passed on.” “True”, Daniel adds, “and, furthermore, we can define corresponding rubrics that help to evaluate that selected measures, so you and your VP can see the direct impact.”

“Rubrics? What would be an example for that?” “For example, for human capital value, you can assess general satisfaction, time per year spent on continued education, and employee fluctuation.” “Okay, that makes it assessable for management—for these metrics, we already have some kind of reporting, so I know where to get the data from.” Daniel wraps up the discussion. “Alright, I
hope I could show you how you can use the value map to identify optimization potential with respect to the different dimensions of sustainability.” “Yes, thanks, I feel quite prepared now for the next meeting with the vice president.”

In the meeting with the vice president, Mick will realize Step 5 of the process in Fig. 1: the vice president will take the decisions and Mick is responsible for their implementation. The Impact Evaluation Patterns will be reused for the assessment of Step 6.

5 Discussion: Transfer to Practice

The presented usage scenario is the first attempt to illustrate the approach and can by no means substitute the evaluation in a sufficiently sized industrial case study. The preparation of such a case study is under way, but as we are looking for academic feedback in parallel we offer our concepts for early discussion.

We believe the approach is promising since its is based on a theoretically solid and empirically evaluated Software Value Map. The Software Value Map and impact evaluation pattern have been used in a case study at Ericsson for identifying value aspects to be considered for requirements selection from different stakeholders’ and they have been proven usable and useful. The industry practitioners did not only verify the benefits of having a consolidated view of value components, relevant for a particular Impact evaluation pattern, for decision-making; they also found the Software Value Map a step towards common definitions and understanding of value components enabling effective communication [20].

We expect to implement a similar case study for a sustainability analysis with equally positive results.

One important issue for consideration are conflicts that arise between different dimensions, as already mentioned in “Interrelationships” on p. 6. The explicit catalogue of values provides means to identifying such conflicts. However, the question of how such trade-offs can be solved while planning requires the prioritisation of goals. Which dimension will be considered most important in our future requires solutions in much broader terms than the approach at hand.

6 Summary and Future Work

In this paper, we have presented an approach to incorporate sustainability related value aspects while taking software product management decisions on the project, product, or portfolio level. Its usage is illustrated in a scenario where there is a dialogue between a consultant and a software product manager.

As a first effort to factor sustainability as primary aspect in value-based software engineering, the approach still needs to evolve and be explored. The proposed list of sustainability related value aspects is ready to use but not necessarily comprehensive; value aspects and perspectives can be complemented as needed. Our next step is to evaluate the approach in an industrial setting with adequate complexity to gain resilient feedback on its application in practice.

Future work is to improve upon how the identified sustainability aspects should
be measured in industry.

**Acknowledgement** This work is part of the EnviroSiSE project (grant: PE2044/1-1) funded by the DFG in Germany and the BESQ+ research project funded by the Knowledge Foundation (grant: 20100311) in Sweden.

**References**


32. B. D. Steven. Using the balanced scorecard process to compute the value of software applications. In *ICSE’06 Proceedings of the 28th international conference on software engineering*, Shanghai, China, 2006. ACM.


Part III

Concepts
The Concepts part contains the publications with regard to the reference model for sustainability [63], the analysis approach and artifact model [53, 76, 58], and the aspects of sustainability as quality characteristic in software systems [77, 73, 37]. These publications were partially developed in collaborations. The following list sets the context for the elaboration of each of the publications.

- **A Generic Model for Sustainability with Process- and Product-specific Instances [63]**
  This contribution was developed with Henning Femmer as coauthor. The publication includes insights from the student theses on the car sharing system [15] and the software development company [32]. It was published at the International Workshop on Green In / Green By Software Engineering in 2013.

- **Supporting Sustainability Aspects in Software Engineering [53]**
  This contribution was single-authored and describes the early stage of the concepts for analyzing sustainability for software systems. It was published at the 3rd International Conference on Computational Sustainability.

- **RE4ES: Support Environmental Sustainability by Requirements Engineering [76]**
  This contribution was coauthored by Bill Tomlinson and Debra Richardson from UC Irvine with whom I was developing the research proposal that funded a significant part of the research work for this habilitation. It describes in more detail the requirements engineering and quality analysis activities envisioned to support sustainability. It was published at the 1st International Workshop on Requirements Engineering for Sustainable Systems in 2012.

- **Infusing Green: Requirements Engineering for Green in and through software systems [58]**
  This contribution was single-authored and describes the usage of guiding questions and the artifact model to systematically include sustainability in requirements elicitation and analysis. It was published at the 3rd International Workshop on Requirements Engineering for Sustainable Systems in 2014.

- **Developing a Sustainability Non-Functional Requirements Framework [77]**
  This contribution was developed in collaboration with Ankita Raturi, Bill Tomlinson and Debra Richardson. It describes a framework for sustainability requirements and was published at the 3rd International Workshop on Green and Sustainable Software in 2014.

- **Safety, Security, ... now Sustainability! [73]**
  This article was coauthored by Ankita Raturi, Debra Richardson, and Bill Tomlinson. It describes an analysis of the lessons learned from safety and security that can partially be transferred to sustainability. This journal article was published in the IEEE Software Special Issue on Green Software in 2014.

- **Framing Sustainability as a Software Quality Property [37]**
  This article was coauthored by Patricia Lago, Ivica Crnkovic and Sedef Akinli Kocak and started as a collaboration at the 2nd International Workshop on Green and Sustainable Software. We developed a framework for sustainability analysis based on a draft standard for architecture evaluation and collected a number of cases to show its application. The journal article will be published in the Communications of the ACM in 2015.
9 A Generic Model for Sustainability with Process- and Product-specific Instances
A Generic Model for Sustainability with Process- and Product-specific Instances

Birgit Penzenstadler and Henning Femmer

Institut für Informatik
Technische Universität München, Germany
{penzenst|femmer}@in.tum.de

Abstract

Motivation: Software systems as we know them often have a economic purpose and/or fulfill human or social needs of their users. The economic purpose is analysed by economy itself; the latter goals are analysed in software engineering by user-centric techniques, such as service orientation. Yet, as software systems have an impact on the environment, environmental sustainability should be supported as a major goal for software development projects.

Problem: Without applicable guidance, sustainability remains an untangible ideal. Therefore, we need a definition and a concrete decomposition of sustainability to relate it to software systems development. It is not sufficient to analyse environmental sustainability on its own, but its interplay with other aspects in order to define appropriate actions and understand their effects.

Principal idea: We analyse the dimensions of sustainability, their values with respective indicators, and activities to support them. These elements compose a conceptual model that allows for analysing and constructing actions both for a company or a product point of view.

Contribution: We propose a generic sustainability model with instances for companies and projects from various case studies. We thus enable analysis, support and assessment of environmental sustainability in software engineering.

Categories and Subject Descriptors D.2.1 [Requirements Engineering]: [Elicitation methods]

Keywords sustainability, definition, requirements, generic reference model, case study

1. Introduction

Sustainability has been recognized as a relevant topic in software engineering, e.g., by the 2012 ICSE theme and a number of workshops. It refers to environmental, social, and economic aspects of software development and the usage of software systems. Most software systems are developed with an economic benefit in mind which shall also be preserved over a longer period of time. In case of software systems that are not intended to serve economic purposes, for example, open source software, the main purpose of the software system is a social benefit which can make specific work tasks easier or just plain entertainment and having fun. Taking into account that the economic and social aspects of sustainability are already analysed in most software systems, our work especially focusses on supporting the environmental perspective. Our proposed approach helps to integrate the objective of environmental sustainability in software systems as part of the requirements engineering. Hilty et al. [5] provide an analysis of the relevance of information and communication technologies for environmental sustainability and conclude that there is no such thing as a “general ICT policy for environmental sustainability”. For an adequate analysis, we need a tangible decomposition of the concept of sustainability and supporting methods in software engineering. These methods can then enable to include the concept of sustainability in software engineering and help to develop such a general ICT policy for environmental sustainability.

Problem: Currently, there is little guidance on how software engineering can contribute to improving the sustainability of the systems under development. Thus, sustainability is not tangible enough as a concept to actually transform it into software requirements. There is no approach available describing how to decompose sustainability for a concrete project and we are missing procedures for incorporating sustainability as an explicit goal into requirements engineering. Apart from the decomposition, we are also lacking a reference catalogue of sustainability-improving activities linked to indicators they affect. The latter would allow for a sustainability assessment in order to evaluate whether the activities affect the sustainability of a project or company.

Contribution: We present a reference model for sustainability that decomposes sustainability into five dimensions: environmental, individual, social, economic, and technical sustainability. The model provides activities and relates them to values they support and assessable indicators. It is intended to serve as a reference model for a process engineer who instantiates the model for a software development company or for a requirements engineer who instantiates it for a specific system under development. The instantiation is illustrated by two examples taken out of current case studies. Our aim is to show how the aspect least supported by our current ways of developing software systems, environmental sustainability, can be aligned with the other sustainability dimensions.

2. Related Work

We identified related work within frameworks for sustainable software engineering, strategy models, I* modeling, requirements engineering techniques, and goal modeling.

Framework for sustainable software engineering Naumann et al. [8] provide a framework for sustainable software engineering.
They investigate how web pages can be developed with little environmental impact, i.e., energy-efficiently, and offer a respective guideline for web developers. In such a framework, the reference model proposed in the work at hand could be used as key element in order to better promote the method.

**Strategy models** Gu et al. [3] propose a green strategy model that provides decision makers with the information needed to decide on whether to take “green” strategies and eventually how to align them with their business strategies. They consider such strategies as “green” which achieve lower energy consumption and perform a case study with Dutch data centers. In contrast, the paper at hand considers a broader definition of sustainability and thus gives a broader view on sustainable software engineering.

**Requirements engineering techniques** Mahaux et al. [6] present a case study on a business information system for an event management agency that advertises environment-friendly events. They assessed how well some current RE techniques support modeling of specific sustainability requirements in that case study. In contrast, our aim is to provide modeling means explicitly for integrating sustainability into the software development process as a major objective.

**Goal Modeling** Lamsweerde [10] decomposes business goals into system requirements, but does not explicitly reflect on sustainability. His work relates positive and negative influences between goals, but our work provides activities for direct realisation.

### 3. Definition and Dimensions of Sustainability

We consider five dimensions of sustainability as important for the analysis of software systems. The basic dimensions for a general sustainability analysis (without referring to software systems) are individual, social, economic and environmental as defined by Goodland [2]. The three latter ones are also known from the UN definition of sustainable development [9]. However, these four dimensions do not offer a possibility to claim and support long-term evolution of technical systems and adequacy for long-term use. Consequently, when looking at (software) systems, we need technical sustainability as an additional dimension.

**Individual sustainability** Individual sustainability refers to the maintenance of the private good of individual human capital. The health, education, skills, knowledge, leadership and access to services constitute human capital [2]. For software engineering (SE), we have to ask: How can software be created and maintained in a way that enables developers to be satisfied with their job over a long period of time?

**Social sustainability** Social sustainability means maintaining social capital and preserving the societal communities in their solidarity. Social capital are investments and services that create the basic framework for society [2]. For SE: Which effects do software systems have on the society (e.g. communication, interaction, government etc.)?

**Economic sustainability** Economic sustainability aims at maintaining assets. Assets do not only include capital but also added value. This requires to define income as the amount one can consume during a period and still be as well off at the end of the period, as it devolves on consuming added value (interest), rather than capital [2]. For SE: How can software be created so that the stakeholders’ long term investments are as safe as possible from economic risks?

**Environmental sustainability** Environmental sustainability seeks to improve human welfare by protecting natural resources. These are water, land, air, minerals and ecosystem services; hence much is converted to manufactured or economic capital. Environment includes the sources of raw materials used for human needs, and ensuring that sink capacities recycling human wastes are not exceeded [2]. For SE: How does software affect the environment during, inter alia, development and maintenance?

**Technical sustainability** From a point of view of (software) systems engineering, there is another dimension that has to be considered. Technical sustainability has the central objective of long-time usage of systems and their adequate evolution with changing surrounding conditions and respective requirements. For SE: How can software be created so that it can easily adapt to future change?

The dimensions Social, Economic, Environmental and Technical can be analyzed on a micro level as well as on a macro level. The decision on which level they are investigated depends on the scope of the system under analysis. These five dimensions are not necessarily encompassing. One could argue, for example, that politics & law should be a separate dimension. On the other hand, the government can be considered as institutionalization of society and, consequently, as subdimension of the social dimension. However, for the analysis scope of software systems, we believe the five given dimensions to be an adequate representation as they serve only as structuring means for the to-be-derived model.

### 4. The Sustainability Model

Our proposed method comprises the generic sustainability reference model (M1 level), the respective meta model behind it (M2), and the instances (M0) derived for specific processes (companies) and systems (software products).

#### 4.1 The Meta Model

![Figure 1. Meta model of generic sustainability model (M2).](image-url)
The meta model is comprised by the types *Dimension*, *Value*, *Indicator*, *Regulation*, and *Activity*: A `<Dimension>` is a viewpoint, represented by a set of values that express the abstract objectives of the dimension. Each dimension is represented by a set of values. A `<Value>` is a rationale that is rooted in itself, and is approximated by indicators. An `<Indicator>` is a qualitative or quantitative metric and is related to a value. A `<Regulation>` is an optional element that affects a value. An `<Activity>` is a means to support and influence a value.

4.2 The Generic Sustainability Model

An excerpt of the generic sustainability model is provided in Fig. 2, see [7] for details. The model consists of three levels: the top level contains the dimensions; the middle level contains values, indicators, and regulations; and the lower level contains activities. Each element in the generic sustainability model is an instance of a type from the meta model explained before. In the following we will explain examples from the generic model, structured by their type.

**Dimensions** A dimension is an aspect of or viewpoint on sustainability, for example, *environmental sustainability*. As described in Sec. 3, there are five dimensions to be considered when analyzing sustainability for software systems engineering. A dimension is detailed in a set of values.

**Values** A value is a moral or natural good that is perceived as an expression of a specific dimension. Each of the five dimensions is represented by a set of values. Values do not necessarily belong exclusively to one dimension but can be considered for a number of dimensions, for example, *healthy environment*, which applies for both the environmental as well as the individual dimension.

**Indicators** An indicator is a qualitative or quantitative metric that expresses a specific degree or score with regard to a value, for example, *satisfaction indices* as qualitative metric and *carbon emissions* or *return on investment* as quantitative metrics. A set of indicators approximates a value.

**Regulations** A regulation is an optional element that affects (i.e. supports or enforces) a value, for example, emission regulations. Regulations commonly set limits for a specific indicator to be of legal use. Many values belonging to the different dimensions are heavily regulated, either supported or restricted in order to protect them. For example, freedom of the individual is supported by the *human rights*, and healthy air is supported by the European Union’s directive on carbon emissions.

**Activities** An activity is a measure taken to contribute to a specific value or a set of values, for example, *use train for mid-distance traveling instead of aircraft*. The impact of these activities on a value is measured by the indicators it influences. For the travel example, using a train instead of an aircraft improves the emissions account of the traveler. For each value, there is a number of activities that can be implemented to support a value. Thereby, the impact of an activity is measured by the indicators it influences.

The generic sustainability model is intended to serve as reference and as a basis for the instantiation of company- or system-specific instances.

5. Instantiation of the Sustainability Model

The generic sustainability model can be instantiated for development processes (companies) and for software systems (products). For the former, a sustainability goal model that represents the goals of the company is created. For the latter, we see the analysis of the sustainability model as part of the requirements engineering for a software system under development. The elicitation of indicators is related to the elicitation of key performance indicators. The resulting activities and indicators are part of the controlling during software development.

5.1 Methodical Description

Applying sustainability modeling within a company consists of two alternating phases: the analysis and the application & assessment phase. These phases and their respective steps are carried out by a process engineer (for a company-specific model) or a requirements engineer (for a product-specific model). Whereas the analysis phase mainly concretizes the definition of sustainability in the present context, the application & assessment phase starts the selected activities and supervises whether the activities are working successfully. Even though these are two separated phases we expect these phases to be alternating, thus refining and tuning the specific instance of the goal model iteratively over time. During the whole process the model guides the tailoring of values, indicators and activities to the context (company- or project-specific) by structuring the goals and giving examples and suggestions.

**Analysis Phase** In the analysis phase the generic goal model is tailored to the context (a specific company or software system). This formalizes what sustainability really means for a certain company or product. As depicted in the center of Fig. 3, the analysis phase includes three steps:

1. For each dimension the architect needs to instantiate the generic values that are provided by the generic goal model into context-specific goals. These goals need to be prioritized to help solving conflicts between potentially contradicting goals.

2. Afterwards the architect defines activities to implement the goals and defines indicators that make it possible to assess the state of this goal at the moment as well as in future situations. In this step he might find trade-offs between indicators that are more exact with respect to this value and indicators that are easier to apply. For example, the architect might have to decide whether to measure resource consumption by weight or by item number. The generic goal model serves as a reference for selecting activities and indicators.

3. When values and indicators for a certain dimension are selected, the architect can now relate the activities that have a positive impact on the indicators. For example, he could find reusable items in this project and set up policies that enforce this reuse.

Indicators and activities are usually developed iteratively. The architect can choose whether it is more adequate for them to start with defining activities or to list the most important indicators for a specific sustainability assessment. This way we instantiated the generic model with goals, indicators and activities specific to a company or a product.

**Application & Assessment Phase** After formalizing the company’s or product’s specific sustainability goal model, two parallel tasks are needed: First of all, each activity needs a person in charge (“Sustainability Activity Responsible”) for the implementation of the previously selected activities. Second, the Quality Engineer needs to continuously monitor the company’s state with respect to its sustainability model. They can do this by assessing the list of indicators that are concretely defined in the model. The results of this assessment are reported to the management and the sustainability architect, so that changes or adjustments can be made accordingly.

In summary, the main goal of the application & assessment phase is starting activities and monitoring how the company or product is performing with regards to its own definition of sustainability.
5.2 Company-Specific Instances

An excerpt of one company-specific instance of the generic sustainability model is depicted in the left in Fig. 3. According to the method described above we first instantiated the generic model in a company-specific goal model in the analysis phase. This could work like the following: For environmental sustainability we instantiate the value Reduce resource consumption into the goal Reduce resources consumption by 30% within 12 months. We find three indicators for resource consumption: physical waste, the company’s energy bill and the size of the bought items list. We can immediately see differences in how easy one can measure these indicators. From these indicators we can select activities that have an influence on the indicators. For example to reduce the size of the bought items list, we can place incentives for less resource consumption. All values, indicators and activities can be refined within the model. For example, the unspecific activity reduce waste is refined within the subactivity recycle packaging for own shipping.

In the application & assessment phase we would now need to establish the activities and measure our indicators. From this information we can deduce how well our activities are performing and whether or not we work towards the company-specific sustainability goals. The first industrial case study for a company-specific instance is currently under development in a small software development company (jambit GmbH) in a series of workshops performed within a Master’s thesis.

5.3 Product-Specific Instances

The first product-specific instances of the generic sustainability model were developed in an industrial case study on a car-sharing platform built on the recently launched mobility service DriveNow. An excerpt of the product-specific instance of the car sharing system DriveNow is depicted on the right in Fig. 3. As explained above, the analysis phase is similar to company-specific instantiations. For environmental sustainability the value reduce environmental impact is chosen. We can select similar indicators from the generic goal model as in the company-specific instantiation, such as the energy bill. Yet, we need to tailor our activities specifically to this product. For example, in this system the usage of green data centers is advised. Furthermore, the business process can be varied by offering older cars for a lower rental rate (instead of depositing them) and thereby reducing waste, and by introducing cars with hydro or electric power. Again, in the application & assessment phase, we need to make sure that these activities are followed during the development of a product and measure the impact of activities.

6. Discussion

The previous example already shows both benefits as well as problems and open work of modeling with the generic sustainability model.

Benefits Many elements of the model (such as the physical waste indicator) reappear in various instances. Hence, much of the modeling can be reused and only needs to be selected. Second, sustainability, a rather abstract concept, is turned into a concrete and measurable property of a company or a product and corresponding actions that directly relate to improvement. By making these implicit definitions explicit via the model, it is possible to discuss and evaluate a company’s or product’s impact on sustainability.

Issues While creating the instances we were faced with the question of how to measure the impact of the proposed improvement activities for a product that does not yet exist. For example, when taking a product design decision such as using green data centers, there is no “before” that could be compared to an “after” situation as the system does not exist yet. In that case, the model can still serve well in a constructive way for decomposing sustainability goals and selecting activities to realise them, but the analytic phase after implementation reveals less concrete results.

Threats to Validity One threat to validity arises from our constructive research approach for the model. The risk is limited by our experience in developing such models, by reviews from and discussion with other researchers, and by feedback from industrial partners.
7. Conclusion

We propose a generic sustainability model with process- and product-specific instances that can help requirements engineers to analyse their projects according to the different dimensions of sustainability and choose actions for improvement. Special emphasis was put on environmental sustainability in and via software systems, as this dimension of sustainability is the one that is least supported by our traditional ways of developing software systems.

Future Work One important issue for future work is how to treat conflicting goals. Conflicts can be identified by contradicting influences on same indicators, but solving them is more complex. For example, the ever-present conflict between a tight budget and quality improvement measures is also found in the sustainability goals of a company—i.e. in indicators for economical vs. indicators for technical sustainability. However, more challenging is the question whether there are conflicts that are not as obvious and how to track them down if they do not relate to same indicators in the first place. The other major issue is to understand the reuse potential of instances (how systems-specific are they really?) and the gradual extension of the reference model by knowledge gained from instantiating the model in various companies. We expect continuous improvement by evolving the model over time in the ongoing cooperation with our industrial partners.

Acknowledgement We would like to thank Susanne Klein, Oliver Feldmann, and Jonas Eckhardt for helpful comments on drafts of this paper as well as Debra Richardson and Bill Tomlinson for feedback on early ideas.

References
10 Supporting Sustainability Aspects in Software Engineering
Supporting Sustainability Aspects in Software Engineering

Birgit Penzenstadler
Institut für Informatik, Technische Universität München, Germany
penzenst@in.tum.de

Abstract. Sustainability is not supported by traditional software engineering methods. This lack of support leads to inefficient efforts to address sustainability or complete omission of this important concept. Our aim is to support the dimensions of sustainability - human, social, economic, and environmental - within different phases of the software lifecycle, especially requirements engineering and quality assurance. We contribute a description of the aspects of sustainability in software engineering. The application of sustainability actions on the basis of these aspects is sketched in usage scenarios.

Keywords— sustainability, environment, software engineering, requirements engineering, quality assurance, guidance

Although many people are by now aware of the general definition of sustainable development as “meeting the needs of the present without compromising the ability of future generations to meet their own needs” [13], there still is no concrete guidance for the different aspects of sustainability that are observable from the point of view of software engineering.

Problem Traditional software engineering has not fully supported sustainability as a relevant, first-class concern. This refers to green IT topics, which represent a part of the technical realization possibilities for a sustainable software system, as well as the broader role of software in understanding and tackling the issue, for example, examining the sustainability of the business processes supported by a software system in its application domain.

Contribution We describe the different aspects of sustainability from a point of view of software engineering and exemplarily illustrate their consideration during requirements engineering and quality assurance.

1 Aspects of sustainability in the software lifecycle

Sustainability aspects can be brought to bear both during the development and use of software systems. We distinguish four aspects of sustainability. The first two focus rather on the developing company and its processes, while the latter two have the system under development in scope.
Development process aspect Sustainability in the initial software development process (with responsible use of ecological, human, and financial resources). For example, Naumann et al. propose guidelines for environmentally sustainable web development [12]. Lago et al. propose to measure the environmental impact of software services by their energy consumption [10].

Maintenance process aspect Sustainability of the software system during its maintenance period until replacement by a new system. This includes continuous monitoring of quality and knowledge management. For example, Albertao measures sustainability performance of a software project according to standard quality properties [6].

System production aspect Sustainability of the software system as product with respect to its use of resources for production, for example, by using green IT principles and sustainably produced hardware components. An administrative tool for strategic sustainable development is the ISO 14001 Environmental System Management Standard [3], which is embedded into a planning framework by MacDonald [8].

System usage aspect Sustainability in the usage processes in the application domain triggered by the software system as product. This takes into account responsibility in the impact on the environment and using green business processes. The probably most important step for analyzing and optimizing business processes is taken during requirements engineering, as proposed by Mahaux et al. [9]. Hilty et al. [7] propose a classification of ICT application types as starting point of such an analysis.

All four aspects are relevant for an encompassing approach to supporting sustainability in software engineering. However, we particularly emphasize the system usage aspect, as our hypothesis is that it might have the biggest impact in terms of improvement potential.

2 Usage Scenarios

The following usage scenarios exemplarily describe how software engineers, in particular a requirements engineer and a quality engineer, can apply various actions improving the sustainability of the software system under development. Requirements engineer Rebecca and quality engineer Quentin are developing software systems at Sustainable Software Inc. A new project is coming up: a car-sharing platform, to be developed within 4 months.

2.1 Requirements Engineering

After the kick-off meeting with their customers, Rebecca sketches a first draft of a Domain Model\(^1\) with a respective ontology. As a description of the operational

\(^1\) Items in italics are part of a domain-independent requirements engineering content model available at http://www4.in.tum.de/~penzenst/sources/DomainIndependent-RE-ContentModel.png
and business context, the model serves as communication basis for discussing and deriving goals during the next meeting. The explicit modeling of sustainability aspects of the domain, e.g., emission reduction, community building, etc., triggers a discussion on how to emphasize these aspects in a System Vision. The goals arising from the workshop are described in a Goal Model.

Using life cycle analysis and impact assessment on the ideas depicted in the System Vision, Rebecca manages to convince their customers of a more environmentally sustainable business process.

The Domain Model, the System Vision, and the Goal Model serve as inputs for the Usage Model. The latter is detailed in use cases and their scenarios to elaborate functional requirements as well as user stories to describe quality goals. For example, the quality goal “The system shall be easily maintainable”, Rebecca sketches a number of user stories, where one of them describes, e.g., how the database administrator edits tables at runtime. Her activities mainly contribute to the System Usage Aspect.

Rebecca hands over the results to her new colleague Quentin, curious how he likes the new way of representing the quality requirements. Quentin likes the user stories, as they enable a requirements engineer to illustratively describe quality requirements and to get the message across, thereby, providing a good interface to the quality model used by quality assurance.

2.2 Quality Assurance

At university, Quentin learned that the constructive part of quality engineering is always accompanied by the analytical part of quality assessment. For objectivity and traceability, Quentin wants to set up an assessment scheme with Key Performance Indicators (KPIs) that rely on common metrics, for example, the KPI library [4], the Environmental Sustainability Index [1], and the Sustainability Index [2].

For assessment of the project management, he uses the Sustainability Maturity Model [11], and for assessment of the product, he considers the sustainability metrics proposed by Albertao et al. [5]. His metrics show that Rebecca managed to achieve a considerable improvement in reducing energy consumption and emissions by optimizing the Usage Model of the car-sharing platform in her analysis and convincing her customer of the adaptations.

Following these activities, Quentin is continuously improving the support of sustainability in their product under development, and his progress can be measured. His activities mainly belong to the Development Process Aspect.

3 Conclusion and future work

This paper presented emerging research on supporting different aspects of sustainability via requirements engineering and quality assurance.

As preliminary evaluation, we are currently elaborating a case study with a major automotive company. This will be followed by a third-party evaluation via industrial collaboration in a different application domain.
Future Work: We envision an encompassing approach to build sustainability into software products with a corresponding assessment model. The author is currently working on a first evaluation of the content model and the scenario activities in student projects in a series of seminars\(^2\) and students’ theses.

For the Quality Model, we intend to combine adequate standard KPIs [4] and metrics used for the Environmental Sustainability Index [1] and the Sustainability Index [2] into a new Software Sustainability Index (SSI). A dashboard tool for ease of use will support the SSI.\(^3\)

References


\(^2\) http://www4.in.tum.de/~penzenst/teaching.shtml
\(^3\) The author would like to thank Bill Tomlinson and Debra Richardson for feedback on early ideas.
11 RE4ES: Support Environmental Sustainability by Requirements Engineering
RE4ES: Support Environmental Sustainability by Requirements Engineering

Birgit Penzenstadler\(^1\), Bill Tomlinson\(^2\) and Debra Richardson\(^2\)

\(^1\) Technische Universität München, Germany
penzenst@in.tum.de
\(^2\) University of California, Irvine, US
wmt@uci.edu, djr@ics.uci.edu

Abstract. [Motivation:] Environmental sustainability is an important concern. Information and communication technology (ICT) innovation is ambivalently positioned with regard to our rapid development and shortening innovation cycles. On one hand, information technology facilitates the (excessive) usage of resources. On the other hand, ICT can also help to significantly reduce human impact on the environment.

[Problem:] Environmental sustainability is currently not supported explicitly in requirements engineering (RE). This leads to the problem that (a) environmental sustainability is not yet given sufficient importance and (b) it is difficult to manifest in requirements & design and therefore hard to assess.

[Principal idea:] We need to combine the knowledge of RE, environmental informatics, and further disciplines, to develop an RE approach that tailors analysis, documentation, and assessment for ICT systems where environmental sustainability is a first class quality objective.

[Contribution:] This paper is a research preview on an approach to help requirements engineers handle sustainability as a first class quality objective. It elaborates on how we plan to refine and validate this approach in the future.

Keywords: requirements, sustainability, environment, requirements engineering, quality modeling

1 Introduction & Motivation

The most cited definition of sustainability is to “meet the needs of the present without compromising the ability of future generations to meet their own needs” [1]. Although our approach primarily aims at environmental sustainability, it must also be socially (and economically) sustainable in order to have practical significance [2]. As Mahaux [3] pointed out, we need a toolbox for supporting it in requirements engineering. We extend the idea of such a toolbox in this research preview and provide some of our drafts.

Problem: The use of information and communications technology (ICT) contributes significantly to the usage of our planet’s resources [4]. However, ICT
bears a lot of potential for “greening through IT” [5] by making our life more environmentally sustainable by technological support for our daily life; this is the context of our research. In contrast, Green IT or “greening of IT” is making hardware and software of ICT systems more resource-efficient; we do not focus on this. We must improve the environmental sustainability of humankind to protect our living space for future generations. Missing is a comprehensive understanding of how software engineering, and especially requirements engineering (RE), can help in this endeavor.

**Contribution:** We are analyzing what and how RE can contribute to the improvement of the environmental sustainability of ICT. We primarily focus on the development of ICT systems that have environmental sustainability in their explicit system vision (and abbreviate these systems with ICT4ES), because we assume the stakeholders of such systems to be more willing to adapt their development processes according to that quality objective. Our goal is to support the ICT4ES development with an adequate requirements engineering approach that integrates the knowledge of environmental informatics. This enables software engineers to handle sustainability as first class quality objective. Our research questions are:

**RQ1:** What are the implications for RE of ICT4ES, i.e., when making environmental sustainability a first-class quality objective for development?
For ICT4ES as we defined the term, environmental sustainability is an overall development goal. However, it is not clear how that impacts the requirements for a system. We seek to understand what is necessary to be taken care of when developing ICT4ES and how the business processes and business goals differ from those of traditional products.

**RQ2:** How can the necessities resulting from ICT4ES be implemented in an RE approach?
We aim at a toolbox to support the demands resulting from the goal of contributing to environmental sustainability. First, we analyze which artifacts are necessary to document the newly arising demands and what their concrete contents are. Then, we investigate which concepts have to be supported and which methods are required to elaborate these artifacts and how they have to be adapted.

**RQ3:** How can we assess the impacts of a given software system for environmental sustainability, including both direct and indirect effects, and considering different groups of stakeholders?
We elaborate metrics to measure environmental sustainability and provide an answer as to how a system can be proven to fulfill the sustainability requirements imposed upon it. Furthermore, we investigate an appropriate way to translate the requirements into acceptance criteria and how these criteria can be incorporated into an overall quality model.

2 Related Work

Sustainability is beginning to play an important role in software engineering, with the RE’08 keynote, the ICSE’09 Software Engineering for the Planet spe-
cial session, the CAiSE'10 panel, the WSRCC 2009, 2010, and 2011, and the conference slogan for ICSE'12. The first author of this paper completed a systematic literature review on sustainability in software engineering [6].

Amsel et al. [7] discuss ideas on how to support sustainability in SE. Cabot et al. [8] performed a case study for sustainability as goal for the ICSE organization with i* models to support decision making for future conference chairs. Naumann et al. [9] investigate how web pages can be developed with little environmental impact, i.e., energy-efficiently, and work on a respective guideline for web developers. Mahaux et al. [3] performed a case study on a business information system for an event management agency to assess how well some current RE techniques support modeling of specific sustainability requirements.

These works look at either a specific application domain or a specific development technique and adapt them to support sustainability modeling, while this project aims at an encompassing approach to be evaluated in various domains of ICT4ES systems. No other work yet proposes solutions for how to support quality modeling of environmental sustainability for software systems.

3 Approach to RE for ICT4ES

Our approach to RE for ICT4ES is planned in two phases: First, we conduct an analysis of domains as well as values and goals of the respective stakeholders, then we design a tailored RE method that supports the gathered specifics for ICT4ES (see Fig. 1). All activities described in this section are in progress, which means we have started but not yet completed them.

3.1 Analysis of Domains, Values, and Goals

Environmental sustainability can be supported by software systems in different ways, e.g., (a) information systems for environmental sciences, including climate models, earthquake warning, etc., (b) information systems that support green business processes, for example environment-friendly event management, and (c) embedded systems that lower our energy consumption. Therefore, we need to analyze the different types of domains that need support in explicitly addressing environmental sustainability in their software engineering approaches.

Based on the distinction of domains, we perform structured interviews in industry and academia with representatives from different domains. The interviews are followed by a systematic analysis and an interpretation that draws conclusions for the design of the envisioned method’s elements.

Starting with the results of the interview analysis, we elaborate a map of values for environmental sustainability and we detail the goals in a taxonomy, focusing on the ones that relate to requirements engineering for ICT4ES systems:

**Value map for environmental sustainability in SE** (RQ1) The value map shall put the value of sustainability into relation with traditional software engineering values as in the framework described by Khurum [10]. Her framework
relies on data gathered in interviews with practitioners and allows to create impact evaluation patterns from value maps.

**Goal taxonomy for sustainability in SE (RQ1)** The goal taxonomy decomposes and details the aspects of environmental sustainability from the point of view of software engineering. The input is the value map and for each value we can deduce supporting goals. Initially, most of these goals are independent of the system to be developed. Each of the goals is then decomposed hierarchically until the goals are sufficiently specific to be transformed into requirements.

![Fig. 1. Environmental Sustainability in Requirements Engineering.](image)

### 3.2 Design of a Tailored RE Approach

From the goal taxonomy, we gather requirements for artifacts, methods, and models for the documentation of sustainability requirements arising by deduction from the goal taxonomy with respect to a specific ICT4ES system. Based on these requirements and the knowledge acquired in the earlier phases of the project, we conduct an analysis and evaluation of different techniques, compare existing approaches, and develop a tailored RE approach including a quality model that provides indicators and metrics to assess environmental sustainability.

**Sustainability requirements artifact model (RQ2)** An artifact model gives guidance on structure and content to be elaborated when documenting sustainability requirements and related information like environmental impact, stakeholders, rationale, etc. Based on our experience [11], we develop an artifact model for representing sustainability requirements and related information.
Adapted analysis techniques (RQ2) To transition from goals to requirements and to adequately document these requirements according to an artifact model, we elaborate analysis techniques and documentation methods that form part of an RE approach tailored to ICT4ES. Solutions include adaptations of creativity techniques, life cycle analysis, environmental impact assessment and risk analysis techniques as well as handling of environmental information in form of data, statistics, and models.

![Fig. 2. Model-based Quality Assurance (adapted from [12]) & Quality Model Excerpt.]

Deduced quality model (RQ3) The quality model is built upon the input from the value map and the goal taxonomy. A quality model is a model with the objective to describe, assess and/or predict quality [12]. The activity-based quality model is elaborated on the basis of concepts proposed in [13]. It includes criteria for sustainability assessment as well as indicators and metrics to evaluate and measure a software system’s compliance to the sustainability requirements. Fig. 2 shows the model-based principle and an excerpt of the quality model draft.

Case studies (RQ1-3) The approach will be evaluated in industrial case studies, including the value map, the goal taxonomy, the artifact model, the analysis techniques, and the quality model. The qualitative evaluation will be implemented as a comparative study. The case study already under way is on car sharing; another one will be on an irrigation system.

4 Conclusion

In this research preview, we have introduced our ongoing research on a tailored RE method for ICT systems for environmental sustainability. The analysis phase investigates the domains and elaborates values and goals with the respective stakeholders. The design phase provides a tailored artifact model with analysis.
methods and a deduced quality model. Both will be evaluated in industrial case studies. We are preparing a guideline for the industry interviews and evaluate approaches from related disciplines in student seminars as described in [14] for preliminary studies.

Our contribution will provide software engineers with a toolbox to handle sustainability as first class quality objective. This enables “greening through IT” — to produce ICT systems that have positive impact on their surrounding eco-systems and therefore not only meet the needs of the present (by satisfying traditional quality objectives) but at the same time preserve the ability of future generations to meet their own needs (by meeting sustainability quality objectives). As software systems have a profound influence on many different facets of global civilization, including sustainability in the design of these systems has the potential to have transformative impacts on the world in which we live.

Acknowledgments: We would like to thank Martin Mahaux for providing feedback on an earlier version of this paper.

References
12 Infusing Green: Requirements Engineering for Green in and through software systems
Abstract—Environmental sustainability can be applied to software systems in two different understandings — either as green in software systems (greening of IT / green IT) or as green through software systems (greening by IT). Currently it is not clear how environmental sustainability can be systematically supported as an objective in requirements engineering for either of these two understandings.

This paper presents a checklist and guide word based approach that demonstrates how to include the objective of environmental sustainability from the very early steps in finding the stakeholders and analyzing the domain to the definition of a usage model and specific requirements. The elaboration is illustrated by a case study on a car sharing system.

As software systems affect most aspects of our daily lives, enabling software engineers to strategically align the objective of environmental sustainability with the other objectives for the software system under development could considerably decrease the impact of people in the industrialized world on the environment.

Index Terms—requirements engineering; environmental sustainability; green through IT; green software systems

I. INTRODUCTION: WHAT IS GREEN REQUIREMENTS ENGINEERING?

Over the last decades, sustainability research has emerged as an interdisciplinary area; knowledge about how to achieve sustainable development has grown, while political action towards the goal is still in its infancy [20], [56].

For a meaningful discussion, sustainability needs to reference a concrete system—such as an ecological system, a human network, or even a specific software system [54]. Software Engineering for Sustainability (SE4S) has developed as a current focus of research due to sustainability being advocated as major objective for behaviour change on a global scale [47], [48]. To denote an emphasis on environmental sustainability, the attribute green is widely used. At the same time, overall sustainability of our daily lives can only occur when the environmental, social, and economic aspects are in balance [59]. This has to be reflected in the software systems we create.

The term Green or Sustainable Software can be interpreted in two ways: (1) green in software: the software code being sustainable, agnostic of purpose (as in [41]), or (2) green through software: the software purpose being to support sustainability goals, i.e. improving the sustainability of humankind on our planet (as in [58]). Ideally, both interpretations coincide in a software system that contributes to more sustainable living. Therefore, in our context, sustainable software is energy-efficient, minimizes the environmental impact of the processes it supports, and has a positive impact on social and/or economic sustainability. These impacts can occur directly (consumed resources), indirectly (mitigated by service), or as systemic effects [23].

Requirements Engineering (RE) is the early phase of software engineering where we determine the exact scope of the system and iteratively elaborate the stakeholders’ needs and concerns [62], [42]. Based on that definition, Requirements Engineering for Sustainability (RE4S) [50] denotes the concept of using requirements engineering and sustainable development techniques to improve the environmental, social, and economic sustainability of software systems and their direct and indirect effects on the surrounding business and operational context. In order to develop such systems, we need awareness (by education) and guidance (e.g., as in this paper), and creativity (to find better solutions). Green Requirements Engineering consequently denotes that same concept with a specific focus on the direct and indirect environmental impacts of systems. However, as sustainability is an encompassing concept and one aspect of it cannot be strengthened without considering the other dimensions, we will still discuss the broader scope of it referring to all five dimensions.

Contribution: Instead of proposing a new framework that might interfere with established practices and be negated by software engineers, as sustainability is only one of many objectives for a system, we are following an integrative approach. It has been achieved to integrate safety and security as additional objectives into requirements engineering [12], [33], and now we propose to do the same for sustainability [49]. We show that requirements engineering can accommodate the new objective of improving the environmental sustainability of software systems using its current techniques and incorporating simply a few more instantiations of known requirements types. An extended version of this paper will be published in a book chapter [7] but we consider it important to get further feedback and stimulate discussion at the workshop.

Outline: We provide the background, related work and foundation in Sec. II, elaborate how to systematically integrate sustainability into RE in Sec. III, discuss the remaining challenges in Sec. IV, and conclude with an outlook in Sec. V.
II. BACKGROUND

We lay out the context from sustainability science and Information and Communication Technology (ICT) & sustainability, outline related work in RE, and describe the foundation for this work: the sustainability dimensions, their requirements, the reference model used for requirements documentation, and the case study used for the running example.

A. Sustainability Science

In general, sustainability is the “capacity to endure”, but interpreting this concept requires context. A popular definition of sustainable development was given by the UN as “meeting the needs of the present without compromising the ability of future generations to meet their own needs” [59]. Although it is not actionable, it is the most cited definition currently in use. Ultimately, sustainability depends on the population at large, so common conceptions of sustainability must be acknowledged: “People sustain what they value, which can only be derived from what they know.” [54] For that purpose, Joseph Tainter has suggested that it is useful to pose the four questions with regard to sustainability: “Sustain what? For whom? How long? At what cost?” [55]

More elaborate theoretic frameworks on the definition of sustainability exist, e.g. [5], [8], [13], [52], [2], [38], all referring to systems thinking [37], but are more extensive than adequate for the scope of this paper. Sustainability science is currently discussing whether sustainability is limited by ecological constraints or enabled by continuous innovation and transformation [34] and software systems might help in either direction.

B. ICT and Sustainability

The relevance of information and communication technologies for environmental sustainability is analyzed by Hilty, Arm-falk, Erdmann et al. [23]. On the basis of that, Hilty, Lohmann, and Huang [20] provide an overview of the fields of ICT in the service of sustainability: Environmental Informatics, Green IT, and Sustainable Human-Computer Interaction. As technological efficiency alone will not produce sustainability (cf. Jevon’s paradoxon [29]), sustainable development requires a combination of efficiency and sufficiency strategies, inter alia by decoupling economic growth from environmental impacts and from the use of natural resources.

Furthermore, the newly established conference on ICT for Sustainability (ICT4S) [22] is establishing a research community that emphasizes interdisciplinary research across various domains.

C. Related Work: RE and Sustainability

The workshop series on Requirements Engineering for Sustainable Systems¹ [3], [46] has brought a number of contributions on aspects like goal modeling, energy saving, complexity, sustainability-enhancing application domains, user participation, quality, and eco-aware design.

Furthermore, Naumann et al. [41] provide a framework for sustainable software engineering. They investigate how web pages can be developed with little environmental impact, i.e., energy-efficiently, and offer a respective guideline for web developers. Gu et al. [17] propose a green strategy model that provides decision makers with the information needed to decide on whether to take “green” strategies and eventually how to align them with their business strategies. In contrast to the focus on energy-saving in both of these works, we consider a much broader definition of sustainability.

Cabot et al. [6] report on a case study for sustainability as a goal for the ICSE’09 conference with i²-models to support decision making for future conference chairs. Stefan et al. [53] extend that with quantitative goal modeling techniques. Both works provide model instances for specific case studies while the work at hand uses a generic model for reference.

Mahaux et al. [35] present a case study on a business information system for an event management agency and assessed how well some RE techniques support modeling of specific sustainability requirements. In contrast, our aim is to provide modelling means for integrating sustainability into any software system as a major objective.

D. Dimensions of Sustainability for Software Systems

Sustainability is characterized by the three dimensions economic, social, and environmental [59]. This characterization is extended with a fourth dimension human (or individual) by Goodland [16], portraying individual development by every human over their life time. When analyzing the sustainability of IT systems, these four dimensions apply, and an additional dimension, technical, supports better structuring of concerns with respect to software systems. We are convinced that a focus on environmental sustainability only makes sense when in balance with the other dimensions of sustainability.

Most concisely, the dimension of sustainability are characterized as follows [45]: Individual sustainability refers to maintaining human capital (e.g., health, education, skills, knowledge, leadership, and access to services). Social sustainability aims at preserving the societal communities in their solidarity and services. Economic sustainability aims at maintaining capital and added value. Environmental sustainability refers to improving human welfare by protecting the natural resources: water, land, air, minerals and ecosystem services. Technical sustainability refers to longevity of systems and infrastructure and their adequate evolution with changing surrounding conditions.

E. Requirements Types for the Dimensions of Sustainability

For general characteristics of sustainability requirements for the respective dimensions, we have found the following subtypes that are used in other requirements categorizations [51]:

Environmental: Requirements with regard to resource flow, including waste management, can be elicited and analyzed by Life Cycle Analysis [19], [1]. Furthermore, impact effects can be analyzed by environmental impact assessment (EIA). The challenge is that usually only first order impacts by a system

¹http://www.ics.uci.edu/~bpenzens/2014re4easy/
are considered, whereas second and third order impacts are not yet accounted for.

Individual: Parts of individual sustainability are covered by privacy, safety, security, HCI and usability as well as personal health and well-being, which still needs to be made explicit in requirements. An example for this could be that an application suggests to take a break after a specific amount of working time.

Social: A share of social sustainability can be treated via computer supported collaborative work (CSCW [4]) requirements, which reflect the interaction within user groups, via ICT for development (ICT4D [18], [57]) requirements, and via political, organizational, or constitutional requirements, as in laws, policies, etc. Still missing are, for example, explicit requirements for strengthening community building.

Economic: The economic sustainability is taken care of in terms of budget constraints and costs as well as market requirements and long-term business objectives that get translated or broken down into requirements for the system under consideration. The economic concern lies at the core of most industrial undertakings.

Technical: The technical sustainability requirements include non-obsolescence requirements as well as the traditional quality characteristics of maintainability, supportability, reliability, and portability, which all lead to the longevity of a system. Furthermore efficiency, especially energy-efficiency and (hardware-)sufficiency [21] is part of the technical sustainability requirements.

Four of these five dimensions are already supported to a considerable extent by traditional software quality characteristics and requirements can be dealt with. The least support exists for the environmental dimension. Consequently, we especially need to consider second and third order impacts in the environmental dimension of software systems.

F. AMDiRE: Artefact Model for Domain-independent Requirements Engineering

The various influences on processes and application domains make requirements engineering (RE) inherently complex and difficult to implement. When it comes to defining an RE reference model, we basically have two options: we can establish an activity-based RE approach where we define a blueprint of the relevant RE methods and description techniques, or we can establish an artefact-based approach where we define a blueprint of the RE artefacts rather than a blueprint of the way of creating the artefacts. In the last six years, we have established several artefact-based RE approaches, empirically underpinned the advantages of applying those approaches in industry, and consolidated the different approaches and established the AMDiRE approach, i.e. the Artefact Model for Domain-independent Requirements Engineering. AMDiRE includes a detailed artefact model that captures the basic modelling concepts used to specify RE-relevant information, tool support, and a tailoring guideline that guides the creation of the artefacts [39]. For the purpose of this paper, we use a reduced version of the model as depicted in Fig. 2. For the full AMDiRE model, please refer to [39].

Guiding Questions for Green RE:
1. Does the system have an explicit sustainability purpose?
2. Which impact does the system have on the environment?
3. Is there a stakeholder for environmental sustainability?
4. What are the sustainability goals and constraints for the system?

Green requirements engineering may as well be applied with an activity-based approach though—the choice was taken for illustrative purposes, as the artefact-based approach provides an overview that is explicitly structured according to the work results. That way, the result excerpts can be seen in context with other requirements engineering work results.

G. Running example: Car Sharing System

In a collaboration with BMW in 2012, we elaborated a case study on the car sharing system DriveNow (partially reported on in [14]). The elicitation was carried out in an interview series with the DriveNow project leaders and by performing additional background research on the domains of hybrid cars, car sharing business models. The requirements basis established in that case study was used for elaborating the illustrative examples in the paper at hand.

The business model is commercial car sharing for registered users with flexible drop-off points for the vehicles on public parking lots. The car sharing system is composed of a web application for registration, reservation and billing, a car fleet maintained by a service partner where each car is equipped with a meter and a transponder, and a central database.

III. INFUSING GREEN: ELABORATING GREEN REQUIREMENTS

Based on the concepts presented in Sec. II, we describe how to elaborate green, sustainable requirements within a generic requirements engineering approach. The guiding questions are summarised in Figure 1.

Fig. 1. Guiding Questions for Green Requirements Engineering

Q1 Does the system under consideration have an explicit purpose towards environmental sustainability? If yes, this can be analysed in depth. If no, it can be considered whether such an aspect is desirable and feasible to add. If, again, that is not the case, then the analysis details the potentials for greening of that IT system (further explored in Q2) instead of greening through IT, but depending on the kind of system this might still lead to considerable improvements of the environmental impact of the system [43]. In case the system is widely used, that is worth the effort.

Q2 Does the system under consideration have an impact on the environment? Any system has an impact on the environment, as any system is applied in a real world context of some kind, which is situated within our natural environment. Consequently, it has to be analysed as to what are the direct (first order), indirect (second order), and systemic as well...
as potential rebound effects (third order). This potentially includes a very large scope, especially for third order effects, but systemic thinking [36] facilitates such an analysis process and may lead to significant insights.

**Q3** Is there an explicit stakeholder for sustainability? In case there is an explicit stakeholder who advocates for environmental sustainability, there is already a significant representative who issues objectives, constraints and considerations to support that quality in the system under consideration. In case there is no such advocate, it can be decided to establish such a role. Otherwise, at the very least, a domain expert should be established as a representative for sustainability for providing information on applying environmental standards, legislation, and regulations.

**Q4** What are the sustainability goals and constraints for the system? Independent of whether the system has an explicit purpose for supporting environmental sustainability or not, there certainly are a number of objectives that pertain to the different dimensions of sustainability that may be chosen to apply. For example, a social network might not have an explicit environmental purpose, but it certainly has objectives supporting social sustainability. Furthermore, any system will at least have some constraints with respect to the environment, as stated in Q2.

For the description of how to elaborate green requirements, we limit ourselves to a few concepts that are commonly agreed on as content items or information elements for gathering and refining requirements, all depicted in the overview in Fig. 2. These are Business Processes, Domain Models, Stakeholders, Objectives, Constraints, System Vision, Usage Model, as well as Quality Requirements, Process Requirements, Deployment Requirements, and System Constraints. There are a number of potential starting points for green requirements engineering with related elicitation and analysis activities as illustrated in Fig. 2:

- In case there is a relation of the business process of the system under consideration to sustainability or environmental issues (see Q1), the Business Process Model is the first piece of information that may explicitly include green concerns in the form of supporting business processes or services. If that is not the case, then there will still be elements in the Domain Model that can be related to sustainability concerns, due to the impacts caused by the system (see Q2). This is denoted by the activity analyse sustainability of context.

- If the business context and application domain lack adequate root elements for a sustainability analysis, the Stakeholder Model may be used as starting point (see Q3), characterised by the activity find sustainability stakeholders. Whichever the system under consideration, the stakeholder model should include a sustainability advocate, at least as representative for legal constraints.

- In either case — a system with an explicit sustainability concern as well as without such a mission — the Objectives & Goals should feature sustainability as one major quality objective (see Q4). This objective should be included in the general reference goal model of a company used as a basis for instantiation for a particular system and then refined according to the system specifics. Apart from elicit sustainability objectives from the stakeholders, it is also necessary to elicit sustainability constraints from the domain model for the Constraints & Rules, which includes sustainability-related constraints for any kind of systems, for example, environmental standards.

From these different starting points, the sustainability requirements and constraints are propagated throughout the content items in requirements engineering as illustrated in Fig. 2. This includes the activities derive sustainable system vision, specify sustainable interaction and refine and deduce sustainability requirements. The following sections walk through these stages and describe the development of the respective content items.

In the example of the car sharing system, the business model aims at promoting cars as a service (as opposed to an owned vehicle), but also includes the objective of reducing
environmental impacts by focusing on hybrid cars and by providing a car sharing service.

A. Analyse Sustainability of Context

Whenever we are faced with a system that has an explicit contribution to sustainability by either improving our ways to analyse the environment and reporting feedback, or by enabling and incentivising sustainable behaviour in its users, we can analyse the contextual elements related to in the Business Processes and the Domain Model. Examples for such systems are the Carbon Footprint Calculator\(^2\), the Story of Stuff Project\(^3\), or car sharing systems like Zipcar\(^4\) or DriveNow\(^5\). If there is no explicit purpose for environmental sustainability, there might still be a purpose for social sustainability, for example different types of local community tools or social networks. Either way the system will cause some kind of impact on the environment, which can span from first order impacts to third order impacts. The system environment and wider context are usually analysed using a Domain Model. This can also serve as the basis for a life-cycle analysis \([30]\) of the system under consideration.

For the car sharing system, first order effects are the resources that the system itself (the application on different devices and the database) consume. The second order effects are the resources the car sharing system triggers in its application domain, i.e. the cars that are being shared on the road and that do consume a considerable amount of resources, but at the same time decrease the overall consumption of resources through more cars (which would have been used otherwise). The third order effects might be a decrease in the number of individually owned cars, less parking space shortage, and eventually less cars, which would lead to better air quality, but this remains to be observed in the long run.

B. Find Sustainability Stakeholders

Stakeholders are the basis for requirements engineering. They pursue goals, include the users of the system under development, and issue constraints \([15]\). In the context of green requirements engineering, the goal is to elicit stakeholders that advocate for sustainability and that are domain experts for life cycle analysis, environmental concerns, legislation for environmental regulations, or environmental standards.

There are different possible approaches to identifying stakeholders for sustainability \([45]\):

1) **Reference list**: Instantiating generic reference lists of stakeholders for the concrete project context (see \([45]\) for a generic list of sustainability stakeholders).
2) **Context**: Inspecting the business and operational context of the system under development, and understanding which concrete roles are involved.
3) **Goals**: Iteratively analysing and refining a generic sustainability model \([44]\) and deducing the related roles.

This is especially helpful for finding passive stakeholders who do not have an active interest in issuing own goals, but whose constraints have to be adhered to, for example legislative representatives.

The example in Fig. 3 illustrates and excerpt of the stakeholder model for the car sharing system, including the ones that advocate sustainability or serve as domain experts on its various aspects.

![Stakeholder Model of the Car Sharing System](image)

C. Elicit Sustainability Objectives, Goals and Constraints

The next step is to elicit the sustainability objectives and goals from the stakeholders and to deduce any sustainability constraints from the business processes and/or the domain model. A **goal** is an objective the system under consideration should achieve \([28]\). They build a hierarchy, and they can influence each other in terms of conflicts, constraints, or support.

To facilitate goal elicitation, we distinguish three subcategories that refer to different levels of abstraction in systems development: **Business Goals** are all business-relevant (strategic) goals as well as goals with direct impact on the system or project. **Usage Goals** are a direct relation to the functional context and usage of the system (user perspective) for behaviour modelling. **System Goals** are system-related goals that determine or constrain system characteristics \([39]\). Each usage goal is related to a business goal and each system goal to a usage goal.

In order to consider the sustainability perspective during goal modelling, we consult a reference model for sustainability, that represents the sustainability dimensions by sets of values. Values are approximated by indicators, supported by regulations, and contributed to by activities \([44]\).

Instantiating the generic sustainability model for a specific system is feasible in a case whether sustainability is the major purpose of the system under consideration. For most systems sustainability will be one amongst a number of objectives, therefore it is more suitable to develop one overall **Goal Model**

---

\(^2\)http://coolclimate.berkeley.edu/carboncalculator
\(^3\)http://storyofstuff.com/
\(^4\)http://www.zipcar.com/
\(^5\)http://www.drive-now.com/
for the system and to detail the submodel for the objective of sustainability by using the sustainability dimensions and the generic sustainability model as a reference. This means to analyse the generic sustainability model and to decide for each value within the dimensions whether it is applicable to the system under development and, if so, to select those related activities which can be operationalized as goals for the system.

D. Derive Sustainable System Vision

The next step is to derive a sustainable System Vision, a common vision of the system under consideration agreed upon by all stakeholders that have an active interest in the system. One frequently used method to create system visions that are easy to communicate is Rich Pictures [40]. A rich picture is a cartoon-like representation that identifies all the stakeholders, their concerns, and some of the structural and conceptual elements in the surrounding work context. The choice of an easy to understand medium instead of a more formal and detailed one arises from the need that stakeholders of various domains and disciplines have to understand the vision. The system vision is usually coupled to a milestone with the scope of an early draft of a common idea of the system. It can be used as an early basis for estimations and planning of the subsequent development process. Furthermore, it is a detection basis for moving targets. In case the purpose of the system is closely linked to sustainability, this shall become very clear in the vision. In case it is a minor aspect, it may still be expressed as one of the concerns.

The system vision defines the system scope and comprehends the system context (business context as well as operational context), which is intended to realise a number of Features. A feature is, in our understanding, a prominent or distinctive user-recognisable aspect, quality, or characteristic of a system that is related to a specific set of requirements, whose realisation enable the feature [9]. Furthermore, it denotes the most important stakeholders and their concerns, in order to serve as communication basis with all stakeholders, including non-technical ones. As example, the system vision for the car sharing system is depicted in Fig. 5. It was elaborated in discussion with various stakeholders from the respective industry domains (car manufacturer, fleet operator).

E. Usage Model

With the system vision established, the next step is to derive the interaction with the user. The content item Usage Model details a Use Case Overview in its Use Cases and Scenarios. We distinguish Services and Use Cases. Both concepts are means to describe (black box) system behaviour. Use Cases describe sequences of interaction between Actors (realising user groups) and the system as a whole. More precisely, a use case represents a collection of interaction scenarios, each defining a set of interrelated actions that either are executed by an actor or by the system under consideration [10]. For each use case, there is at least one Functional Scenario in which Actors participate. Use Cases and Scenarios can be represented in the form of structured text (for example, the Cockburn template [10]), in UML use case diagrams, in UML activity diagrams, and in message sequence charts. For the car sharing system we illustrate the example scenario of a commuter user who wants to carpool.
F. Refine and Deduce Sustainability Requirements

Finally, detailed sustainability requirements and constraints are refined and deduced in four categories: Process Requirements, Deployment Requirements, System Constraints, and Quality Requirements. Further concerns for the system or the project may be managed in a Risk List.

Process Requirements denote demands with regards to the conducted development, for example using a green software engineering process. They constrain the content and/or structure of selected artefact types and the process model, i.e., the definition of the milestones regarding time schedules, used infrastructure like mandatory tools, and compliance to selected standards and approaches like to the V-Modell XT. They are mostly described in natural language text.

Deployment Requirements specify demands with respect to the installation of the system and launching it into operation, for example the migration of the data of the legacy system to the green data center used for the system under development.

System Constraints detail restrictions on a system’s technical components and architecture as well as related quality attributes, for example hardware sufficiency, i.e. that the system shall run on the old hardware without resource-intensive upgrades. They describe its functionality by means of single atomic actions, and its quality by means of assessable system quality requirements. We consider concepts that describe the transition to logical and technical architecture layers according to [60]. Hence, we see a system as a grey box rather than as a glass box, since we restrict systems’ internals, but do not consider their logical structure by interacting components, interface specifications, and functions. They are usually described in natural language text.

Quality Requirements describe the demands for individual quality attributes across a system’s functionality, the satisfaction criteria of those requirements, the qualitative or quantitative metrics, and how the metric will be evaluated. They characterise the attributes of the system either coupled to a specific functionality or as a cross cutting concern. They are usually represented in the form of natural text. Quality requirements are assessed by Measurements that can be either a Normative Reference (e.g. a GUI style guide) or a Metric. Quality Requirements constrain System Actions and can be satisfied by Generic Scenarios. We make use of quality definition models as by Deissenböck et al. [11].

The Risk List includes a description of all risks that are related to project-specific requirements, usually in the form of natural language text. The conceptualisation of requirements risks is considered on the basis of an artefact model [26], [27]. The risks are implied by the various types of requirements and we use the risk list as an interface to risk management.

Example Use Case and Scenario for Carpooling
Primary Actor: Commuter User
Goal in Context: The purpose of this feature is to enable users to carpool, which saves all parties resources.
Preconditions: The user is already registered with the system.
Description:
1) The user logs into the system.
2) The system displays the user's dashboard with statistics, offers, billing, and profile data.
3) The user clicks on the carpool button from the homepage of the system.
4) The system prompts him with a form for entering the carpool requirements.
5) The user enters his home address, destination, and date & time for the desired ride.
6) The system displays either the option to select from possible rides he could add to as a passenger or to offer this ride as a separate new carpooling option.
7) The user selects the best available option to add on to an existing ride as a passenger.
8) The system confirms the ride and informs the driver.

Variations:
7a) The user selects to offer a new ride as carpool.
7b) The system acknowledges the new ride and informs the user that the ride will be offered for passengers until 3 hours before departure time.

Example Process Requirement for the car sharing system
• Develop the system according to the guidance provided by Software Engineering for Sustainability (SEfS) and as described in the book “Green Software Engineering”.

Example Deployment Requirement for the car sharing system
• The testing period with the pilot users will be carried out on the server in the Munich Green Data Center.

Example System constraint for the car sharing system
• The web application shall use Waze as navigation system on handheld devices.

Example High-priority quality requirements for the car sharing system
• High availability (economic, individual and social sustainability)
• High usability, easy to use (individual and social sustainability)
• Affordable (individual and economic sustainability)

Example Risk List for the car sharing system
• Peak times and commuting might lead to an accumulation of cars in specific areas, leading to low availability in other areas.
• Excessive usage could cause high energy demands with peaks that cannot be satisfied by renewable energy. In that case, more gas would have to be used.
• Users are not active enough, therefore no community would be established, and consequently there is no contribution to social sustainability.

By going through these steps, we have obtained detailed sustainability requirements that can be traced back to their respective origins in the Business Process, the Domain Model, the Stakeholder Model, or the Goal Model. From here on, the responsibility for ensuring that the requirements are designed into the system and eventually implemented moves on from the requirements engineer to the system architect and the designers.

IV. DISCUSSION

In this section, we reflect on the mapping of sustainability dimensions to content items, and discuss requirements conflicts, cost modeling, legal constraints, and risk management.
A. Which Dimensions Appear in which Content Items?

Independent of the applied artefact model, it is interesting to take a look at the mapping of sustainability dimensions to content items. Table I gives a coarse-grained mapping of requirements content items to the sustainability dimensions that they contain information from. For example, a system vision will generally include mainly environmental and social sustainability aspects (as it abstracts from technical details), the system constraints will feature many constraints from the environmental and technical sustainability dimensions, and the process requirements will include demands from the environmental, social, and technical dimension.

<table>
<thead>
<tr>
<th>Content Item</th>
<th>Sustainability Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder Model</td>
<td>all</td>
</tr>
<tr>
<td>Goals</td>
<td>all</td>
</tr>
<tr>
<td>System Vision</td>
<td>mainly environmental and social</td>
</tr>
<tr>
<td>Usage Model</td>
<td>social and economic</td>
</tr>
<tr>
<td>Quality Requirements</td>
<td>technical</td>
</tr>
<tr>
<td>Deployment Requirements</td>
<td>technical</td>
</tr>
<tr>
<td>System Constraints</td>
<td>environmental and technical</td>
</tr>
<tr>
<td>Process Requirements</td>
<td>environmental, social and technical</td>
</tr>
<tr>
<td>Risk List</td>
<td>all</td>
</tr>
</tbody>
</table>

B. Sustainability Requirements Conflicts

In traditional qualities considered during software engineering we already face a number of potential conflicts, for example between code maintenance and code performance, or between the development time and the desired quality of a software system. Consequently, the question arises what kinds of conflicts exist between the five dimensions of sustainability and their related goals.

The economic dimension aligns with the environmental one in terms of resource savings (energy, materials, waste), but they may conflict when it comes to additional certifications, building a (environmentally and socially) sustainable supply chain, and turning to more expensive alternative solutions in case they are more environmentally friendly. The reason for that is mainly that up to now, the negative environmental impacts that are caused by our economy are hardly charged. Therefore, the goal of environmental sustainability does not get assigned monetary value but only image value, which is likely to be ranked secondly. These conflicts are also discussed in [44].

Another potential conflict, at least for some systems, is a trade-off between energy efficiency and dangerous materials. This is one potential goal conflict in case energy efficiency would require using more dangerous material. Although not a software system in itself, a lightbulb might serve as example: New energy-saving lamps are much more energy-efficient than the old light bulbs, but at the same time contain toxic mercury that imposes a threat when a lamp breaks as well as phenol, naphthalene and styrene. In the case at hand, considerate users will make sure the lamp is not in close proximity to their heads, but as legislation has banned the old lightbulbs already in some countries, they will have to be used for now. Resolving such a conflict for a particular case means to assign weights to each of the goals and prioritise whether the energy saving is greater or whether the risk and long-term negative impacts of the dangerous materials are greater.

C. Cost Modelling

Another aspect worth discussing is the connection between stakeholders, goals, and cost modelling. The stakeholders are made explicit in the goal model by tracing back to the rationale of a goal, as the information source (e.g. a domain expert) or the issuer of a goal. With respect to assigning costs to the goals there is a limitation, as this only makes sense for business goals, but not for values that cannot be expressed in return on investment. Some goals, for example the protection of the environment, do not have monetary value in themselves and their qualitative value is hard to measure. At the same time, it is important to define measures to ensure the realization of these goals and to show that the approach can make a difference in those resulting measures. Consequently, instead of assigning costs to the sustainability goals, their contribution to higher causes must be made explicit, for example the contribution to objectives commonly agreed on by governments like the sustainable development goals from Rio+20⁴, or the Vision 2050 [61].

D. Legal Constraints

As a consequence of the fact that environmental goals have not yet been prioritized sufficiently by the economy, legislation has established a number of environmental regulations that companies have to adhere to. These regulations will still be extended in the future, which makes legislation probably the most important stakeholder representing environmental sustainability in particular. Individual and social sustainability are also taken care of by law, for example by worker’s rights, which are supported and represented by worker unions.

It would be interesting to see at which point we need new laws and a different legislation to make sure that important questions of sustainability are incorporated into IT systems. Furthermore, it would be interesting to look at other examples such as functional safety and also to a certain extent security, where such laws exist.

E. Risk Management and Environmental Sustainability

Risks, safety and security all strongly relate to sustainability; risks need to be managed in order to enable sustainability, and safety and security are part of sustainability.

Safety is part of individual and social sustainability for preserving human life (no injuries) and environmental (no chemical or other hazardous accidents), but also has aspects in economic sustainability (a product that is not safe will not let a company reach long-term economic goals).

Security is also part of various dimensions, the technical one as it is a standard quality attribute for systems, then

⁴http://www.unccd2012.org/
individual and social (as the users shall be protected), and as a consequence of that also the economic dimension (insecure systems will not have market success).

Both of these quality aspects have not been around forever, but they were introduced as explicit qualities for software systems after the first safety hazards and the first security threats occurred. Consequently, we can learn from this development for systematically incorporating sustainability into software engineering [49].

V. Conclusion

This paper provided an overview of how green requirements engineering may be conducted within the scope of general purpose requirements engineering by asking guiding questions along the way and providing plugs for additional analysis activities that inform the development of environmental issues that should be considered. This approach was supported by illustrating examples and a discussion on different types of conflicts and traceability of information across different requirements engineering content items.

The impact of our contribution is mainly determined by the question how much difference the consideration of sustainability actually makes in requirements engineering. If we can make a sustainability purpose explicit in a system, then the difference is significant. If such a purpose is not given, secondary influence can be achieved by adding sustainability objectives and greening the system itself. The latter has less impact on the environment but is still feasible, the more the bigger the user community of a system. In the long run, the author's hypothesis is that we will not be able to end resource depletion by greening existing systems but only by disruptive change and completely transforming our systems [32]. Creating the mindset for that starts with acknowledging the need for incorporating sustainability as an explicit objective in systems development.

Another way to ensure prioritizing environmental sustainability is to enforce policies based on the compliance-driven economy. One open issue is the standardisation of (environmental and general) sustainability as explicit quality objective in software development, for example within the IEEE 830 Recommendation for Software Requirements Specifications and the ISO 25000 on software quality, informed by the ISO standard families on environmental management [24] and social responsibility [25].

The path towards software engineering for sustainability7 requires a mindset of awareness (by business analysts and developers) and methodical guidance (as provided in this paper), and creative confidence (as in [31]). If sustainability policies and standards are put in place, and software engineers prioritize them in the systems they develop, future software systems may significantly contribute to indirectly influencing the behaviour of users who interact with those systems and enable us to move towards a more sustainable global society as illustrated in the Vision 2050 [61].

7http://www.se4s.org

ACKNOWLEDGEMENTS

I would like to thank Ankita Raturi, Debra Richardson, Daniel Mendez, Henning Femmer, Alejandro Rodriguez, Oliver Feldmann, Susanne Klein, Manfred Broy, Daniel Pargman, Joseph Tainter, Lorenz Hilty, Bill Tomlinson, Juliet Norton, Marcel Pufal, Coral Calero, Xavier Franch, Wolfgang Lohmann, Martin Mauhaux, and Beth Karlin for helpful and inspiring discussions as well as Joachim Kolling from BMW for supporting the case study. This work is part of the DFG EnviroSiE project under grant number PE2044/1-1.

REFERENCES

13 Developing a Sustainability Non-Functional Requirements Framework
Developing a Sustainability Non-functional Requirements Framework

Ankita Raturi, Birgit Penzenstadler, Bill Tomlinson, Debra Richardson
Department of Informatics
University of California, Irvine, CA, USA
| araturi | bpenzens | wmt | djr | @uci.edu

ABSTRACT
Requirements engineers are in a unique position to encourage the consideration of sustainability at a formative phase in the software development life cycle. In this paper, we look at how we can develop sustainability as a non-functional requirement (NFR). We describe an NFR framework that is informed by sustainability models and discuss how it can be used to appropriately elicit and describe sustainability related requirements of the software system to be developed. We outline a roadmap for how we may integrate sustainability in requirements engineering from a theoretical NFR framework to an applicable software quality and relevant software standards.

Categories and Subject Descriptors
D.2.1 [Software Engineering]: Requirements/Specifications

General Terms
Theory, Design, Standardization

Keywords
Sustainability, environment, green IT

1. INTRODUCTION

“An axe can easily be made from recyclable steel, but it will still have a negative environmental impact if used to clear-cut a forest.” [23]

Software is ubiquitous in industrial society. It is like the steel in the quote above; its impacts depend to some extent on how it is produced, but often even moreso on what people use it to do. The broad functional potential of software causes it to span the spectrum of environmental impacts, from profoundly positive to deeply negative, depending on how and what we build.

Requirements engineering involves eliciting, analyzing, documenting and maintaining the complex set of requirements for a software system. This process scopes the system and considers who the stakeholders are, what is to be built, and how it will eventually be deployed into some context. As such, requirements engineers are able to influence the goals and functionality of the software system, and in particular make recommendations about intended quality attributes. Requirements engineering presents an opportunity to start making choices regarding sustainability that could affect both the software development life cycle and the life cycle of the software system itself.

We have only recently begun exploring the role that requirements engineering can play in sustainability, and in particular how sustainability should be considered a “first class software quality” [20]. We know that safety and security requirements evolved to a point where they are now considered important software qualities. An argument has been made that sustainability requirements need a similar evolution[19]. The next step is to actually instantiate the idea of sustainability requirements. In order to develop sustainability requirements for software, we need to begin asking questions like: what aspects of sustainability can software support? How can we prioritize concerns? For example, how do we assess whether it is more important to build a system that consumes less energy or a system that has intrinsic features that enable resource tracking? In the requirements phase, the primary questions would revolve around how to address sustainability issues pertinent to the software system being constructed.

In this paper, we try to address some of these questions by looking at how we can develop a Sustainability Non-Functional Requirements (SNFR) framework that is based on facets of sustainability as identified by sustainability science. Our aim is to describe a method by which this can be done and how this can influence software standards and software development methods.

In Section 2, we described what it means for us to consider sustainability as a general goal and how this can be incorporated into software requirements as a non-functional requirement or software quality. Section 3 describes the SNFR framework in more detail, breaking it down into components derived from sustainability science. We also discuss two concrete examples to show how the SNFR framework can be used to describe the sustainability requirements of a software system. Section 4 deals with how we can take this compartmentalized view of the SNFR and incorporate it into existing and potential software standards as a way to actively influence what requirements are elicited and eventually how the software is constructed. Section 5 outlines a brief
roadmap of how the considerations made in this paper can be a part of the transition of software engineering towards sustainability. Section 6 contains some closing remarks and conclusions of the paper.

2. MOTIVATING SUSTAINABILITY

A potential adaptation of the Bruntland definition of sustainable development [3] to software engineering could be: Software engineering has the ability to support sustainability by ensuring that the software systems developed and deployed meet the needs of the present without compromising the ability of future generations to meet their own needs.

Work at the intersection of information technology and sustainability often appears in the form of the “greening” of IT itself. This area, often called “Green IT,” is “the study and practice of designing, manufacturing, using, and disposing of computers, servers, and associated subsystems ... efficiently and effectively with minimal or no impact on the environment” [14]. However, the software subsystem is often neglected. A few models, for example Naumann’s GREENSOFT model [15], support self-reflection in the software engineering life cycle to reduce negative environmental impacts. However, sustainability is a concept not often addressed within software engineering from a software requirements perspective.

On the other hand, when IT is used to support sustainability, as in the examples discussed in Tomlinson’s book on Greening through IT [24], there are few existing software engineering methods to assist in the elicitation and description of sustainability requirements. Requirements engineering (RE) presents an opportunity to intervene early in the life cycle of software development and influence design decisions to support sustainability. There is interest in the software engineering community for considering sustainability during RE [11, 20], as well as some preliminary examples of sustainability requirements [12].

To date, however, there is no explicit method for the elicitation, analysis, documentation and maintenance of sustainability requirements for a software system.

3. A SUSTAINABILITY NFR

There are a variety of ways in which Non-Functional Requirements (NFRs) have been defined with respect to the way in which they are contextualized, instantiated or satisfied. NFRs “constitute the justifications of design decisions and constrain the way in which the required functionality may be realized.” [10] NFRs are attributes of specific software quality [6] (in this paper, we are referring to sustainability). The IEEE standard 1061-1998 (Standard for a quality metrics methodology) states that “Software quality is the degree to which software possesses a desired combination of quality attributes” [1] where the definition of these qualities depends on what is required of the system to be built.

Research efforts with regards to a classification of requirements, especially NFRs, have resulted in some standards, e.g., the ISO/IEC 25000 family. At the same time, there is still active discussion in research on the correctness of various frameworks for characterizing requirements [5]. However, classification of requirements is a means, not an end, and therefore all frameworks that prove to be useful are appreciated in different contexts. The question is what a good classification framework for sustainability requirements might be.

Due to the broad nature of sustainability, and the variety of application domains, the set of possible requirements is very large. Considering sustainability as a multi-faceted NFR allows us to analyze software requirements with respect to each of these facets. We can map concepts from sustainability science to inform a Sustainability NFR (SNFR).

3.1 Orders of Effect

With the increased pace of innovation in our society, software engineering faces shorter and faster development cycles. This speed of development means that the goals are mostly immediate. As a long term issue, therefore, sustainability tends to be neglected. In a report regarding sustainability and Green IT, Mingay describes three stages of “The Effects of ICT on Environmental Sustainability” [13]. These orders of effect are as follows:

1. First Order: Direct result of existence such as Greenhouse Gas (GHG) emissions, e-waste, hazardous substances, and use of scarce and nonrenewable resources.
2. Second Order: From application. These include travel substitutions, transportation optimizations, e-business/e-government, and the effects of environmental control systems.
3. Third Order: Long term socio-economic structural changes in the form of energy intensity, GHG intensity, transportation intensity, and material intensity.

When detailing a particular sustainability requirement, we should consider how far an impact that requirement, and thus the software, can, and should reach. We will later use these orders of effect to guide the way in which requirements are scoped. The orders of effect are perceived after a certain amount of time, therefore it may be useful to think of them as a temporal scope of a software requirement.

1. First Order: Direct effects of use of a software system. They are immediate and localized.
2. Second Order: Indirect effects of use of the software system in it’s application environment.
3. Third Order: Broad scale socio-economic structural changes that occur as a result of use of a software system.

3.2 Sustainability Dimensions

Goodland describes four types of sustainability [8]. These can be described as follows:

- Human Sustainability: Maintenance of human capital. Health, education etc.
- Social Sustainability: Maintenance of social capital. Focus on community continuance.
- Economic Sustainability: Maintenance of economic capital.
- Environmental Sustainability: Maintenance of environmental capital, including preservation of resource and protection of the environment.
Table 1: Sustainability Non-Functional Requirements Framework

<table>
<thead>
<tr>
<th>Sustainability Dimensions</th>
<th>Environmental</th>
<th>Human</th>
<th>Social</th>
<th>Economic</th>
<th>Technical</th>
</tr>
</thead>
<tbody>
<tr>
<td>First: Orders of Effect</td>
<td>1-E</td>
<td>1-H</td>
<td>1-S</td>
<td>1-Ec</td>
<td>1-T</td>
</tr>
<tr>
<td>Second:</td>
<td>2-E</td>
<td>2-H</td>
<td>2-S</td>
<td>2-Ec</td>
<td>2-T</td>
</tr>
<tr>
<td>Third:</td>
<td>3-E</td>
<td>3-H</td>
<td>3-S</td>
<td>3-Ec</td>
<td>3-T</td>
</tr>
</tbody>
</table>

Facets
- Representation: [Operational, Quantitative, Qualitative, Decorative]
- Satisfaction: [Hard, Soft]
- Kind: [Function, Data, Performance, Specific Quality, Constraint]
- Role: [Prescriptive, Normative, Assumptive]

Penzenstadler adds:

- **Technical sustainability:** long-time usage of systems and their adequate evolution with changing surrounding conditions and respective requirements [17].

It appears as though software engineers currently treat these dimensions separately, if and when they are explicitly addressed in requirements. When building a software system, the dimensions appear in the form of things like cost of energy efficient algorithms (environmental), people hours involved in development (human), development team management (social), development (economic) and code maintainability (technical). However, in order for software engineering to transition to sustainable software, the dimensions of sustainability need to be treated with respect to what software is built (the requirements), and not just how it is built (what we currently look at). It is to this end that we adopt Penzenstadler’s adapted multi-faceted model of sustainability [17].

### 3.3 Existing Sustainability Considerations

Some aspects of the sustainability dimensions overlap with other non-functional requirements. For general characteristics of sustainability requirements for the respective dimensions, we have found the following subtypes:

- **Environmental:** Requirements with regard to resource flows, including energy, can be elicited and analyzed by Life Cycle Analysis (LCA). Other aspects are efficiency and time constraints. The problem is that usually only first order impacts by a system are considered, whereas the second and third order impacts are not even in the conscience of the developers because they won’t be held responsible for them.

- **Human:** Parts of human sustainability are covered by privacy, safety, security, and usability. In addition, there is a strong focus on personal health and well-being, which still needs to be made explicit in requirements. An example for this might be that an application suggests to take a break after a specific amount of working time.

- **Social:** A share of social sustainability requirements are considered in the form of political, organizational, or constitutional requirements, as in laws, policies, etc. What is still missing are, for example, explicit requirements for strengthening community building, and requirements that encourage socially responsible use and behaviors.

- **Economic:** The economic sustainability is taken care of in terms of budget constraints and costs as well as market requirements and long-term business objectives that get translated or broken down into requirements for the system under consideration. The economic concern lies at the core of most industrial undertakings.

- **Technical:** The technical sustainability requirements include non-obsolescence requirements as well as the traditional quality characteristics of maintainability, supportability, realiability, and portability, which all lead to the longevity of a system. Furthermore, especially energy-efficiency and (hardware-)sufficiency [9] should be part of the technical sustainability requirements.

This list shows that four of the five dimensions are already supported to a considerable extent by traditional software quality attributes. The least support exists for the environmental dimension. Consequently, we especially need to consider and better support the second and third order impacts in the environmental dimension of software systems. One way to address these impacts would be to describe sustainability requirements as a cross product of the five dimensions human, social, environmental, economic, and technical times the orders of effect.

### 4. SUSTAINABILITY NFR FRAMEWORK

In this section, we utilize the aspects of sustainability discussed in Section 3.2 to inform a SNFR framework. An NFR framework [6] is a method for reasoning about different decisions while putting together the requirements of a software system. Table 1 describes the 15 different subcategories for sustainability requirements. It combines the sustainability dimensions and orders of effects in order to create fields that allow for breadth in coverage of sustainability topics as well as cater to the temporal scales within sustainability. We can also use the orders of effect to temporally scope a requirement when applicable. The primary challenge, for sustainability goals [3], as with many other software quality attributes, lies in operationalization, validation and verification.

Glinz proposes a faceted classification of requirements (see Figure 1) to allow for increased specificity and potential for evaluation of sustainability requirements [7]. The intent is, that for each sustainability requirement, in addition to belonging to a sustainability dimension and having a certain order of effect, one would also specify the form of representation, the kind of requirement, how a requirement is satisfied, as well as the role the requirement plays. We gain the advantages that Glinz describes of such as classification,
Table 2: Classified Sustainability Requirements from Joulery

<table>
<thead>
<tr>
<th>SNFR</th>
<th>Requirement</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-E</td>
<td>Energy efficient algorithms should be utilized for the aggregation, visualization and storage of data.</td>
<td>This is to minimize the energy of the tool itself.</td>
</tr>
<tr>
<td></td>
<td>Facets: Quantitative representation, Soft satisfaction, Performance kind, Prescriptive role</td>
<td></td>
</tr>
<tr>
<td>1-S</td>
<td>The Joulery tool should be released under open source licensing.</td>
<td>This makes it a contribution to the general software development community’s toolkit.</td>
</tr>
<tr>
<td></td>
<td>Facets: Declarative representation, Soft satisfaction, Constraint kind, Prescriptive role</td>
<td></td>
</tr>
<tr>
<td>1-Ec</td>
<td>Data should be visualized in a meaningful manner.</td>
<td>This will enable development teams to make more informed energy decisions with less effort, costing fewer staff-hours.</td>
</tr>
<tr>
<td></td>
<td>Facets: Declarative representation, Soft satisfaction, Specific Quality kind, Prescriptive role</td>
<td></td>
</tr>
<tr>
<td>1-T</td>
<td>The Joulery tool should be built in a modular fashion.</td>
<td>This will allow for the tool to be extended to include new types of devices due to monitoring needs.</td>
</tr>
<tr>
<td></td>
<td>Facets: Declarative representation, Soft satisfaction, Specific Quality kind, Prescriptive role</td>
<td></td>
</tr>
</tbody>
</table>

which include: separation of concerns, easier identification of verification method, the capacity for precision and completeness, and clarifying the boundary of the requirements [7]. These are especially helpful when considering a complex concept such as sustainability with respect to the hard to grasp non-functional requirements of seemingly intangible software systems.

![Figure 1: Glinz’ faceted classification of requirements [7]](image)

This classification model allows for the modeling of the sustainability requirements of a software system as follows:

**Root Goal:** Support Sustainability

**Leaf Requirements:** Each of the fields in 1, 1-E, 2-E . . . 3-T represents distinct requirements. Each has both a component of sustainability that it covers, as well as an order of effect that can either be expressed temporally, or in terms of impact as put forward by Mingay [13]. Either way, each field is two-dimensional with breadth and depth. In addition, each requirement detailed using Glinz’ facets.

We will now demonstrate the SNFR framework in action by classifying requirements for two tools: the Joulery Energy Awareness tool and a Hotel Resource Tracking System.

### 4.1 Brief Example: Joulery Energy Awareness Tool

Joulery is a lightweight monitoring tool that consolidates and visualizes energy use information across networked devices. The goal is to enable software developers to be more aware of the energy footprint of development activities, and help identify energy sinks in networked environments [21].

We can use the SNFR framework to classify sustainability requirements relevant for the system to be built. As we are retroactively analyzing the Joulery tool for illustrative purposes, we only describe a subset of the requirements. With this in mind, some of the sustainability requirements of Joulery has are shown in Tab: 2. It shows very short sighted requirements that do not include any second or third order effects of the tool, or any in the human sustainability vector. The Joulery tool is clearly one that is geared toward reducing the energy consumption and reducing the negative environmental impacts of software development. It is in itself a tool for the greening of IT. Table 2 sample requirements as a result of using the SNFR framework.

### 4.2 Extended Example: Hotel Resource Tracking System

To demonstrate a more diverse use case of the SNFR framework we also analyzed a hypothetical Hotel Resource Tracking System (HRTS). This system was originally described as a case study for an undergraduate Requirements Engineering course in the Informatics department at the University of California at Irvine [22]. Over the course of 10 weeks, 18 teams of four to five students developed software requirement specification documents for the HRTS with several intentional sustainability issues in mind. Table 3 shows a set of sample requirements that are elicited and categorized using the SNFR framework.

**System Description:**

HRTS is a centralized system to be used by a hotel management. It tracks resources associated with specific events. Events are described as time limited activities involving a specific set of rooms in the hotel associated with a guest(s). There are two components of the HRTS: The Management Subsystem (MGMT), which is available to the the management team, contains resource consumption detail of every room in the hotel. The Guest Subsystem (GUEST) is available to each guest of the hotel and details resource consumption of the guest during their visit to the hotel and allows for guests to make sustainable decisions regarding their stay.

A resource tracking event begins when guests check into a particular hotel room. All resources consumed by these guests within the room will need to be entered into HRTS. Upon guest checkout, the resource tracking event is closed and a final resource footprint is available for the event. Some of the primary resources include (but are not limited to) water,
5. EVALUATING THE SNFR

Requirements need to be validated and verified. Validation means ensuring with the stakeholders that these are the requirements they had, and verification means ensuring that the requirements were correct. To be able to determine whether a requirement has been satisfied, it is necessary to define success criteria. These can be quantitative, with metrics and test cases, or qualitative, including a reasoning about how they will be verified.

During design, different solution options may present themselves and the architect or designer has to determine the best fit to the given set of requirements. For evaluating these options, we need to “expose the positive or negative influence of alternative options on non-functional requirements” [25].

In that case, we are not only looking at the satisfaction of one particular requirement, but instead at a set of requirements and how well it is satisfied or negatively impacted by a particular design decision.

Not every design decision necessarily influences all NFRs but usually they influence at least a subset of them instead of only an individual requirement. For example, for Joulely, the design decision of how to implement data visualization shall on one hand be energy efficient, and on the other hand meaningful. So a more complex visualization might be more meaningful but will also require more energy. Consequently, a tradeoff is required.

6. INTEGRATION WITH STANDARDS

Large software development teams are the most likely to have the largest impact on global sustainability simply due to the size of their footprints compared to small development teams. In addition, they are more likely to use standards in one form or another, and IEEE standards provide this consistency in the complicated world of software. This means that a potentially useful intervention mechanism would be to embed the Sustainability NFR Framework in existing IEEE standards.

The Software Requirements Specification (SRS) is one of the most important documents that is produced during requirements engineering as it contains the scope of the software and what it will do, in addition to supporting the previously stated goals [25].

ISO/IEC/IEEE Standard 29148-2011 addresses requirements engineering processes and products for systems and software engineering. As part of the recommended content of a System Requirements Specification (SyRS), there are certain clauses that can be interpreted as sustainability related. Sub-clause 9.4.11 on Environmental conditions reads: “Include environmental conditions to be encountered by the system. The following areas should be addressed: natural environment (e.g., wind, rain, temperature, flora, fauna, fungus, mold, sand, salt spray, dust, radiation, chemical, and immersion); induced environment (e.g., motion, shock, noise, electromagnetic, thermal); electromagnetic signal environment; self-induced environment (e.g., motion, shock, noise, electromagnetic, thermal); threat; and cooperative environment. Consideration should also be given to legal/regulatory, political, economic, social, and business environment.” [2].

This section is followed by subclause 9.5 which details the software component of a system and the normative content of a Software Requirements Specification (SRS). For the software component, there are no such environmental considerations, constraints, requirements or attributes. What is listed is a partial list of example software system attributes that includes Reliability, Availability, Security, Maintainability and Portability. This encourages the requirements engineers and subsequent engineers involved in the software development life cycle to explicitly and actively consider these concepts. It is important for sustainability to be included and explicitly called out in the SRS so as to allow for specification and later objective verification of whether or not a software system supports environmental, human, social, economic and technical sustainability issues. A potential Sustainability attribute may read:

9.5.17 f) Sustainability

Specify attributes of software that relate to the fulfillment of sustainability goals relevant to the software type and deployment context. These goals are enabling environmental, human, social, economic, and technical sustainability. Specific requirements in this area could include the need to:

1. Reduce resource consumption;
2. Design for non-obsolescence;
3. Enforce humane practices;
4. Utilize energy efficient algorithms;
5. Utilize open source licensing when possible;

Looking at the other software qualities that are defined in ISO/IEC/IEEEStd 29148-2011, we see that they are either quite specific (see Security) or quite vague (see Reliability). In addition to specificity, and based on a review of these attributes, two other criteria may be related to the success of the attribute, even to the extent that entire subsfields are dedicated to study a particular attribute.

The second criterion is visibility. This is the idea that the client knows to ask about it. During requirements elicitation, a client may not know to bring up sustainability as they are unaware that this is a consideration that can be made within software. However, most clients tend to know that they should consider basic security features. While they may not know that it should, for example, involve a cryptographic technique, they are still able to articulate their needs as something even as simple as, ‘I need to protect my account from being accessed by other people.’

The third criterion is support. Developers are able to support a certain software quality when they know how to or are enabled to by the community. Taking the example of security again: If the client wants to protect their account, first, the requirements engineer is aware that this is technologically feasible and that there is an actual method to protect the account. In addition, when this requirement propagates through the rest of the software development life cycle, the different participants know how to translate that requirement into design and later code. This technological support is missing when trying to consider sustainability.

The specificity, visibility and support of an attribute are closely related with the maturity of a successful software
<table>
<thead>
<tr>
<th>SNFR</th>
<th>Requirement</th>
<th>Hypothetical Hotel Resource Tracking System Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-E</td>
<td>The HRTS system shall track resource consumption in the hotel. Resources to be tracked are: energy, water, food, linens, and stationary. Facets: <em>Quantitative</em> representation, <em>Hard</em> satisfaction, <em>Data</em> kind, <em>Prescriptive</em> role</td>
<td>This allows for the short term monitoring of resource to influence day to day resource savings. The impacts of this requirement are direct and immediate.</td>
</tr>
<tr>
<td>2-E</td>
<td>The system shall track hotel wide resource inputs and outputs that will be stored in an external resource tracking database in a standardized format. Facets: <em>Quantitative</em> representation, <em>Hard</em> satisfaction</td>
<td>If other industries have a similar practices, it would enable cross sector resource tracking, that allows for life cycle considerations and analysis of repercussions of the tourist industry on the ecosystems.</td>
</tr>
<tr>
<td>3-E</td>
<td>In the long run, the system is envisioned to become the quasi standard for HRMS by providing a good example in reducing negative environmental impact. Facets: <em>Declarative</em> representation, <em>Soft</em> satisfaction,</td>
<td>If the system can set a good example and help hotels with impact reductions, it may be adopted by a representative share of the industry.</td>
</tr>
<tr>
<td>1-H</td>
<td>The MGMT subsystem shall provide data about resource consumption averages per guest and enable guests with below average footprints to be rewarded. Facets: <em>Operational</em> representation, <em>Hard</em> satisfaction,</td>
<td>This would allow guests to be rewarded for displaying sustainable behaviors.</td>
</tr>
<tr>
<td>2-H</td>
<td>The GUEST subsystem shall provide hotel guests with a useful interface that displays their resource footprint and associated impacts. Facets: <em>Qualitative</em> representation, <em>Soft</em> satisfaction,</td>
<td>This allows guests to view their resource footprint to make more sustainable decisions, encouraging engagement in cleaner living practices.</td>
</tr>
<tr>
<td>3-H</td>
<td>The GUEST subsystem shall provide hotel guests with persistent historical resource consumption data across all their visits to the hotel. Facets: <em>Operational</em> representation, <em>Hard</em> satisfaction,</td>
<td>This would enable guests to analyze their resource impact over time, that may encourage long term individual behavior change.</td>
</tr>
<tr>
<td>1-S</td>
<td>The system shall encourage participation as a community in greener living practices by providing a means for management and guests to interact. Facets: <em>Quantitative</em> representation, <em>Soft</em> satisfaction,</td>
<td>Having a way to share information about reducing resource consumption within the hotel can provide a playful incentive to engage the community.</td>
</tr>
<tr>
<td>2-S</td>
<td>The system shall provide data about the hotel’s adherence to existing sustainability and corporate responsibility practices. Facets: <em>Declarative</em> representation, <em>Soft</em> satisfaction,</td>
<td>This would allow managers to assess how well the hotel is doing at conforming to their organizational sustainability requirements (if any).</td>
</tr>
<tr>
<td>3-S</td>
<td>The system shall provide the option for a guest to log into their HRTS account and save the data of that particular stay to their account. Facets: <em>Declarative</em> representation, <em>Hard</em> satisfaction,</td>
<td>Assuming successful establishment of the system as quasi standard, community engagement and guest incentives would be facilitated industry wide.</td>
</tr>
<tr>
<td>1-Ec</td>
<td>The MGMT subsystem shall provide inventory re-stocking recommendations for resources (only stationary, linens, food). Facets: <em>Declarative</em> representation, <em>Hard</em> satisfaction,</td>
<td>This would reduce the amount of time and effort employees (such as housekeepers) spend tending to each room in the hotel.</td>
</tr>
<tr>
<td>2-Ec</td>
<td>The MGMT system shall identify and report high resource consuming areas of the hotel. Facets: <em>Declarative</em> representation, <em>Soft</em> satisfaction,</td>
<td>This reporting and analysis would allow managers to make decisions about how not only resource consumption but costs can be cut down.</td>
</tr>
<tr>
<td>3-Ec</td>
<td>The system shall enable financial cost cutting due to reduced resource consumption in hotels. Facets: <em>Qualitative</em> representation, <em>Soft</em> satisfaction,</td>
<td>Under the assumption resource tracking system actually encourages reductions, there would eventually be reduced costs incurred in the hotel industry.</td>
</tr>
<tr>
<td>1-T</td>
<td>The system shall be energy efficient. Facets: <em>Quantitative</em> representation, <em>Soft</em> satisfaction,</td>
<td>This would ensure that the HRTS system itself was not a drain on resources. A threshold for what is considered low consumption would be needed.</td>
</tr>
<tr>
<td>2-T</td>
<td>The system shall require minimal specialized hardware. Facets: <em>Declarative</em> representation, <em>Soft</em> satisfaction,</td>
<td>In order for the system to be widely used and easily maintainable, generic or easy to access hardware would be needed.</td>
</tr>
<tr>
<td>3-T</td>
<td>The system shall support easy data exportation. Facets: <em>Operational</em> representation, <em>Soft</em> satisfaction,</td>
<td>Data portability is required for sharing of resource consumption data between people, hotels and sectors</td>
</tr>
</tbody>
</table>
quality. Therefore, in addition to adding a Sustainability attribute to the SRS standard, we should also consider a standard dedicated to Recommended Practice for Sustainability Quality Attributes for the SRS. This would detail the Sustainability NFR Framework starting with the classification system, describe specific attributes scoped by the orders of effect, describe the facets of the requirements, and finally instantiate through example. In order to ensure examples are possible a small set of ‘Sustainability Widgets’ would need to be built. These could be in the form of carbon tracker libraries that can pull geographic environmental data from existing datasets (for environmental requirements), reputation systems (for social requirements), health monitoring systems (for human requirements), open sourcing of different kinds of libraries (for economic requirements) and energy awareness tools for developers (for technical requirements).

This multi-pronged approach of a dedicated and detailed recommended practice document accompanied with some support for developers could allow for the increase in specificity, visibility and support that we believe is necessary for sustainability to be successfully adopted by the software engineering community as a software quality.

There is also a very specific process by which IEEE standards are proposed and developed. It involves creating an open working group, engaging the community in a conversation regarding what the content of the standard should look like, partnering with industry for both input and application potentials and so on. While that process is very structured, based on a brief review of current standards in development, it appears as though people tend to come into this process with a very set idea of what they are trying to achieve to begin with as opposed to exploring options. The early days of developing a standard are bound to be more organic as the group is building from very little history at the intersection of the two expansive fields of sustainability and software engineering (in particular, requirements engineering). We detail a roadmap in the next section for developing sustainability as an NFR, leading to requirements conducive to validation and verification.

7. ROADMAP

In order to understand how sustainability requirements would look like outside of the SNFR framework proposed here, it would be useful to start by taking a look at what is being done already. We have conducted a systematic mapping study [18] that is available in addition to an earlier systematic literature review [16].

1. Evaluation via Industrial Case Study: The next step is to conduct a case study to ascertain what kind of considerations, if any, are being made for sustainability during the requirements engineering process. This involves review of any SRS documents, interviews with the requirements engineers and potentially with the people involved in the realization of certain requirements when necessary. Specific research questions/areas here include:

   - **Existing sustainability requirements:** Are there any explicit sustainability requirements stated in the SRS? If there are, this would then spawn a series of tasks involving tracing the requirements through the development process to see how they dealt with sustainability in their software system.

   - **Implicit sustainability requirements:** There is potential that certain requirements, while not explicitly stated as sustainability requirements may fall under the Sustainability NFR framework. It would be interesting if there are, as this means that while considerations are being made for sustainability, there is a lack of visibility of the topic, understanding of how these requirements fit under sustainability, or a lack of interest in demarcating sustainability as a section of its own.

   - **Potential for improvement:** What is the potential for additional Sustainability NFR considerations to have been made for the system being studied? What were the missed opportunities, potential costs and alternatives that could have made the software a more sustainable system?

   This will give us a deeper understanding of what the sustainability landscape looks like from a requirements perspective and provides insight into the client’s and the requirements engineer’s sustainability concerns.

2. Retrofitting the SNFR: Assuming that the case study yields fruitful information regarding sustainability considerations, we can now feed these back into the previous SNFR framework described here. However, we can now instantiate what these quality attributes should look like, based on both what clients want and what requirements engineers understand. Ideally, stages one and two would be a continuing process with the framework being refined constantly. It would be important to keep track of refinement process to avoid over-specification.

   The issue that we are particularly interested in with this process would be to see how specific of a sustainability NFR framework we can obtain. Would it be more useful to have a general model instead, or even just leave it as a one liner in the SRS standard as described earlier? Unfortunately, we can only speculate about these issues and the only way to resolve them is to perform case studies and refine the framework until we see significant changes one way or the other.

3. Widgets and Standards: Next, reusable sustainability widgets would need to be built, and the process of officially developing an IEEE standard would need to be instigated. We propose these are conducted concurrently as it would engage writers, designers, and builders in widget construction, as well as the development of metrics, standards and methods.

4. Adoption of Sustainability: The recommendation to write an IEEE Recommended Practice for Sustainability Quality Attributes aimed to increase specificity as well as visibility. The widgets aimed to increase support and visibility. Case studies and actively performing the process of eliciting, realizing and following through with sustainability requirements is probably the best way of trying to encourage the adoption of sustainability as a software quality. While there is no guarantee that engaging in such activities will result in successful adoption, we believe that the roadmap laid out in this document in conjunction with the Sustainability NFR framework is the way forward to tightening...
the relationship between sustainability and software engineering through better requirements engineering.

8. CONCLUSION

In this paper we have described a Sustainability Non-Functional Requirements framework that can serve as the basis for requirements engineers to elicit and classify sustainability requirements for the software systems to be built. In addition, we describe how Sustainability can be integrated as a Software Quality into IEEE standards. While we believe that creating theoretical frameworks for sustainability is an achievable goal, a major challenge will be encouraging the adoption of sustainability in requirements and software engineering. To this end, we describe three criteria that have an impact on how successful other software qualities are—specificity, visibility, and support—and by providing means for each of them, sustainability can successfully be integrated as a major software quality.

The software that we build is an integral part of current industrial society; it is critical to be aware of the range of impacts that software has, and seek to build systems that improve the world around us. By developing sustainability as an important non-functional requirement, and then including it as a software quality in an IEEE standard, we hope to lower the bar of entry for software engineers interested in engaging in more sustainable practices.

9. ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under Grant No. 0644415 and DFG EnviroSiSE project under Grant No. PE2044/1-1. We would like to thank participants at the “1st International Spring School on Systems Engineering” for valuable feedback.

10. REFERENCES


14 Safety, Security, ... now Sustainability!
Safety, Security, now Sustainability: the Non-Functional Requirement for the 21st Century

Birgit Penzenstadler, Ankita Raturi, Debra Richardson, Bill Tomlinson
University of California, Irvine
{bpenzens|araturi|djr|wmt}@uci.edu

Abstract

Many software systems today control large-scale socio-technical systems. These systems are not only entangled with the environment, but also with our dwindling resources and mostly unsustainable way of living, while the planet’s population continues to grow.

Dealing with sustainability requirements and supporting their elicitation, analysis, and realization systematically has yet to be solved. Decades ago, the discipline of software engineering faced and solved similar shortcomings in its processes by including safety and security as new qualities for software systems in response to the rising importance of these issues. In light of the increasing consequences of inadequately addressing sustainability in developing software systems, we need to apply the lessons learned from these prior research efforts to this new context and identify the necessary research agenda.

Considering sustainability in software engineering means more than considering energy efficiency and green IT, which are concerned with the 1st order impacts of software systems. Rather, we must also take into account the 2nd and 3rd order impacts in the system context, even if they are hard to assess. By doing so, we as software engineers have the potential to considerably improve civilization’s sustainability.

Index Terms
requirements engineering; sustainability; green software; software, security; non-functional requirements

I. WHY SUSTAINABILITY NOW?

In the last decades of the 20th century, software became deeply ingrained in a wide range of human activities. The growing ubiquity of complex software systems led to both software security and software safety failures [3], [8], and in due course to the need for software engineering researchers and practitioners to address both security and safety during the development of large-scale software systems. In particular, security and safety were added to the set of non-functional requirements, whereas taxonomies of non-functional requirements at that time focused on other qualities such as efficiency, reliability and usability.

For example, in the IEEE standard 730-1989, it is stated that the development of “critical software—that is, where failure could impact safety or cause large financial or social losses” shall be supported, but neither safety nor security are mentioned explicitly as qualities within said standard. In ISO9126:2001 on software quality, security is listed, but safety is not. Finally, in ISO25010:2011, safety is included as an explicit characteristic.

In the 21st century, software systems are central to the operation of most sectors of industrial society. As such, they are embedded in, and enable, the rather unsustainable way in which most of us live. It is not civilization’s intention to harm the Earth, but the collective sum of our individual actions, which often favor local convenience over global responsibility, plus the effects of the societal structures we have created to live together lead to negative impacts on our environment. If our civilization is to transition to sustainability, many sectors of society will need to rethink their mode of operation. The ubiquity of software offers a unique opportunity: a shared point of intervention across a wide range of intertwined and impactful industries. To support the transition to sustainability, environmental sustainability must be explicitly considered as a non-functional requirement in the software engineering process.

Software engineering has considerable potential to support “greening through IT” — that is, making civilizations more environmentally sustainable via IT interventions [15]. To draw attention to such issues in software engineering, we argue that sustainability must be treated as a first-class quality along side other critical attributes such as safety, security, efficiency, reliability, and usability.

Socio-technical IT systems are among the most powerful tools humanity has ever created. Understanding how to engineer the requirements of such systems so that they better enable social wellbeing and sustainability could have a constructive benefit to the world. Instead of merely optimizing current systems, software engineers must embrace transition engineering — an emerging discipline that enables change from existing unsustainable systems to more sustainable ones by adaptation and filtering of demand to declining supply [6].

II. WHY SUSTAINABILITY MATTERS FOR SOFTWARE ENGINEERING

The United Nations Earth Summit of 1992 led to the establishment of the UN Framework Conventions on Climate Change and Biological Diversity, ratified by 195 countries, but applied by only few of them. The UN Rio+20 Conference, which took
place 20 years later, resulted in recommendations that prescribed an objective and were limited in their practical use; they only prescribed an objective: To preserve our world for future generations we have to change consumer behavior with regard to environmental impact; action is currently limited compared to our other economic activities. Psychologist Geoffrey Beattie’s answer to “Why we aren’t saving the planet yet” [1] is that the problem is too complex for a single mind to solve, and therefore we shy away from becoming active. Sustainably-engineered software systems could help in this endeavor, as we may be able to facilitate a more sustainable lifestyle by directly influencing the surrounding system context (including the people, the infrastructure and the environment). Furthermore, software systems have such a significant impact on our everyday lives that changes in these systems towards more environmental sustainability can ripple to other systems with which they interact, and affect other industries involved. This impact can be direct, indirect, or occur as rebound effect. This potential for positive change is why, as software engineers, we should about supporting sustainability in our work. Please see the sidebar for a characterization of sustainability.

Sustainability science is currently discussing whether sustainability is limited by ecological constraints or enabled by continuous innovation and transformation [9]. The ecological constraints are explained by Burger et al. [2] as a flow of resources from environment to society that has to conform to physical laws and balance. This is reflected as information flows in software systems whenever we represent real world objects with natural resources, e.g. in a production site. The optimistic view that we can sufficiently adapt human societies to overcome resource limitation by innovatively transforming our systems is reflected by Matthews and Boltz [10]. As software engineers, we have the opportunity to take up the challenge of integrating sustainability into the systems we build. The ubiquity of software systems in industrial civilization means that the reach of this effort could be quite broad indeed.

Both the ecological pessimism and the technological optimism are also present in the overview of the field of “Sustainability and ICT” by Hilty et al. [5], who differentiate Environmental Informatics, Green (in and by) IT/ICT, and Sustainable Human-Computer Interaction. Their bottom line analysis is that technological efficiency alone will not produce sustainability, rather that efficiency in combination with sufficiency will support innovation for sustainability.

How can we tackle the goal of effectively supporting sustainability through software development? It will take a systematic approach that ranges from incorporating sustainability requirements to performing quality assurance to ensure these requirements are adequately realized, all of which should be guided by an explicit sustainability quality characteristic. Similar arguments apply, and an additional dimension, supports better structuring of concerns with respect to software systems.

Considering sustainability with regard to software engineering is twofold: One can consider improving the sustainability of the process of software development. However, this will have limited impact when compared to the overall impact that software systems affect in our daily lives. Instead, improving the sustainability of software systems in their surrounding context allows software engineering to have a more significant impact on improving the sustainability of human society.

With regard to the impact that a software system can cause in its context, we distinguish three orders of magnitude [10]: First order impacts are direct effects of a software system on its environment, for example energy usage, e-waste production, emissions caused by required infrastructure, etc. Second order impacts are indirect effects or induction effects caused by software systems, e.g., changes in the users’ resource consumption or consumer behavior. Third order impacts are rebound effects, for example the increased efficiency of systems tends to make us use even more systems which, in total, consume even more energy. All of these have to be considered within an encompassing approach to supporting sustainability. For designing these means of support, there is a variety of standards in place and / or under development [11, 12, 13, 14, 15].


Sidebar section

Characterization of Sustainability

In general, sustainability is the “capacity to endure”, but interpreting this requires context. A popular definition of sustainable development was given by the UN as “meeting the needs of the present without compromising the ability of future generations to meet their own needs” [1], and characterized by the three dimensions economic, social, and environmental. This characterization is extended with a fourth dimension human by Goodland [2], portraying individual development by every human over their life time. When analyzing the sustainability of IT systems, these four dimensions apply, and an additional dimension, technical, supports better structuring of concerns with respect to software systems. More elaborate theoretic frameworks on the definition of sustainability exist, e.g. [3, 4, 5, 6, 7, 8], all relying on systems thinking [9] to some extent. For the remainder of this article, we focus on environmental sustainability as the other aspects are, in part or implicitly, already supported by established software engineering practices.

Considering sustainability with regard to software engineering is twofold: One can consider improving the sustainability of the process of software development. However, this will have limited impact when compared to the overall impact that software systems affect in our daily life. Instead, improving the sustainability of software systems in their surrounding context allows software engineering to have a more significant impact on improving the sustainability of human society.

With regard to the impact that a software system can cause in its context, we distinguish three orders of magnitude [10]: First order impacts are direct effects of a software system on its environment, for example energy usage, e-waste production, emissions caused by required infrastructure, etc. Second order impacts are indirect effects or induction effects caused by software systems, e.g., changes in the users’ resource consumption or consumer behavior. Third order impacts are rebound effects, for example the increased efficiency of systems tends to make us use even more systems which, in total, consume even more energy. All of these have to be considered within an encompassing approach to supporting sustainability. For designing these means of support, there is a variety of standards in place and / or under development [11, 12, 13, 14, 15].


### III. History of Safety and Security

Safety and security were included systematically into software engineering due to serious accidents and other negative effects of software systems:

1) Software engineers started to consider safety issues in their processes [7], [8], particularly hazard and fault analysis, following a series of failures in which users were hurt and/or systems were damaged.

2) Software engineers began working on security algorithms and mechanisms [3] after security threats led to widespread identity theft and financial fraud, among other issues.

In both cases, however, it took a while until these characteristics were included in requirements engineering standards. In the IEEE830, safety and security are only mentioned as types of constraints in all three editions (1984, 1994, and 1998). Finally, in the most recent RE standard—the ISO 29148:2011—both security and safety are central topics. In the graphical timeline of Figure 1, we illustrate the progression of the most important phases in safety and security development with respect to requirements and process; these phases are abstracted from [3], [8].

**Safety:** A safety requirement is a constraint derived from identified hazards. Nearly all the serious accidents in which software has been involved in the past twenty years can be traced to requirements flaws, not coding errors [7]. The reason is that the requirements usually reflect incomplete or wrong assumptions about the operations carried out or about the context. Extensive investigation into specification and analysis of requirements for safety-critical systems began in the 90s, especially in the area of formal methods [8].

**Security:** A security requirement is a “manifestation of a high-level organizational policy into the detailed requirements of a specific system” [3, p.227]. Although security models have been around since the 1970s, until recently elaborating security requirements had been added as an afterthought and often a compromise to satisfy compliance to standards. Early integration with systems engineering promotes that a model of domain ontology is first constructed, and this drives the rest of the requirements process [3].

**Implication:** First, we realize that some aspects of safety and security can be handled by more traditional quality attributes. For example “Keep spurious alarms to a minimum.” [7, p. 300], although safety-related, can be labeled as a performance requirement.

Second, not every requirements engineering process might require safety and/or security, but it is generally accepted that there is a need for these quality characteristics to be explicit in addition to the traditional ones as in the quality models by Boehm, McCall, and the ISO standard 25010 (referenced in the sidebar).

Regardless of both points, safety and security are called out and treated specifically because they are very important characteristics. We argue that the same is true for sustainability, especially the dimension of environmental sustainability, and that we need to find adequate means to analyze, support, verify and validate sustainability requirements in software engineering.
### Software Security Considerations

<table>
<thead>
<tr>
<th>Year</th>
<th>1980</th>
<th>1990</th>
<th>2000</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirements and Process</strong></td>
<td>Push towards unified security and system models</td>
<td>Secure computations, not just secure computers</td>
<td>Push towards formalisms</td>
<td>Verification of systems</td>
</tr>
<tr>
<td><strong>Software Piracy and Protection</strong></td>
<td>Push towards adapted solution and protect</td>
<td>Trusting components</td>
<td>Push for secure software interfaces</td>
<td>Secure development</td>
</tr>
<tr>
<td><strong>Architecture and Design of Secure Systems</strong></td>
<td>Push towards designing for security</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Testing at Different Levels for Reliability and Safety</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Software Safety Considerations

<table>
<thead>
<tr>
<th>Year</th>
<th>1980</th>
<th>1990</th>
<th>2000</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Designing for Safety</strong></td>
<td>Push towards safe-product families and safe reuse</td>
<td>Testing at different levels for reliability and safety</td>
<td>Push for more testing and evaluation of safety-critical systems</td>
<td></td>
</tr>
<tr>
<td><strong>Certification and Standards</strong></td>
<td>Push for more cooperation with related fields</td>
<td>Resources available on software safety</td>
<td>Push for more software engineering education on safety</td>
<td></td>
</tr>
<tr>
<td><strong>Safety Requirements Specification and Analysis</strong></td>
<td>Push towards integrating formal and informal methods</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hazard Analysis</strong></td>
<td>Push for autonomous runtime monitoring</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Software Sustainability Considerations

<table>
<thead>
<tr>
<th>Year</th>
<th>1980</th>
<th>1990</th>
<th>2000</th>
<th>2010</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Characterization of Software Sustainability</strong></td>
<td>Characterizing software sustainability</td>
<td>Sustainability dimensions: environmental, economic, social, human, technical</td>
<td>Considering the orders of magnitude of the impact of software systems</td>
<td></td>
</tr>
<tr>
<td><strong>Mythbusting of Software Sustainability Misconceptions</strong></td>
<td>Mythbusting of software sustainability misconceptions</td>
<td>Interfacing to problem driven engineering</td>
<td>Taking responsibility and establishing a sustainability culture</td>
<td></td>
</tr>
<tr>
<td><strong>Sustainability Requirements Analysis</strong></td>
<td>Specifying objectives and goals</td>
<td>Performing LCA</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sustainability Policies and Standards</strong></td>
<td>Identifying sustainability standards</td>
<td>Developing sustainability standards</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Metrics for Sustainability</strong></td>
<td>Development of Assessment Techniques</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 1.** History of safety, security, and sustainability considerations in software.

### IV. Comparing Sustainability to Safety & Security

When taking a look at safety and security, there are quite a few concepts that might be helpful when transferred to sustainability: a) there are similarities in characterizing them; b) there are similar myths and misconceptions; c) we need adapted methods to analyze these requirements and assure quality; and d) there exist policies and standards that impose constraints but also support their realization.

These four aspects are explored briefly in the following subsections. An overview of past considerations of sustainability as well as present considerations and work in progress of sustainability in software systems is depicted in Figure 1.

#### A. Characterization

Safety is “an emergent property that arises when the system components interact within an environment” [7, p. 67]. An emergent property means a characteristic that is only exhibited by the system in use, as opposed to an inherent characteristic already exhibited in the system’s static existence. In other words, a software system is not unsafe in itself, but it can be unsafe in its application environment [7, p. 57].

In contrast, sustainability is not completely an emergent property, but must be dealt with in two parts: there is an inherent part that shows up in the 1st order and some 2nd order effects, and there is the emergent part that manifests itself in 2nd and 3rd order effects.
Security is the degree of resistance to, or protection from, harm to information (e.g., unauthorized access). A security policy is the set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information. A similar mechanism of policies and requirements is needed for supporting and enforcing environmental sustainability.

An additional challenge in characterizing sustainability is posed by the fact that measuring first order effects is easier than measuring second order effects (as they occur only in combination with human behavior) or, even more difficult, third order effects, which require measuring systemic impacts.

B. Myth Busting

The myths and misconceptions where parallels can be drawn between safety and security on one hand and sustainability on the other on include goal conflicts, isolation, solution-driven engineering, and culture.

**Goal Conflict:** “A classic myth is that safety conflicts with achieving other goals and that tradeoffs are necessary to prevent losses. In fact, this belief is totally wrong. Safety is a prerequisite for achieving most organizational goals, including profits and continued existence. [...] The ‘conflict’ myth arises because of a misunderstanding about how safety is achieved and the long-term consequences of operating under conditions of high risk.” [7, p.416]

Similarly, there is a misperception that sustainability will conflict with a company’s economic goals. However, taking a practical view on the actual economics of how we deal with environmental impacts caused by companies, it becomes clear that these goal conflicts only appear because companies do not have to pay for the emissions they produce during manufacturing, nor are they responsible for the disposal of a product at the end of its life cycle. If, on the other hand, companies had to include these expenses, sustainability would form a natural part of their economic bottom line. Current efforts in emission trading among companies and disposal/recycling fees for consumers are a first step in this direction although its contribution to sustainability is questionable because planting trees does not undo caused harm — it only tries to offset its effects.

**Isolation:** A second common misperception is that safety, security, and sustainability can be dealt with in isolation of other quality characteristics. In isolation, requirements are replaced by solution-specific constraints, or the scope is simply set too small so that not all related aspects are taken into account and the result is a suboptimal solution. For example, security requirements are often (mis-)treated by replacing them with security-specific architectural constraints that may hinder the security team in using the most appropriate security mechanism to deal with an actual security requirement [4, p.53]. Security and safety engineers by now are aware that they have to look at the “real” requirements instead of simply replacing them by solution elements; the same applies to sustainability requirements.

**Solution-driven Engineering:** Security requirements are often solution mechanisms in disguise [4]—“The user logs into the system with his security code.” is not a good security requirement. This leads to suboptimal solutions as a solution is already prescribed instead of analyzing the problem space and separating the solution finding. The same holds for sustainability requirements—e.g. when energy-saving mechanisms are applied instead of questioning and analyzing the process that the mechanism is required for. It is necessary to explore the problem space and the application domain in depth to find an optimal solution.

**Culture of Blame and Denial:** Establishing a safety culture with responsibility prevents the tendency of putting blame on someone only after safety incidents ("Safety, Culture, and Blame" [7, p.426ff]) and one could imagine the same with security breaches. In contrast, hardly anyone accepts responsibility for environmental sustainability or damage, so we are talking about **Sustainability, Culture, and Responsibility.** We have to develop a culture of responsibility for our planet that includes everybody, and where the still-present and strongly counter-productive denial of sustainability issues ceases to exist because the arguments for sustainability and the indicators of various collapsing systems are visible already.

C. Requirements Analysis

If sustainability is a quality attribute, then we need adequate analysis and quality assurance techniques as for safety and security. For analyzing sustainability requirements, we can make use of parallels in the definition of objectives and goals as well as risk analysis techniques.

**Objectives and Goals:** In safety, System-Theoretic Process Analysis (STPA) uses intent specifications that contain the safety-related goals and objectives for the system under consideration [7, p.311-312].

For sustainability goals and objectives, we have developed a generic reference model that allows engineers to structure sustainability goals according to the five dimensions and to detail them further with requirements and constraints or supporting activities [12], [13]. An overview is available at http://www.se4s.org. This allows engineers to link the environmental sustainability goals more strongly to other sustainability goals. Furthermore, as in safety goal specifications, the systematic decomposition and derivation of requirements allows for traceability and validation of objectives during quality assurance.

**Risk Analysis / LCA:** In safety, three hazard analysis techniques are widely used for risk analysis: Fault Tree Analysis, Event Tree Analysis, and Hazards and Operability Analysis.

A common technique used in sustainability science is life cycle assessment (LCA, see [2]), which is based on systems thinking. The insights from LCA might benefit developing sustainable software systems as LCA helps identifying risks and hazards, which provides the basis for defining sustainability requirements and constraints.
Misuse cases are used for security requirements elicitation by various researchers [14], [11]. This technique can be used in addition to LCA to identify sustainability risks that might otherwise be overlooked during the analysis. These identified risks must be dealt with by specifying constraints that when met avoid threatening conditions.

D. Policies and Standards

Specifying constraints is a common practice to incorporate regulations from policies and standards into requirements specifications in safety and security. Constraints assure that these regulations are adhered to during quality assurance. Constraints are derived from the identified hazards as a way to avoid accidents. A method to deal with risks and hazards in sustainability management would also be to derive constraints from regulatory sources.

Aiming for standardization of environmental sustainability to support specifying constraints, the next step is to extend software engineering standards. For example, the ISO 29148 on requirements engineering could explicitly include a section on sustainability in the software requirements specification document template.

To ensure that such regulations and standards have the desired effects, however, it may be necessary to trigger institutional change so that environmental regulations consider more than 1st order effects. Institutional change cannot be solved within requirements engineering, but it may be the appropriate trigger for systemic improvements.

V. Outlook

We have described a number of areas in software engineering that can learn from safety and security to address sustainability in an effective way; each of these areas require extended further research. In particular, this paper has explored sustainability as a non-functional requirement along with consequent requirements analysis and standards. Quality assurance techniques corresponding to the requirements are also required, which necessitates future research in establishing sustainability metrics as well as assessment techniques.

Thorough research requires evaluating various methods through case studies in a variety of application domains to develop guidance on the most appropriate, adapted methods in the context of supporting environmental sustainability as it pertains to software systems.

Metrics

For both standards and quality assurance, we need to define a set of metrics for the different dimensions of sustainability by relying on the respective sets of metrics available:

- The ISO 14000 family for environmental sustainability;
- The ISO 26000 for social sustainability;
- The Environmental Sustainability Index ESI.

Steps in that direction are already being taken with the IEEE 1680 family of standards for environmental assessment of IT. Furthermore, there is a need for standards that give guidance for development, as is done through the IEC 61508 on the functional safety of electronic safety-related systems.

Assessment Techniques

For assessment techniques to address quality assurance, we propose to:

- Evaluate and adapt LCA to software engineering;
- Make use of environmental impact assessment in software engineering.

VI. Conclusion

Sustainability in software engineering encompasses not only energy efficiency and green IT, but must also consider the 2nd and 3rd order impacts of software systems. This paper argues that to do so, sustainability must be considered as a first-class quality attribute and specified as a non-functional requirement of IT systems.

Alternative Solutions

Considering the five dimensions of sustainability – human, social, environmental, economic, and technical – all but one can be supported to some extent by current software engineering methods as these concerns are not new per se. Missing from consideration is environmental sustainability. While few people deliberately intend to harm the environment, it is often more convenient to take “the easy way out”, failing to explore environmentally friendlier options, given that we already have all the advanced technology and facilities that science and economy have brought us. This seems to be true in current software engineering methods as well.
Feasibility

The challenge of incorporating this conflict into software engineering is an issue of requirements prioritization, which is usually handled by negotiation between system stakeholders. Perhaps the solution requires an explicit stakeholder for environmental sustainability. Another way to ensure prioritizing environmental sustainability is to enforce policies based on the compliance-driven economy. Although we can hope that every citizen takes responsibility for their impact on the planet, we know it will be a long time before this comes to fruition.

Take-away Message

If sustainability policies and standards are put in place, and software engineers prioritize them in the systems they develop, future technology may significantly contribute indirectly to influencing the behavior of users who interact with those systems as well as directly to saving the planet.

ACKNOWLEDGMENTS

This work is part of the DFG EnviroSiSE project (grant number PE2044/1-1) and was funded in part by the National Science Foundation Grant No. 0644415.

REFERENCES


AUTHOR BIOGRAPHIES

Birgit Penzenstadler is a postdoctoral researcher at the University of California, Irvine (UCI), USA, with a doctoral degree from Technische Universität München (TUM), Germany. Her research currently focuses on supporting sustainability from within requirements engineering and quality assurance. She is a member of IEEE.

Ankita Raturi is a PhD student in Software Engineering at UCI with a Master’s degree from UCI. Her research interests are involving sustainability considerations in software development, software standards and metrics.

Debra Richardson is a Professor of Informatics at UCI with a PhD from University of MassachusettsAmherst in 1981. Her current research interests are in exploring how software engineering can be made to address socially relevant problems such as sustainability.
Bill Tomlinson is a Professor of Informatics at UCI with a PhD from MIT in 2002. He studies the fields of environmental informatics, human-computer interaction, multi-agent systems and computer-supported learning.
15 Framing Sustainability as a Software Quality Property
Framing Sustainability as a Software Quality Property

Patricia Lago
Department of Computer Science
VU University Amsterdam
The Netherlands
p.lago@vu.nl

Sedef Akinli Kocak
Data Science Laboratory, Environmental Applied Science
Ryerson University, Canada
sedef.akinlikocak@ryerson.ca

Ivica Crnkovic
Department Computer Science and Engineering
Chalmers University of Technology
Sweden
crnkovic@chalmers.se

Birgit Penzenstadler
Institute for Software Research
University of California, Irvine
USA
bpenzens@uci.edu

ABSTRACT

Sustainability is an increasingly important software quality concern. In this paper we introduce a framework for classifying a software sustainability in terms of four dimensions: economic, social, environmental and technical. We illustrate our initial observations and new research avenues toward sustainable software with two case-study examples. Thanks to its simplicity, the framework provides practitioners with a holistic view of the relevant sustainability-related software qualities and how they influence each other. This approach can facilitate confident decision-making during the software development process.

1. INTRODUCTION

Sustainability is defined as the ‘capacity to endure’ [11] and to ‘preserve the function of a system over an extended period of time’ [5]. Consequently, discussing sustainability requires referring to a concrete system—such as a specific software system. In order to analyze the sustainability of a specific software system we must consider four major dimensions—economic, social, environmental, and technical (cf. Section 2)—which entail in their interrelationship various trade-offs [10].

The first three dimensions stem from the Brundtland report [12], whereas technical is added for an adequate discussion of software-intensive systems [7] at a lower level of abstraction. The economic dimension is concerned with preserving capital and value. The social dimension is about maintaining societal communities. The environmental dimension seeks to improve human welfare by protecting natural resources. The technical dimension intends to support long-term usage and evolution of software-intensive systems. Sustainability can only be achieved when all dimensions are taken into account. Moreover, including the environmental dimension makes it possible to aim at dematerializing production and consumption processes in order to save natural resources [3]. The connections between these dimensions involve different dependency types and stakeholders [8, 9]. Potential conflicts among the stakeholders’ concerns make it crucial to analyze the relationships between goals in the four dimensions.

The shortcoming of current software engineering practice with regard to sustainability is that the technical and economic dimensions are taken into account while the environmental and social dimensions are neglected. The question we address in this article is how these concepts relate to software and how to break down the respective concerns into software quality requirements. In particular, we focus on the (currently neglected) environmental dimension and its relation to the other dimensions. While most efforts in environmental sustainability in software to date have focused on energy efficiency, we also tie this concept to other sustainability aspects, particularly to address second order effects [5]. Second order effects denote the effects of a software system in use in its operational context, for example the effects of a car sharing system used by many users over a number of years on the surrounding environment.

Our contribution is a sustainability analysis framework that aids practitioners in exploring the spectrum of software qualities related to the four dimensions of sustainability and explicitly represent dependencies between dimensions. To illustrate the application of this framework we present two case-study examples from different domains.

2. SUSTAINABILITY ANALYSIS FRAMEWORK

Our framework aims to capture the relevant qualities that characterize the sustainability concerns of software systems, and help identify how these qualities influence each other with respect to the different aspects of sustainability. Softwar qualities as non-functional properties have been extensively studied and adopted in software engineering. In particular, in software architecture various methods for quality evaluation/assessment have been defined to support holistic...
box: Background on Software Sustainability and Its Qualities

Software Engineering and Sustainability

In the past few years a transformation in the role of IT in sustainability occurred due to rising energy demands with increasing adoption and usage of IT systems and growing negative impacts on the environment. As outlined by Gartner [25], industry is moving toward sustainability in order to enhance compliance, operational efficiency and performance. The report suggests that achieving sustainability objectives requires considering the integration of IT, which provides efficiency, performance and business processes. While industries are integrating many IT-enabled solutions, they are also having challenges to integrate sustainability programs, such as lack of awareness on the emissions reduction and economic savings that IT offers, lack of robust policies to address climate change, and lack of frameworks, systems, tools and practices, which provide decision support and connect sustainability performance to economic performance [6].

As IT industry becomes aware of sustainability, the software engineering research community has recently been paying attention to this development, as demonstrated by an increased number of publications, empirical studies, and conferences. Surveys [18, 20] show that over 50% of the studies on sustainability in software engineering were published between 2010 and 2012 indicating the emergence of the sustainability topic in the community. Software technology enables systems to increase energy efficiency, streamline processes, or adapt to changes in the environment. There is a rich body-of-knowledge regarding energy estimation [9] and optimization (e.g., energy-efficient algorithms) as well as tools and methods to measure energy efficiency [10, 14] particularly for mobile devices [5].

Researchers often rely on estimates or focus on hardware rather than software. There is a growing attention on considering energy efficiency as an objective of software development life cycle and related development tools and methodologies. Recently, a practical evaluation model that could serve as a method to evaluate the energy efficiency of software applications has been developed by Kalaitzoglou et al. [11]. All of those energy related studies focus on the environmental dimension of sustainability.

The other dimensions of sustainability, as related to software, have also been actively discussed. For example, Tate [24] characterizes sustainable software engineering as “the ability to react rapidly to any change in the business or technical environment” but only considers economic aspects. Mahaux et al. [15] analysed the usage processes of a software system with respect to social and environmental aspects. Naumann et al. [17] identified a lack of models and descriptions covering the entire spectrum of software sustainability aspects. Razavian et al. [22] applied the four dimensional sustainability model onto the e-services and mentioned the conflicts between dimensions. However, their dimensional applications on the other industries as well as conflicts and trade-offs across the dimensions have not yet been discussed. More concrete initiatives are emerging in industrial practice [7].

All the related studies certainly help to build awareness of sustainability in software engineering. The next step is to create best practices and guidance by applying definitions, frameworks and models to case studies. In this direction, our framework provides the necessary mean for software sustainability by using all four dimensions, economic, social, environmental, and technical. Further, our case-study examples may lead to overcome the challenges on sustainability practices in software engineering.

Software Quality and Sustainability

Various systems (e.g., energy, management, and computer systems) have targeted sustainability as a quality objective. Models, tools, and metrics/indicators have been developed to instrument systems for sustainability assessment. A survey on green software metrics [8] revealed that metrics are strictly related to energy consumption and models to assess green software qualities are lacking. Mocigemba [16] defined a sustainable computing model focussing on product, production and consumption-process assessments, for both hardware and software. Recently, Afgh [1] introduced a multi-criteria assessment method, with economic, environmental and social indicators, as a tool for the quality assessment of the energy system as proxy for sustainable development. Other preliminary initiatives investigate how to define, measure, and assess sustainability as a software quality attribute [19, 2, 13]. In general, these efforts point to the natural multi-dimensionality of sustainability and the need for an interdisciplinary approach.

The quality models introduced by ISO (i.e., ISO/9126 and ISO/IEC 25010) do not (yet) consider sustainability as a quality aspect. However, WG42 (working on ISO/IEC 42030) is considering including sustainability evaluation at the software architecture level. Architecture assessment methods have matured to aid the design of software-intensive systems that must support system properties depending on software and hardware, users and other systems. While software architectures provide a holistic perspective on the whole system, architecture assessment methods so far do not yet address sustainability as a quality objective.
Kern et al. [12] developed a quality model for green software that refers to quality factors from ISO/IEC 25000 based on direct and indirect software-related criteria. Calero et al. [4], who consider sustainability as a new factor affecting software quality, presented a quality model based on ISO/25010. In the latest study, Akinli Kocak et al.[3] evaluate product quality and environmental criteria within a decision framework and provide a trade-off analysis between the criteria. On the other hand, previous studies discuss the relations between software quality aspects and sustainability and highlight that both product and use qualities need to be considered when assessing software sustainability. On the other hand, no study has specifically investigated the multi-dimensionality of sustainability, their dependencies and the trade-off between the dimensions in software engineering practice. Sustainability analysis frameworks are starting to appear in software engineering research [21, 23]. Our work is a first attempt to emphasise on the environmental dimension, which is currently neglected in the other studies.

reasoning and decision-making that embrace software, hardware, human, and system elements. We exploited this holistic approach and defined our framework by extending an existing model—the Third Working Draft of ISO/IEC 42030 Architecture Evaluation [4] as depicted in Fig. 1. Gray boxes denote generalized pre-existing components from the working draft. While this draft specifically targets evaluations, the potential context of our framework is broader, embracing any activity that relies on a sound representation of qualities, including requirements engineering, design-decision making, trade-off analyses, and quality assessments. Here we do not focus on any specific context, but rather illustrate how the framework can be used to frame sustainability quality requirements and concerns.

The following describes the dimensions used in our framework to characterize sustainability in the context of software intensive systems.

**Social Sustainability** focuses on supporting current and future generations to have the same or greater access to social resources by pursuing generational equity. For software-intensive systems, this dimension encompasses the direct support of social communities in any domain, as well as the support of activities or processes that indirectly create benefits for social communities.

**Environmental Sustainability** aims at improving human welfare while protecting natural resources. For software-intensive systems, this dimension aims at addressing ecologic requirements, including energy efficiency and ecologic awareness creation.

**Technical Sustainability** addresses the long-term use of software-intensive systems and their appropriate evolution in an execution environment that continuously changes.

**Economic Sustainability** focuses on preserving capital and (economic) value.

As shown in Fig. 1, an Evaluation Criterion can be a quality requirement. In particular, as we focus on characterizing the sustainability-related software qualities, we need to address how quality requirements relate to sustainability—Sustainability Quality Requirements. In this context, they may include both traditional quality requirements (e.g., performance, usability, security, or maintainability) and sustainability related ones (e.g., energy efficiency).

Moreover, whenever we specifically target sustainability (cf. Fig. 1 where the association ‘aims at’ links the Evaluation Objective to the Sustainability Dimension), we must perform trade-offs among the various qualities classified as belonging to each of the four dimensions. In particular, we observe that traditional software decision-making considers trade-offs either between different technical sustainability criteria (e.g., performance versus availability), or between technical sustainability criteria and economic sustainability criteria (e.g., performance versus costs). In contrast, sustainability-related software decision-making involves trade-offs between environmental sustainability criteria (e.g., energy efficiency) and social, economic, and technical sustainability criteria.

To frame the software qualities in this context we position them in the four sustainability dimensions, and relate them to the Concerns of the relevant Stakeholders. For the sake of simplicity, this information has not been included in the illustration of the case-study examples. However, the description of Case A refers to three main stakeholders: the Surrounding community & Society at large (concerned about environmental sustainability like forest sustainability), the Customer (concerned about economic sustainability like production savings expressing productivity and economic value creation), and the Producing Organization including managers and engineers (concerned about technical sustainability like optimization of configurability and performance).

Moreover, interdependent quality requirements may influence each other in a positive or negative manner (cf. association/association-class influences among Sustainability Quality Requirements). For example, looking at Case A and Fig. 2, Performance and Energy Saving may influence each other: increasing Performance might demand more resources that in turn will consume more power and ultimately have a negative impact on Energy Saving. Using our framework to make these influences explicit helps designers of
3. EXAMPLES FROM THE TRENCHES

We show the applicability of the sustainability analysis framework using two examples extracted from case studies performed by the authors. For each case-study example, we briefly introduce the domain and then discuss its sustainability qualities and their interdependencies. Moreover, we illustrate the added value of using the framework for practitioners by discussing different aspects: stakeholders (in Case A), and specialized influences relations between qualities (Case B). The granularity of requirements ranges from coarse-grained high-level goals to fine-grained detailed system requirements. Our case-study examples are at the high-level end of this spectrum (cf. van Lamsweerde [6]).

For each case-study example, Figs. 2 and 3 focus on the following framework elements: sustainability quality requirements (for which we detail parameters and metrics to capture quality levels), their influences and inter-dependencies, and the sustainability dimension they belong to (represented as swim-lanes). In these diagrams we are not proposing a new notation, but a framework that presents the approach we suggest for capturing the interrelations among the four sustainability dimensions and trading off between them. For formalizing and modeling in more detail than we could present in this article, the notations proposed in [1] could be used. In this article we use a simple notation based on UML class diagrams.

3.1 Case A: Paper Mill Control System

The paper production industry is an example of successful sustainability improvement through continuous technical and economic solution advances [2]. Thirty years ago, a typical plant control system (PCS) would have a paper production cycle of several days. The energy consumption was very high (though energy was cheaper, the energy costs were three times more per ton of pulp than today) and so was the pollution, mostly in a form of water polluted by chlorine compounds (water pollution at that time just started to be an issue). A PCS would manage the entire process by using and controlling a few hundred sensors and actuators. A typical plant would employ 2-3 thousand people, with a considerable number of low-educated employees, and several tens of experts that would optimize the process in respect to the production quality using their experience. Today, a PCS can handle several hundred thousand signals while reducing the production cycle to an hour and at the same time lowering environmental impact significantly (e.g., the water consumption of 200-300 m³ per ton of pulp in 1970 is decreased to less than 50 m³/ton, in some mills even below 10 m³/ton). The number of employees has decreased more than 75%, but their qualification has increased - today more than 50% of employees are highly qualified. The production has increased dramatically - at least 10 times. Nowadays, the main concern is energy savings, including energy spent in the technological process (e.g., cooking paper pulp) and energy used by the PCS. This gives environmentally sustainable software a double role: a) to decrease the energy consumption of the PCS itself (which is distributed and complex with many devices), and b) to decrease the energy consumption of the entire production system by choosing smart algorithms and energy-efficient technologies controlled by software. Consequently, the survival of the companies was dependent on all four sustainability dimensions, primarily driven by customers and by competitors, but also by the local society.

Figure 2 (Case A) shows example sustainability quality requirements, sorted per dimension, and some of the relations among them. We distinguish between vertical (within a dimension) and horizontal (between dimensions) relations. Social dimension refers to the changes in the infrastructure in the companies and in the society needed to support the requirements on employees’ skills. A company would need highly educated people, which poses demands on their supply from society. The company would need to make a short and long-term plan for re-qualification of the employees, and society (typically a municipality or county) would take responsibility for retraining people. Increase in education level would improve the environmental sustainability awareness. This is an example of horizontal relations. An example of a vertical relation in the environmental dimension is the following: The companies deploy new technologies that lead to lower water pollution, and higher effectiveness of the process, which leads to increased environment sustainability (cleaner water, energy savings, forest savings and forest regenerations). That would, however, require a wise tradeoff between increased effective production (expressed by scalability, performance, and configurability), and economic and environmental requirements. For example, increased productivity is a threat for the environmental demands, and that requires new technologies as well as changes in the process - direct and indirect changes, like selective tree cutting, paper recycling, and planting new trees, which in their turn require changes in the technology of the control system.

The horizontal relations also show a balancing of the stakeholders’ interests; trade-off decisions are typically required between economic and social sustainability requirements, or between economic and environmental sustainability requirements. Technical requirements, in contrast, provide the solutions that improve economic and environmental characteristics.

This case example illustrates how the sustainability analysis framework can be applied in the development processes of large, long-lived systems that require public investment and feature significant profit margins. The economic and technical sustainability are customer-driven. The environmental and social sustainability requirements do not come from the customers, but rather from the surrounding community and the society at large (i.e., region and state). Due to the large public investment in such an industry, the society can impose requirements. Since the environmental and social sustainability requirements do not come from the customers, they tend to be overlooked by managers and engineers. We argue that integration of our four-dimensional sustainability analysis framework into the engineering processes of such long-lived industrial systems provides a valuable support to
Figure 2: Sustainability quality requirements: Paper Mill Control System
managers and engineers to satisfy not only economic and technical but also environmental and social sustainability requirements. In the case of the paper production industry the trade-offs and the innovative solutions have led to a success in all sustainability dimensions.

3.2 Case B: Car-Sharing Platform

In a recent study, we analyzed the sustainability impact of the Munich-based car sharing platform DriveNow [7]. DriveNow is created to serve customers who do not possess a car, in short-distance inner-city trips. Fig. 3 depicts the aspects contributing to both the environment and DriveNow’s business success. The primary quality requirement is High usage of the car sharing platform in the Economic Sustainability Dimension. It is supported by a Well-designed application, which in turn supports (in the Social Sustainability Dimension) a High public acceptance of application.

The focus of this case lays on the different types of influences framework relations. As with any kind of requirements or goals, sustainability can be linked through various types of influences relationships (cf. goal relations [6]). We focus here on supports and conflicts. The following discusses one requirement and its interrelations.

This example illustrates the problems arising from direct and indirect effects on quality requirements. Environmental Sustainability, in terms of Energy savings, is affected in at least three ways:

1. For a well-designed application, reliable GPS functionality is needed, and adding this will, in turn, negatively affect energy savings in the application.
2. DriveNow aims at making people share cars. This leads to less car production, hence positively affecting energy savings in production.
3. Moreover, DriveNow can generate revenue not only through the platform itself, but also through the marketing value created by driving brand-new cars through the city. These cars will be seen by potential customers, who may be triggered to buy them, which in turn leads to more emissions and less energy savings due to increased car production.

This is a well-known phenomenon referred to as first-, second- and third-order effects [5]. While the app use leads to more energy consumption due to GPS use (i.e., first-order effect in (1), the direct effect of a software system), it also facilitates sharing more cars and, hence, reduces the total energy usage (i.e., second-order effect in (2), the indirect effects triggered by the usage of a software system in its operational context). On an even larger scale, the effect might turn around yet again and lead to a completely different end (i.e., the third-order effect in (3), systemic effects triggered by the long-term and wide-spread usage).

The original development of the car sharing system did not consider all dimensions. The primary dimension was economic, and the secondary one was the technical dimension. Both the social and the environmental one were not taken into consideration, and there are a number of consequences from that:

- Social: When the service had been available over a few months and was analyzed, it turned out that a user community had developed where individual users had established an understanding of themselves as being part of a larger community supporting the idea of shared mobility services. Had there been a consideration of the social aspect upfront, the user interface of the system could have been developed so that it was easier to create car pools amongst users.
- Environmental: The DriveNow service mostly uses environmentally friendly hybrid- and electric cars, which provides a good basis for environmental sustainability. However, as the environmental aspect of the service was not considered during the initial business case analysis, no Green IT options were explored for the server side. Similarly, there was no comparative (simulation) study on how the long-term widespread usage of the service would affect city traffic and parking situation. Consequently, the environmental sustainability of the system can still be improved.
- Interrelation of dimensions: One example for the often underestimated relations between the dimensions, which our framework helps to analyze, is the usage of electric cars. Those cars have to be driven in the right way to actually cause less pollution (environmental aspect), so there is a need to offer training (social aspect) for that type of driving, which includes or leads to further investments (economic aspect).

While still simplified, this case illustrates the importance of understanding the interdependencies among qualities. Our framework can be used by practitioners to understand and address them, hence avoiding dangerous pitfalls.

4. OBSERVATIONS

Our case-study examples illustrate how the sustainability analysis framework links the four sustainability dimensions that are seemingly unrelated to software qualities. Determining and analyzing the relations among the qualities as depicted in Figs. 2 and 3 provides decision makers with a blueprint to analyze sustainability qualities and a way to gain insights toward sustainability stewardship. By framing all sustainability dimensions, the framework enables to make assessments and trade-offs across different dimensions. For example, in Case A, by using the framework, one can easily identify not only technical and environmental, but also social and economic trade-offs. The framework also helps to capture the influence of different stakeholders on the various qualities regarding the four dimensions. Both case-study examples showed that sustainability quality relations carry positive or negative influences. Moreover, they revealed that when making sustainability quality evaluation of the system, all the aspects have to be taken into consideration. For example, in Case B, environmental and social dimensions were originally not under consideration, thereby hindering potential positive impact on the environment. The framework allows to draw a more comprehensive picture of the relevant quality aspects and, as a result, aid better informed decision-making.

The diagrams in Figs. 2 and 3 are snapshots in the time of these case studies and do not characterize the life cycles. The concrete case studies, the four dimensions, as well as the elements and relations in these diagrams have their own life cycles. In particular, the relations and their quantification will likely change over time—the initial deployment...
of infrastructure for a PCS requires a substantial energy investment up front, but situation-aware systems will accrue significant benefits over time. While first-order and second-order effects can indicate one trajectory in the assessment of sustainability, the effects on global goals can easily change or even reverse this trend. Furthermore, the impact of a software system on the environment would be very different depending on the framework conditions as influences and concerns. Thus, any concerns that are related to sustainability quality requirements have to be prioritized and traded off.

The notion of sustainability is very broad and entails a long chain of (possibly circular) consequences crossing all dimensions. When identifying the concerns pertaining a software system, it is imperative to define a clear scope, the boundaries between the sustainability concerns directly influencing a software system, those that fall outside the scope but that could be useful for decision making, and those that are too far away and hence are not considered. The ISO/IEC 42030 working draft models the Environment in which a system is situated. In our understanding, part of such environment falls within the system scope, while part (outside the system boundaries) falls outside. However, sustainability requirements and concerns will likely increase the system scope. Further research is needed to better define how environment and boundaries can help in, e.g., architecture evaluation.

There are limitations to what the sustainability analysis framework can provide. The influences between the Sustainability Quality Requirements have to be determined by the developers and/or stakeholders, as the framework can only provide the means for linking them but not the analysis itself. Constraints and parameters have to be chosen by the developers as it is not possible to list them in a way that is generic enough to be applicable in all circumstances and, at the same time, specific enough to actually be useful. The best guidance we can provide with this framework is via examples that show how to apply the framework and what the benefits are. Part of future work is to extend this guidance with further examples.

5. CONCLUSIONS

This article presented a framework for trading off sustainability quality requirements from the different dimensions of sustainability. The sustainability analysis framework is based on the Third Working Draft of ISO/IEC 42030 Architecture Evaluation [4] and is a first attempt to discuss the multi-dimensional impact of software on its environment.

As such, our contribution may assist software practitioners in making careful trade-offs. These trade-offs exist not only among technical and economic quality aspects, but also in relation to the social and environmental ones. We focus on classifying sustainability quality requirements as the first step toward sound decision making, trade-off analyses and quality evaluation. The application of the framework in practice enables business software developers to specifically consider the neglected environmental and social dimensions in relation with the technical and economic dimensions.

It is imperative that we develop a common understanding of the relations among sustainability aspects to improve our capabilities as software engineers to enhance software sustainability. Using the framework, practitioners may determine their sustainability goal and use the framework as the basis to see the potential outcomes and impacts of the criteria. We hope with this work to provide food for thought for new research directions, and constitute a foundation for discussing the integration of the different ISO quality models.

Future research will focus on how sustainability quality requirements of the framework can be systematically deduced from a goal model while considering the various impacts of software on its environment. This includes how to refine
such information in form of constraints towards design and implementation. Furthermore, the resulting models could be used for cost estimation, specifically in terms of the impact on architecture and infrastructure. Another open challenge is scoping, specifically distinguishing sustainability concerns outside the software system but directly influencing it, so that the information would be useful for taking the best decisions, and selecting too large a scope. Finally, as there are no standardized metrics, applying the framework can help to establish and build up sound metrics, which would in turn serve as basis for building satisfactory tool support.

6. ACKNOWLEDGMENTS

This work has been partially sponsored by the European Fund for Regional Development under project RAAK MKB Greening the Cloud, the Deutsche Forschungsgemeinschaft under project EnviroSiSE (grant PE2044/1-1), and Swedish Foundation for Strategic Research (SSF) via the project RALF3. Thanks go to the participants of the GREENS Workshop at ICSE 2013 who contributed thoughts and reflections, and especially Henning Femmer and Hausi Müller.

7. REFERENCES


Part IV

Evaluation
The Evaluation part contains the publications with regard to feasibility studies on the concepts developed in this habilitation in different application domains for software systems.

These publications were partially developed in collaborations. The following list sets the context for the elaboration of each of the publications.

- **RE@21: Time to sustain** [64]
  This contribution was developed in collaboration with Henning Femmer and shows the application of the reference model for sustainability applied to conference organization activities. It was published at the 2nd International Workshop on Requirements Engineering for Sustainable Systems in 2013.

- **Domestic Plant Guilds: A Software System for Sustainability** [52]
  This contribution was developed in collaboration with Juliet Norton, Alex J. Stringfellow, Joseph J. LaViola, and Bill Tomlinson at UC Irvine. It was published at the 2nd International Workshop on Requirements Engineering for Sustainable Systems in 2013 and shows the application of the artifact elaboration for a system from the permaculture community.

- **An Assessment Technique for Sustainability: Applying the IMAGINE Approach to Software Systems** [79]
  This contribution was developed in collaboration with Alejandra Rodriguez at TUM. It was published at the 2nd International Workshop on Requirements Engineering for Sustainable Systems in 2013. The paper evaluates the application of a scenario analysis technique from sustainable development to software systems.

- **Supporting Physicians by RE4S—Evaluating Requirements Engineering for Sustainability in the Medical Domain** [47]
  This contribution was developed in collaboration with Joseph Mehrabi and Debra Richardson at UC Irvine. It shows the application of the RE4S approach to a system from the healthcare domain. The paper is currently under review at the 4th International Workshop on Green and Sustainable Software.
16 RE@21: Time to sustain
RE@21: Time to Sustain!

Birgit Penzenstadler
Donal Bren School of Informatics and Computer Science
University of California, Irvine
Irvine, CA, USA
bpenzens@uci.edu

Henning Femmer
Software & Systems Engineering
Technische Universität München
Munich, Germany
femmer@in.tum.de

Abstract—The requirements engineering (RE) conference is turning 21, our conference is “grown up” now. We take this as a motivation to rethink our responsibility for this conference, and to prepare the path for sustainable RE conferences for future generations.

Although sustainability has been broached as conference topic in RE’08, hardly any long-lasting consequences followed. We consider this as vital for improving the research community experience while minimizing especially the environmental impact of the conference.

We contribute a model for improving the sustainability of the RE conference by systematically analysing the different dimensions of sustainability and their corresponding impact with regards to this conference. On that basis we define actions that improve a specific aspect of sustainability and describe how the actions are implemented at RE’13 in Rio.

Our envisioned long-term impact for the conference is that by providing a hands-on case study every conference participant can actively support, this paper can lay the foundation for a sustainability guideline for further improving the sustainability of the RE conference series.

Keywords—requirements engineering; sustainability; generic model; case study; conference organization;

I. INTRODUCTION

This year, the requirements engineering (RE) conference is celebrating its 21st birthday. We consider this an adequate moment to show the maturity by taking responsibility for the sustainability of the conference: to maximize its benefits and to allow positive experience while reducing its negative impact on the environment, for this conference as well as for future iterations. This issue was already identified at ICSE a few years ago [4] and chosen as topic at RE’08, but not dedicatedly continued and therefore led to only little change and no long-term effect.

To achieve an actual long-term effect, it is necessary to define an explicit goal and to assign a responsibility, as a goal will only be realized if it has an actual stakeholder. This stakeholder must take care of a systematic approach, which includes defining sustainability, understanding what sustainability means for this conference, defining and starting actions, and finally understanding if these actions were a success.

Consequently, for the first time in this conference series, the general chair defined a role within the organization committee and thereby created an explicit stakeholder for sustainability: the sustainability chair, which the first author of this paper was asked to fill this year.

For identifying the goal, we have to create a tangible definition for sustainability in our context. The first author defined sustainability for software systems (based on the definitions by Brundland [16] and Goodland [5], and the frameworks by Burger and Christen [3] and Robert [14]) as not only conserving the environment but also satisfying individual, social, economic and technical needs [11].

Definition: In the context of the requirements engineering conference, we define sustainability as finding a good balance between technically sound and inspiring research, a socially enjoyable community gathering, and an environmentally sustainable event with a positive spirit.

Goal: The aim of the sustainability chair is to optimally support this perception of sustainability at the International Conference on Requirements Engineering 2013 (RE’13) and to provide the basis for continued optimization at the following conferences.

Problem: Sustainability is not an issue completely new to the RE conference series. The theme of RE’08, the conference giveaway, and two out of three keynotes of that year were targeted to sustainability. However, since then the topic has likely circled the minds of some organizers but, amongst all the challenges an organization chair has to face, was not an explicit objective, nor supported with concrete actions.

Contribution: The contribution is to make sustainability more tangible for the RE conference and to be able to systematically analyse and effectively support the objective within the conference preparation and realization. Therefore, we provide a sustainability model that covers the different aspects of sustainability, define actions for RE’13 and indicators that help to assess their impact, and describe the implementation of a subset of these activities along with the evaluation plan.

Impact: This paper provides a means of communicating that we as the RE community should consider sustainability as an important objective. It is the basis for a conference sustainability guideline that will enable the future organization chairs to build upon previous knowledge and lessons learned to further pursue the goal. Furthermore, the case study provides an illustration for engineers of how to use the generic
sustainability model to systematically analyse and implement sustainability measures in a specific context.

Outline: The remainder of this paper is outlined as follows: Section II presents related standards and previous work on the topic, Section III describes the generic sustainability model, and Section IV proposes the instantiation for the requirements engineering conference 2013 and the actions taken towards implementation. Section V discusses the model and the approach and Section VI concludes the paper with an outlook on future work.

II. BACKGROUND & RELATED WORK

A. Standards

The ISO 14000 [1] is a family of standards related to environmental management that exists to help organizations minimize how their operations negatively affect the environment, comply with applicable laws and regulations, and continually improve in the above. The ISO 26000 standard [2] offers guidance on socially responsible behavior and possible actions; it does not contain requirements and, therefore, in contrast to ISO management system standards, is not certifiable. Both have been considered in our work.

B. Related Work

Related work includes modelling sustainability for software systems and for conferences as well as previous initiatives at the RE conferences.

1) ICSE i* Sustainability Model: Cabot et al. [4] report on a case study for sustainability as a goal for the organization of the ICSE’09 conference with i*-models to support decision making for future conference chairs. Stefan et al. [15] extend that work for managing environmental sustainability with quantitative goal modelling techniques. Both works provide model instances for specific case studies while our work also provides a generic reference model.

2) Software System Sustainability Goal Model: Mahaux et al. [9] assess how well some current RE techniques support modelling of specific sustainability requirements in that case study. In contrast, our aim is to provide modelling means explicitly for integrating sustainability into the organization process as a major objective.

3) RE’08 Theme: The theme of the RE conference in 2008 was sustainability. The efforts of communicating the topic were a matching conference giveaway (reusable water bottle) and specifically targeted keynotes, held by van Ypersele [17] and Piñón [13].

4) RE4RE and RE Interactive: At RE’11, Martin Mahaux and Alistair Mavin started a process of collaborative requirements gathering for the requirements engineering conference (see also http://re4re.cetic.be/). At RE’12, a half-day workshop led to a validation of the goal model and over 200 requirements and solution ideas. As a consequence of the ideas most voted for, the new initiative “RE interactive” launched by the RE’13 Program Chair seeks to make the RE conference more engaging. The purpose of those initiatives is to foster social interaction, one major aspect of social sustainability.

5) Common Cause: The Common Cause Handbook [8] presents a framework for a movement towards a more sustainable, equitable and democratic world that serves as one of many inspirations for our work.

III. THE SUSTAINABILITY MODEL

We present a reference model for sustainability that decomposes sustainability into five dimensions: environmental, individual, social, economic, and technical sustainability (longevity of technical infrastructure) [11]. The model provides activities and relates them to the values, which they support, and to assessable indicators.

The generic sustainability model is intended to serve as a reference model for a process engineer, who instantiates the model for organization processes or development processes, and for a requirements engineer, who instantiates it for a specific system under development. Our proposed method comprises the generic sustainability reference model, the respective metamodel behind it, and instances derived for specific processes and systems.

A. The Metamodel

The metamodel is comprised by the types Dimension, Value, Indicator, Regulation, and Activity and their relations [12]: A <Dimension> is a viewpoint, represented by a set of values that express the abstract objectives of the dimension. Each dimension is represented by a set of values. A <Value> is a rationale that is rooted in itself, and is approximated by indicators. An <Indicator> is a qualitative or quantitative metric and is related to a value. A <Regulation> is an optional element that affects a value. An <Activity> is a means to support and influence a value.

B. The Generic Sustainability Model

The model consists of three levels, see Fig. 1: the top level contains the dimensions; the middle level contains values, indicators, and regulations; and the lower level contains activities. Each element in the generic sustainability model is of a type from the meta-model explained before. For example, for the dimension social sustainability, the spirit of the community is an important value that can be decomposed in different values such as trust or education. The education value is regulated, amongst others, by human rights. This value can only be assessed roughly and individually by indicators, where one indicator contributing to that assessment is the level of graduation of a person. Education is fostered by different activities, such as knowledge management, education programs or mentoring.

The generic sustainability model is intended to serve as a reference for the instantiation of process- or system-specific instances, for example, the value education can directly be reused, while the activity knowledge management must additionally be instantiated in the application context, e.g. into use the company-wiki in the intranet for a concrete company. Further details and the generic instantiation process are explained in [12], where instantiations for various case studies are described.
IV. Instantiation for the RE Conference

This section reports on the development and the implementation of the explicit support of sustainability at RE’13. The methodical approach for the usage of a sustainability model consists of analysing the sustainability dimensions and constructing the model, applying identified actions, and assessing the defined indicators.

At RE’12, the organizing committee (OC) informally talked about the effect that emphasizing sustainability as an objective for the RE conference could have in its different dimensions, e.g., environmental impact, economic balance, and knowledge management. However, every objective needs a stakeholder, so before starting to develop the sustainability model in depth, the OC decided to create an explicit stakeholder by naming a role: the sustainability chair. The first author of this paper was asked to take on that role and the following sections describe our systematic approach to not only improve the sustainability of RE’13 but of all following RE conferences in the hope that we might be able to provide an example for other conferences.

A. Analysis: The RE Conference Sustainability Model

The RE conference sustainability model evolved in a initial brainwriting sessions on the basis of the generic sustainability model, a few additional short brainstormings (for various levels of change acc. to [10]), and a number of iteration cycles between members of the organizing committee who provided feedback.

The model was decomposed into five submodels, based on the five dimensions of sustainability as defined in Sec. III. For simplicity, we removed relations between multiple dimensions of sustainability, but placed activities and values where they fit best. These submodels, as depicted in Fig. 2-6, were used by the organizing committee of RE’13. The legend of Fig. 2 is also used in the later ones.

The dimension individual sustainability (Fig. 2) focuses on the personal sustainability over the course of a person’s lifetime. Values for individual sustainability that can be associated with the setting at a conference are development & growth, dignity, curiosity, and health. Some of the activities we derived for these values are to provide sufficient time and space for discussion (in the sessions and also separately for new collaborations), to provide a continuous forum for how people can make the most of the conference (see the RE4RE initiative), to enable and improve the access for people with disabilities (hearing impaired, etc.), to enable people who cannot travel to follow the conference, and to digitalize the discussions for follow-up and knowledge management.

For the social dimension (Fig. 3), the values rated most important were community building, fairness, trust, and tolerance. These values can be supported by enhancing and establishing the conference culture, and by ensuring that the conference has a positive impact on the local community and their relation with the conference participants.

The environmental dimension (Fig. 4) analyses where we can save resources by dematerialization [7] and avoid waste, from water and air to energy, emissions and garbage. Clearly the biggest environmental impact of the conference is having people fly in from all over the world for a few days of
Create sustainable pace in time table

**Goal** Sustainability

**Dimension** Individual Sustainability

- **Value** Human Health
- **Value** Curiosity
- **Value** Dignity
- **Value** Development / Growth

**Activity** Discussion: How can people make most of the conference?

**Indicator** Vividness of discussions

**Activity** Enable access for people with disabilities

**Activity** Enable people who cannot travel to follow the conference

**Activity** Enable people who cannot see/hear to follow the conference

**Activity** Provide time and space for discussions

**Activity** Create shepherding or workshop for newbies

Fig. 2. Example instance for the individual dimension of the Requirements Engineering Conference.

**Goal** Sustainability

**Dimension** Social Sustainability

- **Value** Community Building
- **Value** Fairness
- **Value** Trust
- **Value** Tolerance

**Activity** Make attendees want to return

**Activity** Watch discussion fairness

**Activity** Ensure Conf has a positive impact on people working for us there (fair trade)

**Indicator** Number of returning attendees

**Activity** Ensure the conf helps people to discover the Brazilian people and culture

**Activity** Create open data platforms

**Activity** Establish an open, supportive conference culture

**Activity** Questionnaire on subjective Impression regarding value

Fig. 3. Example instance for the social dimension of the Requirements Engineering Conference.
conference. This is also the most obvious goal conflict in the model, i.e., researchers are aware that they cause bad environmental impact by having global conferences, but at the same time the informal networking at these conferences is the most common spark for new research ideas and collaboration.

**Economic sustainability** values can be differentiated in saving money for the organization, saving money for the attendees, in economic fairness, and long-term break-even. For each of them, the submodel in Fig. 5 provides at least one supporting activity.

**Technical sustainability** (Fig. 6) is reduced to the aspect of knowledge conservation, as we wanted to abstract from the specifics of technical platforms.

### B. Implementation: Realizing Actions

On the basis of these models, we selected a number of actions according to the practicability we estimated for them. Target areas were the preparations of the conference, sensitizing participants for travel impact (analysed in [6]), the conference giveaways, the advertising by sponsors at the conference, the conference venue and its catering, interaction, and lastly knowledge management.

1) **Conference Preparations**: Communicating with the organizing committee, especially the program chair, the local chairs, and the homepage team involved mainly email exchange. We agreed on the chosen subset of the actions with the program chair, addressed the respective organization committee members with our suggestions and asked for their opinion and support in realizing them as described below. The efforts are also summarized on the conference website.

2) **Sensitizing participants for travel impact**: To communicate to participants how they can make their travel to Río more (environmentally and economically) sustainable, we added some information to the conference website.

   - Hostels along with the hotels, so that participants can choose a more economic option for accommodation that is likely to produce less environmental impact than a middle-to-high class hotel.
   - Eco-tourism providers in the “Explore Brazil” section, so participants can plan trips before or after the conference in a sustainable way.
   - Information on public transport (airport, venue) and a suggesting note on flight emission compensation.

The option of enabling online participation at the conference was briefly discussed with members of the organizing committee. There has to be found a business model that balances a reduced fee that is adequate for participation via a web conference with being able to provide a specific service level agreement at the same time for an online participation that provides less social interaction than a physical presence.

3) **Conference giveaways**: We contacted two manufacturers of reusable water bottles with a built-in filter and asked them to act as sponsors for the conference giveaways. Although filtered water is available at the venue, these bottles provide a useful accessory for traveling after the conference or back home. One of them agreed to our proposal and expressed their enthusiasm for helping us make the conference more sustainable.

4) **Advertising**: For other sponsors, for example IEEE Computer, we invented new ways of advertising to avoid shipping great amounts of printed magazines where it was not feasible (environmental and economic sustainability). We
5) Venue: The venue for a conference dinner with Brazilian specialties and cultural background was already chosen by the time the model was developed. However, a local caterer was chosen and we agreed not to use plastic dishes/cups/cutlery, and to provide water to refill the reusable glasses and conference giveaway bottles instead of providing small plastic water bottles.

6) Interactive Sessions: The initiative “RE Interactive” mentioned in Sec. II implements some of the social sustainability actions, e.g., to provide sufficient time and space for discussion and interaction between the participants.

7) Conference News and Knowledge Management: In social media, the RE conference is already present on Facebook and Twitter. In order to provide better reporting during the conference, we reflect the tweets on the website. For RE’14, we envision to add a blog to the website that keeps track of the discussions during the conference as well to improve information for people who cannot attend the conference.

For new committee members (whether organizing or other), there is always the question of how to get the information from the people who filled the role before them, and for the ones leaving the roles, the question is how to conserve their knowledge and lessons learned for their successors. The answer is knowledge management, but the implementation remains to be chosen. At RE, knowledge transfer happens mainly via email, so that the knowledge is distributed and likely to get lost. This year, we attempt to collect the most important information in a draft RE OC guide, most likely in
a wiki. Naturally, this includes information for preparing the RE’14 conference by the future sustainability chair.

8) Summary of Actions: When trying to quantify how many activities we tried to realize and which ones we had to skip for this conference, we have to look back at the submodels presented in the last section. In the submodels, there are also activities included that have been worked on before at the RE conference, for example, to “create a sustainable pace in the timetable”. In the individual and the social dimension, we are working on all activities. In the environmental dimension we initiated 14 out of 16 activities — following the conference via Skype and registering for printed conference materials were omitted. In the economic dimension, we have not created stipends yet, but student volunteers are the best opportunity without significantly more budget.

C. Assessment: Plan for Measurement and Improvement

The assessment of the model is two-fold: First, we need to evaluate the indicators to understand the impact of the activities on sustainability. Second, we need to understand and improve the quality of the model itself.

1) Measurement of Indicators: The effectiveness of the activities will be measured via a number of indicators (see submodels in Fig. 2 - 6), quantitatively where possible and qualitatively, throughout and after the conference. The main aspects are listed in Table I. We are interested in the percentage of people who made use of our suggestions and offers, and especially how many people used these suggestions the first time.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vividness of discussions</td>
<td>Questionnaire &amp; comparison</td>
</tr>
<tr>
<td>Number of returning attendees</td>
<td>Questionnaire</td>
</tr>
<tr>
<td>Subjective impression on social values</td>
<td>Questionnaire</td>
</tr>
<tr>
<td>Costs of conference</td>
<td>Accountancy</td>
</tr>
<tr>
<td>Individual expenses for Conference</td>
<td>Questionnaire</td>
</tr>
<tr>
<td>Water consumption</td>
<td>(Prob. not measurable)</td>
</tr>
<tr>
<td>Aggregated emissions</td>
<td>(Prob. not measurable)</td>
</tr>
<tr>
<td>(New) Usage of emission compensation</td>
<td>Questionnaire</td>
</tr>
<tr>
<td>Used garbage bags</td>
<td>Feedback from caterer</td>
</tr>
<tr>
<td>Organisation effort</td>
<td>Feedback from chair</td>
</tr>
<tr>
<td>Number Knowledge Documents</td>
<td>Counted after conference</td>
</tr>
</tbody>
</table>

2) Feedback for Improvement: We understand that the sustainability model is just a first step that needs to continuously improve and be adapted. As such, it heavily depends on the support and feedback of the conference participants. Hence, we not only welcome, but need suggestions and criticism from chairs and attendants of the conference. As visibility is a key success factor, we will advertise the on-going process by awarding the best improvement suggestions (in terms of importance and applicability) during the conference.

The sustainability model gives a well-structured form for finding new feedback: The feedback will assess existing activities, and furthermore focus on finding new values, activities and indicators.

V. Discussion

In this section, we discuss the completeness of the model, cost estimation, success approximation, problems and limitations, and dissemination to future conferences.

A. Completeness of actions and impact

An interesting question is the completeness of the model with respect to the possibilities of action and with respect to the impact the conference actually has.

The model relies on the generic reference model, which provides a basic selection of activities and indicators. However, due to the goal of being generic, there is still a lot of room for exploration of the specific context and its possibilities. We cannot claim that the model is complete and therefore neither that it would absolutely maximize the sustainability of the conference.

B. Cost estimation

The costs for implementing the actions are rather low. It required personal effort to gather information, to email and convince people by new ideas, and to organize logistics and technical platforms. Consequently, there is no reason against supporting the initiative — it just needed a stakeholder.

C. Success approximation

To estimate what we can actually achieve by implementing these activities at the conference, we have to rely on a number of indicators that shall provide measures for their successfulness. However, the choice of indicators always reduces a goal to just that number or qualitative measure and it is hard to actually approximate a value by such reduced means. Therefore, we are aware that the indicators can only roughly approximate the effectiveness of the activities and we have to rely on mainly qualitative data, such as personal opinion and individual impression by the conference participants.

D. Problems & Limitations

There is a number of problems and limitations to our initiative: There are inherent goal conflicts between some of the objectives in the submodels. For example, there is the ever-present trade-off between the most environmentally sustainable choice with minimized environmental impact and the most economic choice. Furthermore, there is a trade-off between making the conference a great experience for all participants and the budget limits. There is no actual solution to the problem — as in standard goal modelling, we can only opt for a good balance.

Travelling to the conference is definitely the biggest environmental impact that is caused, but following the conference via Skype or webcast as indicated in Fig. 5 is currently not supported in the registration process.

Furthermore, as measuring is difficult due to the reasons named above, reduction to indicators and the necessity to rely on individual feedback, actual proof of the effectiveness of the activities and their impact is limited.
However, the most important limitations are acceptance and support by the community. We do not know how successful we will be in establishing a culture of sustainability at the RE conference, how willing participants are to adapt some of their behaviour, for example, with regard to traveling. The idea cannot be to impose rules and constraints but has to be to convince and motivate people so they feel inspired and voluntarily initiate change.

E. Dissemination to future RE & other conferences

Comparing our work to Cabot et al. [4], we can only assume that their goal model is still used as inspiration by the general and local chairs of the conference. According to a private conversation with a recent former ICSE chair, sustainability is being taken into account to some extent, but there is no explicit strategy and no explicit stakeholder for it, as a conference organization in that dimension already holds a huge number of challenges to be mastered.

For disseminating the initiative to future RE (and other) conferences, we have to show the actual benefits and to establish a culture of sustainability at the conference, including an explicit stakeholder in the organizing committee.

VI. CONCLUSIONS

This paper presented a sustainability model tailored to the RE conference and described how it was developed and implemented for and at the current RE’13. This included the definition of activities for the different dimensions of sustainability and the selection of indicators to measure their impact as well as the communication and logistics to realize them at the venue. We furthermore discussed limitations and trade-offs that had to be made.

a) Next steps: We will update and assess the indicators to report on the effectiveness and impact of the implemented activities. On this basis, the future pursuit of the initiative will be shaped.

b) Appeal: We want to encourage participants to take part in the activities and contribute new ideas. We hope to carry this forward throughout the “grown-up” RE conference series.

c) Future Work: The proposed model is a bottom-up approach to improve the sustainability of the conference. More fundamental questions that need to be discussed in our research community are:

- Is the RE conference sufficiently supporting research about sustainability in RE?
- Is the process of the RE conference sustainable? Do current review mechanisms and closed access publication support or hinder good research?
- Does the RE community have a positive impact on the real world? Is the effect of the scientific contribution greater than the negative environmental impact?

ACKNOWLEDGMENT

The authors would like to thank Martin Mahaux for suggesting the explicit role of a sustainability chair at the RE conference and his feedback on drafts of this paper. Olly Gotel and Alistair Mavin for feedback on proposed activities, to the local organizing committee for their support in implementing the activities, and Jonas Eckhardt for feedback on a draft version of this paper. This work is part of the EnviroSiSE project (grant number PE2044/1-1) funded by the DFG in Germany.

REFERENCES

17 Domestic Plant Guilds: A Software System for Sustainability
Abstract—This paper presents the design concept for a software application to aid users in the development of a sustainable backyard food and resource system. It presents how we gathered and analyzed the requirements for an application that achieves a balance of user convenience, awareness, and sustainability in the context of creating a domestic plant guild. A plant guild is a community of plants that sustains itself and provides for people’s essential needs.

Based on a field study, the authors present requirements for building a Plant Guild Composer as one approach for addressing the aforementioned challenge. The requirements have been generalized because future applications that aid in the construction of a sustainable human system, which supports some human needs in the future nor contributing to environmental degradation, may encounter similar design challenges.

Index Terms—requirements engineering; sustainability; plant guilds; permaculture; human-computer interaction

I. INTRODUCTION

Complexity characterizes many of our lifestyles. Empirically speaking, it is a side effect of our options for nearly every aspect of life, from which career to pursue to what to eat for dinner. The use and development of technology is often intended to alleviate some of life’s complexity via convenience, e.g., finding the exact doll your daughter wants for her birthday online is simpler than calling and driving to local toy stores. However, in effort to make life simpler, technology and the information provided by it has given us more choices and options than ever before. In sorting through these complexities, the need and ability to act sustainably is unnoticed or forgotten. In her visit to Change Islands, Phoebe Sengers discovered a lifestyle characterized by simplicity due to the severely limited access to technology and exotic resources, and the necessity to work for subsistence [13]. In many ways this simplicity rendered a more sustainable society and arguably improved their quality of life. As observed in the first authors ongoing ethnography of transition movements in Central Florida and Southern California, there are communities

with an abundance of resources strive to achieve a similar simplicity, but are inclined to use technology to achieve it.

A. Ties to RE4SuSy and Related Work in HCI

One of the objectives for the RE4SuSy workshop series is to develop Requirements Engineering (RE) techniques that help us design software systems that support sustainable lifestyles. Last year’s contributions looked at how to improve existing RE techniques for green software [1]. Past HCI research has primarily approached the problem of unsustainable practices by outlining improper resource usage and its implications [3]. The issues associated with transitioning to a simple, sustainable lifestyle remain scarce amongst HCI publications, as it is established as a field of research in [2], and not explicitly mentioned in [3]. In contrast to both RE and HCI existing research, this paper provides an example system that supports a sustainable lifestyle and may be used as a case study in the future.

B. Sustainability and Permaculture

From our understanding, sustainable living can be achieved by managing consumption of extraneous materials, i.e., things we replace and dispose of without regard, and increasing self-sufficiency, i.e., providing for yourself. While Pierce approaches the topic of extraneous materials by encouraging reflection of how they are used [11], we are interested in enabling people to live independently of these extraneous materials by providing themselves with materials they need. Practices like permaculture (the eco-, human-sustainable design for permanence) advocate for similar ideologies, e.g., apply self-regulation and accept feedback [5]. Although independently ensured food and resource security is an innate step in becoming self-sufficient, this transition poses rather intensive, short-term complications. Namely, a great time investment is required to learn these methodologies and their implementation.

In permaculture, a domestic plant guild can foster human independence from extraneous materials. A domestic plant guild, as for example depicted in Figure 1, is a family of plants that can sustain itself and provide people with many of their essential needs (e.g., food, building materials, etc.) [7].

---

1 Portions adapted from a previous workshop paper [8].

Copyright © 2013 for the individual papers by the papers’ authors. Copying permitted only for private and academic purposes. This volume is published and copyrighted by its editors.
is an enabler of simple, sustainable living, i.e., reduces cost of living, ecological footprint, and the need for consumer goods.

C. Example Domestic Plant Guild Scenario

Picture yourself stepping out into a quiet backyard; your property is fenced in with an assortment of sugarcane and bamboo, protecting your privacy, and your other plants from wind. This same bamboo plant was used to build the bench you just sat down on. You gently stir your morning tea with a sweet cutting of sugarcane, yet another function of your privacy fence.

A brigade of bees and butterflies hover busily around your native goldenrod, gaillardia, coreopsis, milkweed, and sunflower. Where you would normally have grass, the native and hardy Gopher Apple gladly covers the ground; you even occasionally see a Gopher tortoise creep into your yard and lazily munch on his favorite snack.

A large persimmon tree provides shade for your bench as well as a nice little treat for you and the local wildlife. A passionflower entangles the persimmon tree, while its flower bobs happily around in your tea.

You begin to search for salad ingredients, your lunch, occasionally picking a few blackberries and sparkleberries; they taste quite like a blueberry, but half the maintenance. You pull a couple of leaves off the French Sorrel, the Moringa, and fennel. To top it off, you snag an avocado and some rosemary. You don’t have to water or prune, you just have to eat your harvest. This is an ideal domestic plant guild.

D. Challenges in Plant Guild Design - A Need for good RE

The design and construction of a plant guild requires time and expert knowledge, two factors that prevent many from incorporating one. Therefore, if we can provide support tools to make it easier for people to develop and establish domestic plant guilds, then the gap between the idea of sustainable environments and their realization can be reduced. We believe that utilizing adequate RE techniques [9], [14] in the development of these software tools is essential in providing easy to use yet powerful interfaces to support domestic plant guild construction.

Outline

This position paper discusses, in the context of the Plant Guild Composer (PGC), the challenges of designing a software application with the intended use of supporting a sustainable lifestyle. Its focus is to mask the complexities of domestic plant guild creation through simple interfaces utilizing common and easily observable information about the environment.

We believe our topic is suitable for the RE4SuSy Workshop because (i) it focuses on gathering requirements at the intersection between human and environmental sustainability, and (ii) provides an example system in a new application domain for supporting the facilitation of sustainable lifestyles while masking complexity.

II. PLANTS GUILDS

In the natural environment, plants exist in complex mutualistic communities. Within these communities, or plant guilds, each plant has a unique impact on the surrounding environment. Their impact may make one nutrient or resource more available or remove harmful constituents for neighboring plants. When considering an assessment of plant guilds, we assume that each plant has needs (i.e., inputs), products and behaviors (i.e., outputs), and intrinsic characteristics (i.e., physical traits); all are components of the plant’s functional analysis. Once these plant characteristics are understood, a plant guild could be created to act as a semi-closed system, the only external inputs coming in from the natural environment. When this plant guild is used to provide outputs for human consumers, it becomes a domestic plant guild.

A. Central Objective of a Plant Guild

A domestic plant guild is one that is designed to insert a human component into this naturally closed system. Extraneous products from the plants provide edible, medicinal, and useable parts. Thus plant selection requires consideration of both fulfillment of the guild’s and human’s needs. The plants used in a guild are predominantly perennial (living more than 2 years) or self-seeding annuals, and if at all possible, native to the region. If the plant guild is designed and established properly, the human role is reduced to harvesting what is needed.

B. Principles for Designing Plant Guilds

To achieve a balanced domestic plant guild, experts initially model natural system dynamics (i.e., plant community functional analysis), then fit it to domestic constraints. The guild is designed to be convenient for humans (e.g., co-location of similar plant types and navigable paths) while still encouraging natural states of action between species (e.g., coaction, interaction, and inaction) [7], and features plants that support the humans’ desired lifestyle (e.g., berries for jam production). The design of a plant guild is also dependent on implementation constraints (work, financial, and material resources). In general, the higher the human demand, the more initial energy required for the guild to reach establishment. In context of each constraint, plants are placed in appropriate geographic locations and functional arrangement. If designed correctly, the plant guild should not require human support past its establishment (i.e., when the trees reach maturity), though minor manipulation may increase its robustness (e.g., seasonal pruning of nutrient rich plants for mulch and fertilizer). However, due to a variety of variables, success is not guaranteed. It is possible that further consultation may be required, especially for complex, high-yield guilds. The basic design principles that permaculture designers follow are lined out in Figure 2.

Due to the intensive process of creating domestic plant guilds, the knowledge required to properly design and establish one is currently only possible by experts. As such, we’ve conceptualized the PGC, a tool to support the development of
domestic plant guilds which, in turn, will empower a simple, sustainable lifestyle.

III. REQUIREMENTS GATHERING IN THE FIELD

The first author is actively engaging in a longterm ethnography of transition networks, specifically those that practice permaculture. During the fall of 2011, the first author began her participant observation by completing a 9 week, 72 hour Permaculture Design Course (PDC) certified by the Green Education Center\(^2\) and Simple Living Institute\(^3\). It was in discussions of learning curves and hardships of transitioning to sustainable living that the concept of the PGC came to light. Through the spring of 2012, she continued her observation at the Econ Farm in Orlando, Florida, sharpening her knowledge of plant guilds and other sustainable human systems. For designing the PGC, she consulted the Education Director and then Garden Manager (second author) at the University of Central Florida Arboretum\(^4\), and three community permaculture experts, all of whom are well versed in plant guilds and the cultivation of plants for food, medicine, and other materials. In Fall 2012 the first and second authors continued the field research at a private residence in Winter Park, FL, installing a plant guild of tropical fruit trees, shrubs, and ground covers.

In the Winter of 2013, the first author relocated to Southern California to compare and contrast plant guild design between the two drastically different locations.

A. Interview Series

From expert interviews in Florida, five in total, we have come to the conclusion that the user experience design of the PGC dictates the tool’s ability to effectively model and assist the design of a plant guild. Identifying and placing plants in a functional arrangement is difficult for non-experts. They also do not have the skills necessary to acquire environmental information (e.g., soil type, sun patterns, points of erosion, etc.) or the means to implement such a complex design. Due to plant guild implementation constraints (e.g., budget and time), the experience of creating it, beyond the interaction with the PGC, must be considered in the tool’s user experience design and how it models and simulates a domestic plant guild.

B. Central Requirements by Permaculture Designers

From the results of our ongoing field study, we’ve established the following set of requirements for the design of the PGC. We have generalized these requirements because we feel future applications that aide in the construction of a sustainable human system (i.e., an environmentally beneficial system established with the intent to support some human need) may encounter similar design challenges.
1) Use ecocentric and anthropocentric metrics to qualify the designed human system as sustainable: The qualifying metrics will ensure that humans’ needs are supported in addition to the plants'. These metrics may include, but are not limited to: benefit and utility to the human, protection of the native ecosystems, and soil fertility restoration. The designs produced via the application must qualify as sustainable to alleviate the problems contributing to environmental detriment.

2) Identify environmental data required to design the most efficient sustainable system: These requirements include data only available to experts or acquired by professional technology. Given all the necessary environmental data, the application should enable the user to produce a sustainable human system design optimized for the intended environment.

3) Design to condense the time the user spends with the application relative to the entire process of creating the sustainable human system: The user’s existing knowledge of the system and receptiveness of the lifestyle change are key factors in the time they’re willing to spend in the design process. To determine an appropriate amount of time spent using the system an upper-bound may be found by observing potential users who are highly motivated to make the sustainable lifestyle transition, and a lower-bound may be found by observing those who are resistant.

4) Provide the user with the information they need to implement the design and utilize the system: The application should be incorporated into a complete system construction program, whether it is computerized or teacher-guided, so that the system is installed and utilized. Such a program should take into account the directly related socioeconomic concerns of the user in the application design, an implication for design established in [4].

5) Maximize both sustainability and convenience: Maximizing both sustainability of the designed human system and user convenience in designing and implementing the system is key for creating effective software in this domain. The convenience metric is dictated by the amount of work the user is willing to do, and will vary depending on user expertise. Systems that focus only on convenience are frequently unsustainable (e.g., plastic plates and utensils so the 1950’s housewife doesn’t have to wash dishes after a party). Conversely, systems that prioritize the sustainability of the designed human system may be too difficult for non-experts to use without an intensive time commitment to learning.

6) Require users to provide easily observable environmental data, but use experts and power users to fill in information gaps: Typical users should provide the system with easily observable information so that they become better acquainted with their environment. Experts and power users (i.e., non-expert self-motivated users) will likely put forth more effort than average users to provide the system with detailed information. This information should be catalogued and used to infer necessary environmental information not provided by average users (see [12] for an example) and to optimize future designs created by the sustainable human system design application.

These generalized design requirements, intended to enable simple transitions to a convenient, more sustainable lifestyle, are geared towards small-scale sustainable human systems (i.e., for families or communities versus regions or countries).

IV. REQUIREMENTS DOCUMENTATION

After the first set of interviews and the first data collection sessions of the ongoing field study, we used an artifact-based approach to consolidate the gathered information. The requirements artifact model is based upon earlier work by [6] and [10].

The figures provide an overview of the stakeholder model, the goal model, the context and system vision, and the constraints.

A. Stakeholder Model

The stakeholder model (Fig. 3) depicts the organizational and constraining units like the housing authority and the local government, the research environment with the university, the advisor and the developers, as well as the future users and customers for the system.
B. Goal Model

The goal model (Fig. 4) provides an overview of objectives and goals for the system. There are three types of goals: business goals depicted in purple, left side of Fig. 4, usage goals in green, top-right of Fig. 4, and system goals in red, on the lower right side of Fig. 4.

C. Context and System Vision

The context and system vision (Fig. 5) give an overview of the most important elements of the business and operational context of the system as well as the core features of the Plant Guild Composer.

D. Constraints

The constraints (Fig. 6) provide the most important restrictions by the rules of the local housing authority that have to be adhered to and influence the design of the plant guilds. These have to be taken into account for design constraints during the modeling of use cases and scenarios.

E. Further Development

These models and further artifacts will be extended over the duration of the ongoing field study and the design of the system.

V. PLANT GUILD COMPOSER - A VISION

Long before you sat down on the bamboo bench in your backyard, you faced the challenge of creating the plant guild. When you decided to transition to a simple, sustainable lifestyle you weren't sure where to start. Then you were introduced to the Plant Guild Composer, the app that helps you grow a self-sustaining garden. The steep expense of purchasing fresh food, and the deaths of your late potted plants, motivated you to give plant guilds a try.

A. Location Requirements

The application first asked for your address. Once entered, a diagram of your lot containing your house’s location was presented. It then asked you to confirm or edit the diagram. You went outside and walked around the house to make sure the PGC didn’t miss anything, and spotted the utility connections under the Live Oak canopy in the front yard. At this time you also took note of high and low-lying areas, points of erosion, and soil properties. After you completed your property’s diagram, the program asked for your priorities:
food, natural medicine, building materials, household items, wildlife sightings, environmental restoration. You chose food as the primary function and decided that growing building materials and household items would also be useful on site. Then you specified cost of implementation in terms of time, money, and resources.

B. Plant Guild Requirements

From there you specified your primary food requirements from a generated list of plants that grow in your climate and location. First, you investigated the fruits and noticed a native variety of persimmon, a fruit you used to buy. When you chose to include the persimmon tree, a simulation showed where it could be planted on your property. The PGC displayed suggestions for plants, relative to your goals, that could provide the persimmon tree with its needs. It took into account that your soil was alkaline (pH) and persimmon prefers a neutral range. The PGC suggested perennial peanut, a low-growing legume that makes nitrogen more available for the persimmon. While you continued choosing plants based upon the PGC’s suggestions, the simulation continued updating in size, arrangement, and location. You realized you had wanted to include another plant. It was no longer eligible and the system told you why (too expensive, not compatible with guild optimal location), so you removed some plants that were less important to add it instead. You proceeded with this until the guild reached a closed system, (i.e., each guild member’s needs is supported by the other members and the environment) and your desires were met.

C. Result: Design Layout

Once the design was finalized, the PGC produced the guild layout and location in your yard, the places to obtain the plants, tools, and compost within your budget, and instructions on how to implement the guild. Two weekends later a mound with thirty young plants setting root was in your backyard. It had only taken you one morning to get the free municipal compost, a day to acquire the plants and about 2 hours to put them in the ground with two friends. You watered the guild a couple times a week at first, then about once a week when they started really growing, until you found they no longer needed watering. In a few weeks fresh herbs spiced your dinner, in one season the fragrance of flowers was in the air, and shortly after your first crops were harvested. Now, less than a year later, you obtain most of your food, and even some building materials and cleaning supplies, from your backyard. You don’t worry about going to the store in rush hour traffic or minimizing expenses so you can eat healthy. Life really did become simpler with the domestic plant guild.

VI. DISCUSSION

Our PGC vision is an example of an application designed with a balance of user convenience, awareness of user role, and environmental sustainability.

A. Reduction of Complexity

We believe that the PGC has the potential to reduce the complexity of transitioning from our modern, complex, consumer lifestyle to one that is simple and sustainable. We described how the transition to a simple, sustainable lifestyle can be obstructed by up front complexity with the Domestic
Plant Guild example. We believe there are many human sustainable systems, especially in permaculture, that also have this complexity challenge.

Designing earthworks for water collection requires intimate knowledge of natural water flow through land. Building an off-the-grid house requires extensive knowledge of the many ways energy can be produced. We’ve suggested masking the complexity by finding the point of greatest convenience that still produces a sustainable system. We believe the RE community should be researching how to enable people to utilize sustainable human systems without being bogged down by the complexity of learning how to get started; it is a contribution RE can provide in the world’s movement towards simple, sustainable living.

B. Conveying Sufficient Knowledge

Our vision and requirements feature the idea of reducing complexities to the threshold where users still have the opportunity to learn at a higher, more leisurely level. We also believe that by removing all the complexities, the user wouldn’t have the opportunity to understand, utilize, and appreciate the support the system provides (i.e., awareness of user role). We feel that the awareness challenge complements the convenience challenge and that a single solution can be found for both, although it will vary for each sustainable human system.

C. Providing Access

Accessibility to information is key in achieving the balance of convenience, awareness, and sustainability. It is essential to explore ways to gather information that can’t be acquired from public resources and are too complicated for average users to acquire. Methods that should be researched include: gathering and analyzing information produced by experts and power users into designs, using mobile technologies to aid the user in a more detailed analysis of the environment, and unconventionally utilizing common household items to indicate otherwise elusive properties of the environment.

VII. Conclusion

In this paper we described the need to develop technologies that enable people to transition into a simple, sustainable lifestyle. We introduced the concept of a domestic plant guild to show how sustainable human systems can effectively support such a lifestyle. We presented a series of requirements for building the Plant Guild Composer, a tool to develop plant guilds.

The requirements suggest the incorporation of requirements analysis techniques to reduce the complexities associated with its creation. Without these techniques, the Plant Guild Composer would only be usable by permaculture and horticulture experts. We feel that this research area needs further exploration and that our requirements could be applied to tools that support a user’s journey towards a simple, sustainable lifestyle.

Future Work

Right now we are working on the first prototypes of the Plant Guild Composer. A screenshot is provided in Figure 7.

ACKNOWLEDGMENTS

We would like to thank Kristina Richards at the UCF Arboretum for her guidance and support of this work, and those who provided constructive feedback on early versions of this paper.

This material is based in part upon work supported by the National Science Foundation under Grant No. 0644415.
REFERENCES


18 An Assessment Technique for Sustainability: Applying the IMAGINE Approach to Software Systems
An Assessment Technique for Sustainability: Applying the IMAGINE Approach to Software Systems

Alejandra Rodriguez
Software & Systems Engineering
Technische Universität München
Munich, Germany
alejandra.rodriguez@mytum.de

Birgit Penzenstadler
Donal Bren School of Informatics and Computer Science
University of California, Irvine
Irvine, CA, USA
bpenzens@uci.edu

Abstract—Sustainability is a concept for which exist many definitions, but most of them are either vague or too limited; no consensus has been achieved. One pragmatic solution is to provide an implicit definition by setting up a standard with criteria that have to be fulfilled, e.g., by a company in order to reach their self-set sustainability goals.

The problem is that even with a defined goal or strategy for sustainability, practitioners lack of reference frameworks to align to and from there to derive concrete objectives and activities. Consequently, it is hard to implement such goals or strategies through the use of IT.

This paper presents the application of the IMAGINE approach [5], for analyzing and assessing sustainability, on a system supported by IT and software systems. The approach was implemented and its applicability assessed in an industrial case study in a Master student’s research project.

The approach provides project managers, business analysts, and requirements engineers with the capability to devise a specific strategy for particular contexts and sustainability goals.

I. INTRODUCTION: CONTEXT & PROBLEM

Sustainability standards are used to assess companies interested in validating and certifying their products, projects, or development and management practices. They consist of norms and reference criteria related to ideas that pursue sustainability, and the assessment is usually performed by a third party.

Several standards that are currently available focus on sustainable development [1], [2], but we can also find standards for sustainability reporting [9], [10], or sustainable design [7], [3], among others. Furthermore these standards can be segmented in the industry-specific sectors, like those listed by the Industry Classification Benchmark [8], for instance food, oil and gas producers, mining, transportation, healthcare, and telecommunications.

A. Problem

None of these standards is designed for the application in all contexts, and considering all sustainability dimensions. Norms like the ISO 14000 for Environmental Management, cover the environment dimension but do not give guidance for software systems. Besides, the software systems and IT supporting such developments are often obviated from the lifecycle analysis of a sustainable development project or not clearly indicated in the analysis. Especially for software systems, sustainability standards are not yet available and the majority of research focuses on green IT, energy-efficient software, and human computer interaction [11], [12]. Consequently, practitioners in software development lack a guideline for assessing sustainability in their systems.

B. Contribution

We provide the results of a case study in applying the IMAGINE approach [5], an analysis approach from the domain of sustainable development, to a software-intensive system that exhibits a significant impact on the sustainability of city mobility. It has been adapted to include the role of technology into the analysis, and to inspect in depth the rationale, drivers and impact of the planned solution. It can be applied from the early phases of idea definition up to project finalisation, production and maintenance.

II. BACKGROUND: IMAGINE AND DRIVE NOW

This section gives a short introduction to the IMAGINE approach and background on the DriveNow case study.

A. The IMAGINE Approach

The IMAGINE approach [5] originates from the environmental studies field and applies systems thinking principles [6].

It was designed to guarantee the cooperation of users, experts in all levels, performers and public representatives, through the identification and understanding of problems of sustainable development, definition of optimal indicators to measure success, and to develop on decisions about further development, to conclude with the activities for achieving the desired scenarios.
Thus the application of IMAGINE involves the use of participatory techniques, the inclusion of varied groups of stakeholders, the identification of meaningful and relevant indicators for the groups, and the use of scenarios for current and future states. This enables the tracking of the evolutionary behavior, the recognition of deviations from the goal to apply timely corrections. It has only been applied so far in environmental systems.

The IMAGINE approach is carried out in five steps:

1) Understanding the context: identify the stakeholders, their perspectives, and scope the system to be assessed.
2) Agreeing on Sustainability Indicators (SI) and bands of equilibrium: identify relevant indicators for each group of stakeholders, agree in a common set for the whole system, and establish reference boundaries within which each SI remains sustainable.
3) AMOeba scenario making: develop the desired future situations of the system in terms of the selected SI to measure and depict it in an AMOeba diagram.
4) Review and Metascenario making: review the status of the system conducting the whole process again and contrasting the scenarios and diagrams over time.
5) Publicity and Marketing the message: end of an Imagine iteration where the outcomes are publicized among potential consumers of the information.

For each one of the steps a variety of instruments to ease the adoption in companies is available in [14], for example: Controlling, Corporate Social Accounting, Corporate Volunteering, Cross-impact analysis, Dialog instruments, Eco-design/Design for environment, Environmental Shareholder Value, Mission Statement, Reporting, Scenario analysis, Supply Change Management, Sustainability Balanced Scorecard, and Total Quality Management.

B. The DriveNow Case Study

Our industrial case study is developed for a car sharing system deployed in 2011 in three major German cities by BMW, Mini, and Sixt in a 50%-50% venture. The project concept was designed to provide new mobility services that are individually attractive and socially sustainable.

The business model of DriveNow presents the rent of premium vehicles for a short period of time within the city using public parking areas inside an established perimeter without incurring in additional parking costs.

We chose DriveNow for our case study as it is been marketed as a positive contributor towards sustainability and environmental protection. Among their main goals are the reduction of CO2-emissions by: replacing old private cars for new shared cars, integrating new technologies and introducing electronic cars, and reducing the number of cars with only one passenger by encouraging car pooling.

The project involves automobiles and technology for efficiency, care-hire know-how, IT systems and a comprehensive customer registration and interaction network. The whole lifecycle is highly dependent on technology, from the development of the equipment and software, the infrastructure, disposition and maintenance of registration, authentication and interaction platforms between the provider, partners, users and members of the community, and the creation of startups extending the service.

The initial idea of modern mobility services was integrated with environmental focuses, engaging the project in sustainability initiatives.

C. Outline

The remainder of this paper describes the application of the IMAGINE steps for the case study of the DriveNow system. Section III describes how the system of interest is scoped, Section IV presents the process of selection of sustainability indicators (SIs) and definition of bands of equilibrium for these indicators, Section V explains the development of future scenarios and the AMOeba diagram, Section VI discusses some open issues and limitations. Section VII concludes with open issues and suggestions for future work.

III. UNDERSTANDING THE CONTEXT

The assessment begins with the comprehension and delimitation of the system or project of interest, and the context where it is applied. This first step is crucial to obtain a well defined problem and a precondition for a successful assessment.

A. Stakeholder Perspectives

The system of interest is scoped from four main perspectives as illustrated in Fig. 1, by directly gathering information from relevant stakeholders and available documentation. The four perspectives are formed by individual representatives of stakeholders under the roles of owners (customers), implementers (developers), beneficiaries (users), and regulators (government, legislation).

Figure 2 shows the results of the stakeholder analysis for all four perspectives in DriveNow. The owners are BMW, Sixt, and Stattauto, and the regulators are the government, certifying organizations, and controlling agencies like the police. The beneficiaries are the drivers, the community, and friends, and the implementers include the whole development process as well as marketing and additional service providers.
B. Driver-Pressure-State-Impact-Response Approach

The information can be gathered by any participatory technique, using the Driver-Pressure-State-Impact-Response (DPSIR) approach [4] shown in Fig. 3. The DPSIR seeks to identify the Drivers to design the system, the Pressures to use unsustainable products or practices, what aspects of the current State might seem affected by the introduction of the system, which Impact and level of severity is expected, and what are the Responses of the environment and users to the system regarding sustainability. The most common technique in the early steps of the analysis are structured interviews. As major result after the consolidation of the interview data, the main objectives of the system and the assumptions made are identified and concisely stated. A graphical overview of the DPSIR analysis approach is provided in Fig. 4. For the complete set of DPSIR indicators in DriveNow, see [13, p. 25-29].

C. Root Definitions

For a succinct statement of the result we use Root Definitions. A Root Definition is a structured description of a system and a clear statement of activities which (might) take place in the context of our system. A properly structured root definition comprises three elements: What the aim of the system is, How that aim is to be achieved, and Why the activity is carried out w.r.t. a long-term aim. This is stated as "A System to do W, by means of H, in order to achieve Y".

The root definitions elaborated for DriveNow were:

1) The Car Sharing Project focused on private users that do not own a car, and realized by the implementers, in order to establish the brand as a mobility service provider, while removing old cars from the streets, assuming behavioral patterns, government support and managing feasibility, capacity of production, offer and demand, prices and easiness of use.

2) The Car Sharing Project focused on offering community members that do not own a car, in order to provide a support and convenience when needing a car for occasional use, while involving them into the membership and
maintaining the initiative sustainable without profiting, assuming behavioral patterns, government support and managing prices and schedules of use.

D. Data Collection in Interviews

The interviews to collect the information in our case study were conducted with three representatives for the groups of Owners, Beneficiaries and Regulators. The implementers’ point of view was partially covered by the representative of the owners. Once the system and context are clearly defined, the next step is the selection of reference measures and the establishment of sustainability criteria for each one of them.

The full documentation of the case study is accessible as Technical Report [13].

IV. AGREEING ON SUSTAINABILITY INDICATORS AND BANDS OF EQUILIBRIUM

![Sustainability Indicator Catalogue (Excerpt)](image)

There is no general consensus on the concept of sustainability; the definition of sustainability varies from company to company and from person to person, i.e. a precise conception of sustainability varies depending upon who is using it and in which context [5, p. 28]. Hence flexibility on the selection of important measurements is needed, without losing standardization and the comparison capability among companies.

A. Catalogue of Sustainability Indicators

A general catalogue of sustainability indicators (SIs) is therefore employed here and only those relevant and suitable for the context are pre-selected. They are prioritized in a subsequent step by simultaneously looking at the priority assigned for each group of stakeholders.

![Identified Topics in DriveNow (Excerpt)](image)

The catalogue is part of our research results and was created based on general indicators and extensions for which we could find official measurement values. The catalogue is structured into groups, with corresponding standard themes, sub-themes, a list and a description of each indicator; Fig. 5 shows a fragment. For the full catalogue, please refer to [13].

B. Prioritization of Concerns

The final selection of SIs from the set of pre-selected ones is performed by a multi-dimensional stakeholder prioritization of concerns, here the most relevant SIs for each stakeholder are contrasted with the other stakeholders. The contrast is graphically depicted in a 2x2 matrix were each SI is assigned a point in the grid according to the relevance for each pair of stakeholders.

Next, the SIs are grouped into topics, subtopics and finally listed individually. Our extension takes as basis the three dimensions of the Triple Bottom Line, respectively the Environmental, Social, and Economic perspective as topics, and adds two more, namely the Human and Technology dimensions. The Human dimension associates the SIs pertinent to individuals, contrary to the social dimension that refers to the society collectively. The Technology dimension makes reference to the technological infrastructure supporting the different tasks over the lifecycle, as well as technology capacity limits, availability and access to technology based on demographics, extension and integration of additional services, and communication mechanisms enabled by technology. The structure is, hence, based on the five topics of our extended approach, i.e. environmental, social, economic, human and technology, each with subtopics and a list of selected SIs to be measured.

C. Bands of Equilibrium

With the priority SIs selected, we define a band of equilibrium describing the boundaries within which our SIs values must stay. This band is determined according to the selected measurement unit and method, and is given by two values, one for the minimum value our SI can have such that it is still sustainable (any value lying below is unsustainable by lack), a second value for the maximum value of our SI to still be sustainable (consequently, any value above is unsustainable by excess).

For DriveNow, a general catalogue of SI indicators was developed, and a sub-set of them was selected and structured based on the information gathered in the previous step [13, p. 37-47]. The bands of equilibrium were defined using values found in standards, regulations and publications, applicable to the DriveNow project.

At this point we have the foundations for the elicitation of current and future scenarios. The next step evaluates the weaknesses and strengths of the current status of the system, the potential for improvement of particular SIs, as well as the overall improvement. It also gives a suggestion on the prioritised corrective actions to take.
V. AMOEBA AND SCENARIO MAKING

In this third step we start with the current situation appraisal of the system and the definition of the future scenarios we want to attain with the time. A consolidation of the previous conducted steps can be summarized in the future scenarios and AMOEBA diagram. By obtaining a current measure of the system for each SI, and defining the values we ideally expect to reach at different future points in time, we obtain the current and goal values for each scenario. For simplicity in this paper we only consider one future scenario, therefore only a goal value.

A. Data Collection for Values

The current and goal values for each SI were obtained from official concept descriptions, sustainability reports, press publications, public statistical data from the city and government, market analyses, and other documentation. The measurement data from all sources were obtained by experts in the field, in some cases with the aid of specialized equipment (e.g. sensors measuring air pollution) and in general being reliable data.

B. Future Scenarios

The scenarios are depicted considering the four values determined for each one of the priority SIs, the two values of the band of equilibrium established in (Step 2), the value of the current measurement and the value for the goal in the future, all four for one scenario. These are plotted in an AMOEBA diagram, for each defined future scenario, see Fig. 7.

The graphical representation of the future scenarios in an AMOEBA diagram enables the visual identification of SIs that are lying outside the band of equilibrium either by exceeding the maximum sustainable, or by not reaching the minimum sustainable limit, and those SIs closer or further from the goal value. By overlapping these four scenarios global insights can be gained about the system, its sustainability level, its weaknesses and strengths. An AMOEBA with too many ‘teeth’ will indicate several weaknesses on particular SIs in comparison to the level of the other SIs, on the contrary a more circular amoeba is an indication of equally evolved SIs as far as they lie within the band of equilibrium.

The primary corrective actions must be those that accomplish a sustainable and effective positive reaction without negative effects, such that a more sustainable current situation of the system can be achieved in a short time. A more sustainable situation is characterized by an AMOEBA diagram where no SI lies outside boundaries, and ideally all of them are close to the goal value.

C. Analysis According to Sustainability Dimensions

An additional element of the assessment is the balance of SIs belonging to each one of the five topics of our approach, see Fig. 8. The amount of SIs belonging to a certain topic are grouped together, plotted and the whole topic highlighted, building a color coding of five shades.

In the AMOEBA, we can observe the general balance of the system, depicting the priority SIs for multiple stakeholders in parallel. In our example (see Fig. 7) a shadow color is assigned to each dimension (environmental, economic, social, human, and technology), the SIs belonging are plotted close to each other, and then highlighted with a triangular surface.

D. Current Challenges in DriveNow

For readability, only a sub-set of indicators was selected for the AMOEBA diagrams presented here. Since the project was launched only one year ago, some SI values are undefined or did not change with respect to the initial scenario. However, Fig. 8 clearly depicts the challenges that DriveNow is currently facing:

- With regard to the environmental aspect, the number of cars that could be saved still has to increase.
- In the technological sector, there are some availability issues for system improvements that have yet to be solved.
For the economic perspective, the market acceptance is currently too low and needs to be improved.

As the analysis can only provide insights into misalignments with goals, the respective actions to be taken to solve the challenges are subject to individual efforts at BMW and Sixt, but we could provide them with a concise overview of the current status of the project with regard to the sustainability dimensions.

VI. DISCUSSION

We are aware that the application of the IMAGINE approach [5] in this setting is not in an application domain originally intended by its inventor. This research had the goal to investigate the applicability and usefulness of an approach from within classical sustainability research to requirements engineering for software-intensive systems.

A. Informal Evaluation

There is no formal evaluation possible as there is no data officially available that would have all the information gathered using the IMAGINE approach. However, the feedback by our industrial partner indicated that the analysis results included not only a summary of the information they had originally elaborated when gathering scope and requirements of the project, but also held some new aspects due to its completeness and integrity.

B. Assessment of the AMOEBA Diagrams

From the AMOEBA diagram we can determine where flaws occur, distinguish the issue areas, and draw some conclusions about the corrective actions. Further investigation and feasibility analysis can be performed to ensure that sufficiently informed decisions are made.

A sustainable system in the AMOEBA diagram should have all the colors in an equal proportion. Whenever a topic is left unattended, this is an indication for the inclusion of additional SIs in that specific topic, it is mainly achieved through strategy revision and sensibilization. The topics with a high proportion of unsustainable SIs in the current scenario implicitly advise future steps to achieve the desired scenario and the main points to invest.

After some corrective actions have been applied and a reasonable time has passed, new data must be collected, and the results of the whole process must be revised and adjusted.

C. Assessment of the DriveNow Case

The DriveNow project was originally designed to encompass mainly economic and social aspects. The introduction of the environmental facet has not shown positive results yet, since it takes long time to exhibit changes. Moreover, the results have been affected by the plan of future important contributions to the environment as is the introduction of electronic cars. The assessment presented a current overview of the project, revealed relevant faults regarding environment, the achievement of goals, and promising results with respect to
the social and economic aspects. It provides a basis for review and forthcoming analyses.

VII. Conclusion

The extension of the IMAGINE approach, an encompassing analysis approach from the sustainability science domain, has been successfully applied in an industrial case study with a system that has been online for a year, supported by IT and with a focus on sustainability in its roots. The developed indicator catalogue is available for use in other assessments in related application areas.

A. Summary

The first two steps are mainly used to scope the right problem and the participation of several stakeholders from each group enriches the result. The third step provides succinct information and an overview of the current system status, as well as it provides insights and indications of strengths, weaknesses and potential for improvement. The fourth step is there to enable flexibility and make possible an evolutionary assessment over time. This information can later be used for informed and precise decision making relating to sustainability matters.

B. Benefits

Our experience with the IMAGINE approach with regard to requirements engineering practices exhibits improvements in the elicitation process. The identification of stakeholders in different areas and at different hierarchy levels, including government and certifying organizations, sustainability experts, potential and real partners, as well as market factors, endorsed the balance regarding prioritization and completeness of the elicited requirements.

The formulation of root definitions helped to state purposes clearly and to define a more sharp scope. The SIs help to state finer measurable objectives, as well as to convey long and short term goals. The AMOEBAs diagrams depict information of different scenarios in a manageable and compact way, understandable for the participants, and useful for the assessment over time. The flexibility of the approach regarding the instruments that can be used in the different business areas eases and assist the adoption into companies.

C. Assessment

The take-away message is that it is worth looking over the rim of one’s teacup and evaluate the use of techniques of related domains in a different setting. It is unlikely that approaches are applicable one-to-one, but it is very likely that it can be adapted in a way that contributes more than developing new techniques from scratch. In the case at hand, the study brought new insights for the system and a means to perform an encompassing analysis that will be reused in the future evolution of the project.

D. Future Work

We intend to develop a toolset to support the usage of the IMAGINE approach, which eases the current and historic information management for a system under consideration, and the review step in posterior revisions. This work can be integrated to be part of a broader sustainability quality model and established as a state of the practice standard for assessing sustainability in any context.

Acknowledgment

We would like to thank our industry partners from BMW for providing us with the necessary input to perform the study and for giving feedback on the analysis results. This work is part of the EnviroSiSe project (grant number PE2044/1-1) funded by the DFG in Germany.

References

19 Supporting Physicians by RE4S—Evaluating Requirements Engineering for Sustainability in the Medical Domain
Supporting Physicians by RE4S
Evaluating Requirements Engineering for Sustainability in the Medical Domain

Joseph Mehrabi, Birgit Penzenstadler, Debra Richardson
Department of Informatics, Donald Bren School of Computer Science
University of California, Irvine
Irvine, CA
{jmehrabi|bpenzens|djr}@uci.edu

Abstract— Sustainable development applied to software engineering means developing systems in a sustainable way (domain-independent), as well as supporting sustainability in the application domain (domain-dependent). Developing sustainable software begins with requirements engineering (RE), which answers the general question, “What should this software system do?” RE for sustainability (RE4S) outlines the process of RE while taking steps to make the system more sustainable. This process answers the question, “What should this software system do, while contributing to overall sustainability?”

This paper considers how sustainability impacts the requirements engineering process for a software system for medication adherence. The examined system, Project Cognatio, is targeted toward outpatient medical practices with the goal to improve medication adherence among patients. We found considering the dimensions of sustainability can significantly enrich a number of RE artifacts. This analysis proves useful for the future development of Project Cognatio and shows that RE4S can improve software development’s contribution to sustainability.

Index Terms—requirements engineering, sustainability, medication adherence, evaluation, stakeholder, artifact

I. INTRODUCTION

Over the past century, human innovations have started to erode certain aspects of our planet. But for the past few decades, industries, corporations, and governments have started taking steps to be more sustainable and maintain resources for future generations. One way they have been aiming to do so is through sustainable development. Sustainable development highlights the ability for the present generation to meet its own needs without jeopardizing the ability for future generations to do the same [1]. Technology has the ability to impact different dimensions of sustainability. We consider the following dimensions [2]:

• **Human sustainability**: the maintenance of human capital, for example in form of health, happiness, education, etc. over the course of a person’s lifetime. For software systems, this is specifically relevant to analyze for the users, but also for the developers.

• **Social sustainability**: the maintenance of social capital and services and preserving the societal communities in their solidarity. For software systems, this is an important analysis aspect with regard to the user communities.

• **Economic sustainability**: the maintenance of assets in the form of capital or added value. This relates to the software developers, whether companies or free open source.

• **Environmental sustainability**: the preservation and protection of the environment when utilizing its resources, which every software system does in direct (energy) and indirect (other resources) ways.

• **Technical sustainability**: the improvement and maintenance of the longevity of systems in a rapidly changing technological environment.

Technology depends on its systems, which also contributes to each dimension of sustainability. To develop software, one follows the software engineering process: (1) requirements engineering (RE), (2) design and implementation, (3) testing for fulfillment of requirements, and (4) deployment, maintenance, and enhancement. To develop what we understand as sustainable software [3], certain alterations need to be made to the software engineering process.

The research objective of this study is to analyze how the RE process would be altered in considering all five dimensions of sustainability, during what we have previously described as RE for sustainability (RE4S) [3]. Standard RE is defined as the elicitation and specification of a software system’s requirements that are agreed upon by all stakeholders, where the specified requirements must be addressed and adhered to throughout the entire software engineering process [4]. RE answers the general question of what a software system should do. RE4S not only answers the same question but also asks and answers how the system contributes to overall sustainability.

**Contribution:** We contribute a case study from the medical domain that shows how considering sustainability during RE can improve our understanding of the encompassing impacts of software systems on its surroundings. We use the artifact-oriented approach to RE4S (as described in [3]) as validation study to evaluate its usefulness in a not yet explored application domain.

**Outline:** We provide the background on related work, the methods used in this study and the project context, an excerpt of the requirements specification conducted for the study system, the discussion of the results, and conclusions for the evaluation and for future work.
II. BACKGROUND

For an understanding of the context of this work, we describe related efforts, the methods in use for this study, and the software system under consideration. The context of the study outlines the causes of the pursuit toward this research.

A. Related Work

There is a limited amount of related work for guiding requirements engineers towards more sustainability in software development practice. The workshop series on Requirements Engineering for Sustainable Systems (RE4SuSy) [12][13] offers a number of contributions on aspects like goal modeling, energy saving, complexity, sustainability-enhancing application domains, user participation, quality, and eco-aware design. Mahaux et al. [14] present a case study on a business information system for an event management agency and assessed how well some RE techniques support modeling of specific sustainability requirements.

In contrast, our aim is to provide modeling means for integrating sustainability into any software system as a major objective. This has been illustrated in [3], where the RE4S approach used in this paper is described in full detail, using a car sharing case study as running example. The approach has been preliminary evaluated in case studies on a permaculture software application called the Plant Guild Composer [15] and on the collaborative drawing tool Calico [16].

B. Materials and Methods

The elicitation of requirements was performed by interviewing a number of stakeholders, namely a doctor and other members of his medical practice. The RE4S approach [3] is a checklist and reference model based approach that demonstrates how to include the objective of environmental sustainability from the very early steps in finding the stakeholders and analyzing the domain, to the definition of a usage model and specific requirements. The documentation is carried out using an artifact-oriented approach that follows the Artifact Model for Domain-independent Requirements Engineering (AMDiRE) [5].

Each modeled artifact was developed using Cacoo, a web-based diagramming tool, and Microsoft Office for other documentation. Each artifact was reviewed by requirements engineering experts, Dr. Debra Richardson and Dr. Birgit Penzenstadler, a sustainability expert, Dr. William Tomlinson, and a medical doctor, Dr. Don Mehrabi, to verify that the requirements were coherent and consistent with the desired system. They were then altered and amplified to fit the desired requirements and specific software engineering standards.

C. Project Cognatio

1) Description and Purpose: Project Cognatio is a two-tier system that relies on a mobile application to be used by medical patients and every day consumers and a desktop application to be used by medical practitioners. The overall goal of this system is to improve medication adherence among patients.

Medication adherence is a common problem among patients who do not feel it is entirely necessary and/or sometimes forget to take their medication as prescribed [6]. There are many factors that cause this, but the aim of Project Cognatio is to reduce the impact of those factors and encourage patients to better adhere to their prescriptions. The system depends on collaboration between employees of the medical practice and the patients themselves to achieve the desired goal.

2) Features: The central feature of this system is its ability to exchange medication information between the mobile application and the desktop application via the Internet. The mobile application relays patient activity to the desktop application that will be viewable by members of the medical practice. The desktop application keeps the prescriptions on the mobile application updated so the patient understands which medications to take and when to take them.

On the side of the mobile application, the user will have the ability to send usage reports of medications to his or her doctor to confirm that he or she has consumed the medication at a certain time. The patient also has the ability to send user-typed reports of possible side effects that would be caused by the medication. Because this is an Internet-enabled function, if there is no Internet connection, the data will be stored until a solid Internet connection is established, and then it will be deleted once it is successfully sent. The graphic user interface of the application will be designed to make reporting of medication usage will be easy.

All members of the medical practice will have access to functions on the desktop application specific to their role in the practice. The desktop application will be proprietary for each individual practice that adopts the system, so it will be custom and accommodate for any special characteristics of the practice. With this application, verified and medically licensed individuals like doctors and physician assistants will be able to view, assign, and update prescriptions for patients in the system. Secretaries and other unlicensed individuals will only have the ability to view patient activity and send reminders to patients who have not taken their medication as directed or if their activity on the application has been idle.

All of these functions and the participation of both parties will encourage a higher level of medication adherence among patients.

3) Context of the Study: The first author was given the opportunity to participate in the Summer Undergraduate Research Fellowship in Information Technology (SURF-IT) at UC Irvine in 2014. With his focus on premedical sciences and ambitions to attend medical school, he spoke with an outpatient dermatologist, Dr. Don Mehrabi, and they came up with the idea for a system aimed to improve medication adherence among his patients.

They met regularly to elicit and specify the requirements of the system. The artifacts were laid out in this document to earn Dr. Mehrabi’s approval on the goals and vision of the system. In addition to regular meetings with Dr. Mehrabi as the client, the first author also regularly met with Dr. Penzenstadler and Dr. Richardson to discuss the progress of the project and for feedback on the artifacts.
III. REQUIREMENTS SPECIFICATION FOR PROJECT COGNATIO

This section provides an excerpt from the requirements specification as documented for the medication adherence app; including the (reduced set of) AMDiRE artifacts. The full specification is available in a Technical Report [18].

A. Business Case Analysis for Project Cognatio

The business case analysis for the medical adherence software system follows the template according to the case method at the Harvard Business School [17].

1) Problem: There remains a real discord between doctors and their treatments of patients. Doctors do not have control of patient treatments on a regular basis to the extent that patients will adhere to their regimens with certainty. Due to a patient’s regular lack of time, the cost of seeing a doctor, and his laziness, he may not adhere to his medication, he may feel no urgency to contact his doctors about problems with the medication, and he may not follow up with them regarding treatment [6]. To some degree, the relationship between the doctor and the patient is a brief, impersonal one that only exists during a regular, scheduled appointment, impairing the implied requirement for patients to adhere to their medication. As a result, patients may not be compelled to follow what is called “medication adherence,” the regular use of medication as prescribed by a medical professional to treat a patient. This can lead to worse conditions and higher costs for healthcare.

2) Analysis: There is documented evidence that patients do not necessarily follow doctor’s orders regularly. According to Figure 1, when the daily dosage of a patient’s medication is higher, the patient becomes less likely to follow through with the treatment [7]. This becomes the case when patients did not receive any sort of reminder or reinforcement to take their medication.

Patients also have many reasons for not wanting to adhere to their treatments. According to Figure 2, reasons for medication non-adherence include forgetfulness, lack of doctor concern, poor physician relationship, and side effects from the drugs [8].

3) Solution Options & Cost-Benefit Analysis: A simple, technical solution could improve medication adherence among patients. The most effective tool would be an improvement to the relationship between the patient and the doctor.

a) Mobile Application

One solution involves the creation of a connected mobile app that ties together patient and doctor throughout the patient’s treatment. From a functional standpoint, the

Figure 1: Adherence to medication according to frequency of doses [7].

Figure 2: A distribution of the most common reasons for medication non-adherence

application would improve the communication between the doctor’s practice and the patients, making medication non-adherence less of a problem. Patients would take their treatments more seriously, and as a result, avoid higher health costs from worsening conditions in the future. This could significantly improve the overall sustainability of the healthcare system.

b) Personal Contact

Another solution would involve more personal contact between the doctor’s office and patients by any medium including phone calls, emails, text messages, and regular doctor visits. A doctor would not appreciate additional contact from patients via telecommunications because of a lack of time. If personal contact is a result of constant appointments, patients will incur high costs to directly visit the doctor. While this may be a benefit to the doctor, a practice generally focuses on treating as many patients as possible, and may hinder this goal of the practice.

c) Do Nothing

The final option would be simply do nothing in response to the medication non-adherence problem. Medication non-adherence will continue to be a problem. If a patient wants questions answered about side effects or any other parts of the treatment, he will need to visit the doctor’s office, using gas and increasing carbon emissions. Therefore, this option is not favorable with regard to the sustainability of our health system.

4) Recommendations: After assessing the three options, it would be best to develop a mobile application that will more closely connect a patient to the doctor and medical practice. It would be recommended that physician assistants and secretaries interact directly with patients via the app, while the doctor should only be utilized for cases that are extreme or cannot be handled by such personnel.

Requirements and the design for the functions of the application should be developed in collaboration with the physician, medical practice staff, and test patients.

5) Project Description: In consultation with the stakeholders, it was decided that the most effective solution
would be a tool to influence medication adherence and a closer relationship with a patient’s physician would be through a mobile app to automate the communication process between the physician and the doctor while still maintaining the personal relationship. A patient would document when and if he followed the treatment regimen that day, and the physician, or office assistants, would monitor the patient and his or her treatment. If any irregularities in the regimen erupt, the patient would receive a notification from the doctor’s office via the application to remind the patient to regularly use the treatment as needed. The doctor then would be able to respond to these notifications in his or her own way.

A system described as the one above would eliminate the patient’s forgetfulness as a factor of medication non-adherence and hopefully improve the patient’s relationship with his doctor.

6) Tentative Project Timeline: The design of the system could be finished at the end of November to make way for the beginning of its implementation later that month. The system could first be developed into a mobile application for the patient and web application for the medical users. Most of the time was allotted to the implementation because we expect a number of prototypes of this system to be developed. By the summer of 2015, the application will be fully developed and available to more medical practices for its use.

7) Sustainability: This system will contribute to all five dimensions of sustainability through first, second, or third order effects as described in [19]. Direct, first order effects are the immediate opportunities and effects created by the physical existence of software technology and the processes involved in its design and production. Indirect, second order effects are the opportunities and effects arising from the application and usage of software. Systemic, third order effects, finally, are the effects and opportunities that are caused by large numbers of people using software over time [20].

Human Sustainability: The system is designed to promote the health of those who receive treatments from their doctors. The mobile application will help patients adhere to the use of their medications to improve their health further. Because this system is aimed toward healthcare, improving Human sustainability is the primary focus of the system.

Economic Sustainability: Adherence to medical treatments will prevent patients from encountering even worse health problems, following up with their doctors, or attending hospitals. Each of these situations raise healthcare costs of the patient and others in the United States. This is a second order effect.

Environmental Sustainability: The mobile application is designed to have the patient adhere more closely to his treatments in order to prevent follow-ups, which entail transportation to the doctor’s office possibly in the form of driving. The reduction in the needs to drive decreases carbon emissions and gasoline consumption. Closer adherence to the provided medication will also reduce the need to dispose of any unconsumed medication into the environment. Those are second order effects as well.

Technical Stability: The system should be available on a wide variety of platforms, making it compatible with many mobile and desktop devices, reducing the need for users of the application to dispose of their older technologies to utilize this application.

Social Sustainability: A decrease in the need to drive to the doctor’s office will reduce traffic on the roads, allowing other drivers to get to their destinations more quickly.

B. Stakeholder Model

The purpose of the stakeholder model is to depict the stakeholders involved in the development and operation of the system and how they are related to the system. In the model in Figure 3, there are ten stakeholders. Each stakeholder either has a direct or an indirect relationship with the development of the system. A direct relationship includes having a primary interaction with the development or usage of the system. An indirect relationship is applied to those stakeholders who are not primary actors in the development or usage of the system. The stakeholder matrix in [18] contains a more exhaustive description of each stakeholder involved in the development of the system.

The Stakeholder Model provides a basis for the System Vision (see section III.D).

![Figure 3: Stakeholder Model](image)

C. Goal Model

The purpose of the goal model (Figure 4) is to outline the goals and subgoals that the system will achieve and the goals to achieve proper development of the system. The main goal of this system is to improve the five dimensions of sustainability. All five dimensions can be achieved through the development, implementation, deployment, and successful usage of this system, which aims to improve medication adherence. A specific illustration of sustainability concerns in
the goal model allows for showing the relations between the different dimensions (upper part of the goal model in Figure 4 shows the sustainability goals) and how they can be considered in the system goals (lower part shows the system goals). The most central and obvious sustainability goal for Cognatio is Human Sustainability, as the system seeks to improve patient wellbeing. For example if the system is well-developed, it will help patients adhere more closely to their medications, their health will improve, and Human Sustainability will be enhanced. The collection of these goals and subgoals would make for a functional system that will encourage patients to consume their medication as directed and provide medical practices with valuable information concerning their patients and their corresponding treatments.

D. System Vision

The purpose of the system vision (Figure 5) is to give an overall view of the function of the system and how the stakeholders interact with the system. The business context of the system depicts the overall process of and the stakeholders involved with the development of the system.

In the operational context, the function and purpose of the system is outlined by the interactions between the members of the medical practice, their patients and the two components of the system: the desktop system and the mobile application. The purpose of the system is to have doctors, physician assistants, and secretaries of a medical practice track their patients’ usage of medication and encourage patients to adhere to the usage of medication as it is directed.

This is done by having secretaries remind their patients to take their medication when irregular activity is reported and by having doctors and physician assistants use the data produced by patients to adapt treatments for their patients based on their medication usage. This would enhance the patients’ consistency of medication consumption so that their medical conditions will improve, fostering advancement of Human Sustainability.

The System Vision uses input from the Stakeholder Model and the Goal Model, and provides a basis for the Domain Model.

Figure 5: System Vision of Project Cognatio
Figure 4: Goal Model
E. Domain Model

The purpose of a domain model (Figure 6) is to map out the various entities, their corresponding attributes, and their relationships to other entities. The medical practice has three types of employees that interact with the desktop system: the secretary, the physician assistants (PAs), and the doctors themselves. The Desktop system gathers and organizes information supplied by the patient concerning their habits with medication consumption and any side effects. The information is transmitted from the mobile application that contains the data about the patient’s medication adherence and reported side effects. From here the doctor may take further action to enhance the treatment of the patient and improve his or her condition, improving Human Sustainability. The content of the Domain Model is linked to the System Vision.

F. Usage Model

The purpose of the usage model (Use Case Overview depicted in Figure 7) is to give a black box depiction of how users will interact with the system and how the system responds to those interactions. It is a birds-eye view of the overall functions of the system. Also included are the three key components of the system: the desktop sub-system, the mobile application, and the patient database. Each function on either client subsystem of the Project Cognatio system interacts with either the database server or the other client subsystem. Solid lines depict an interaction between a user and the system, and the dashed lines depict the actions that the system itself performs to manipulate and transmit any reported or updated data. For example, the secretary uses the desktop system to send a reminder to the patient to take his or her medication, and that patient uses the mobile application to receive the reminder and hopefully adheres to it, improving his or her health and thus overall Human Sustainability.

G. Nonfunctional Requirements

The purpose of defining non-functional requirements (NFRs) is to highlight certain qualities that the system must adhere to that do not define any specific functions that it will perform. In this model, there is a focus on the Project Cognatio’s deployment, development process, constraints that limit the system, and requirements that establish the quality of the system. Of course, this list is not exhaustive, but the most important and tangible requirements are stated and will be met. An example Legal Requirement that implements an environmental sustainability concern is shown in Figure 8.

<table>
<thead>
<tr>
<th>NFR</th>
<th>Rationale</th>
<th>Satisfaction Criterion</th>
<th>Measurement</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does not violate any EPA standards</td>
<td>Improve environmental sustainability</td>
<td>System does not harm the environment</td>
<td>We would be notified by the EPA if this does infringe on the environment</td>
<td>One of the main goals of the project would not be achieved. The environment may be harmed. We may be fined.</td>
</tr>
</tbody>
</table>

Figure 7: Usage Model Overview

Figure 6: Domain Model

Figure 8: An example legal requirement for Project Cognatio
IV. DISCUSSION

This section discusses the presented artifacts and their usefulness in representing the sustainability aspects of the system under consideration.

A. Business Case Analysis

The Business Case Analysis for this requirements specification goes into great detail on how Project Cognatio will impact sustainability. The Business Case Analysis displays the perspective that the stakeholders have on the problem that will be solved by the software system under development. In addition to the core elements (problem, analysis, solutions, recommendations, and project description), the Business Case Analysis for this requirement specification also included a project timeline and a whole section devoted to how the pursued project will contribute to sustainability.

B. Stakeholder Model

The stakeholder model documents all of the stakeholders involved in the system and if their relationships directly or indirectly impact the development of the system. When considering sustainability, we thought of the Environmental Protection Agency (EPA) as representative for the environmental dimension.

The EPA is the governing body that makes sure all technologies, goods, and services comply with governmental environmental protection standards. They would of course only be indirectly involved, but definitely considered for the development of this system when thinking about the impact that this system has on Environmental Sustainability. They regulate the impact that all products have on the environment, and software systems are no exception.

If sustainability were to not be considered, the involvement of the EPA would have been absent because it is not a direct stakeholder in the development of the system. One can argue that software development that has little involvement with the environment would not consider the involvement of environmental regulations because such software would not typically effect the environment unless it introduces a new device or physical component that could be disposed when its product life ends. The purpose of a sustainability expert is to document how specific services and products would affect the different dimensions of sustainability. Current practice is to weigh if it is truly necessary to consult a sustainability expert when developing software that is not targeted toward improving any aspect of sustainability. This is because a sustainability expert may introduce new challenges to the system in order for it to maintain or improve sustainability. However, any software system has indirect impacts on the environment in its application domain, and therefore the EPA should be considered during requirements elicitation.

C. Goal Model

Since the goal of the system is to improve all five dimensions of sustainability with this system, the Goal Model considers each dimension. In fact there is a specialized blue key on the legend that is used to distinguish sustainability goals from all the other goals. The top half of the Goal Model contains sustainability goals, which would have been absent if not considering sustainability and the primary goal of “Improving Medication Adherence” would be focus of the project rather than improving all five dimensions of sustainability. Most of the sustainability goals are results of second or third order effects, and those effects would probably still occur or at least be possible with the development of the system, but they could be anticipated and mitigated to some extent.

The consideration of sustainability certainly plays a part in choosing the compatible operating systems for the system. Technical Sustainability would be achieved if more people with older devices could use the system in order to prevent them from disposing of their older devices. This is why Windows 7, Android 4.0, and iOS 7 were chosen as the compatible operating systems for Project Cognatio. Of desktop computers, Windows 7 is the dominant operating system, running on 50.06% of machines as of April 2014 [9]. For Android, if the mobile application is compatible with older operating systems, it will be compatible with newer operating systems. All OS’s that are Android 4.0 and above occupy just over 85% of all Android devices as of July 2014 [10]. And 90% of all iOS devices are running on iOS 7 as of July 2014 [11].

If sustainability were to not be considered, the goal to target popular operating systems would be based on dominating market share by reaching as many consumers as possible. But, this would not change any goals to develop for other operating systems. Only the motive for developing for other platforms would be changed and the two subgoals that denote which OS to develop for would stem from the goal to “dominate market share for unique service.” The overall goal of the system and its development to improve medication adherence would not change with the consideration of sustainability. The system would still manage to promote human sustainability regardless of its consideration while the emphasis on the sustainability goals would be absent.

D. System Vision

The System Vision gives a focus of the function of system and the involvement and roles of stakeholders on the system’s development. When considering sustainability, the most important factors in the System Vision that account for sustainability are the involvements of a sustainability expert and the EPA as well as the emphasis on human sustainability due to improved patient wellbeing. As the system vision is the central artifact that is used for communication with a whole range of stakeholders (including the non-technical ones), an illustration of the sustainability concerns within this artifact is of crucial importance.

E. Usage Model

The various components of a usage model focus on the different scenarios and cases that depict how the different users interact with the system. The usage of the system does not really address any aspect of sustainability; it simply involves a black box view of how what the system does and how it works under certain circumstances outlined by the
user’s needs for the system. The usage model for Project Cognatio does not specifically address any aspects of sustainability other than the improvement Human Sustainability, the main goal of the entire system.

F. Non-functional Requirements

The model for non-functional requirements merely documents the various components of specific, important requirements and goals that the system must meet when it is deployed for usage. In this model, Environmental Sustainability, Technical Sustainability, and Human Sustainability are specifically addressed while the reference to Social and Economic Sustainability is more implicit. In the list of non-functional requirements (full list available in the TechReport [10]), the consideration of sustainability involves the involvement of the EPA and few of the second and third order impacts on sustainability when addressing the risks if certain requirements are not met, as in Figure 8.

If sustainability were not considered when documenting this model, the consideration of these risks and the EPA would be absent, which could lead to environmental regulations being overlooked. It would not be entirely necessary to consider the regulations of the EPA when developing a software system like Project Cognatio when physical devices are not being created or tampered with which may impact the environment. Referring back to the Goal Model, if Technical Sustainability were to be eliminated from the model, the development of the software to be compatible with certain operating systems would be missing, an aspect to meet the goal of dominating market share. For this model, sustainability goals need to be considered to make sure there is a balance with other business goals and risks.

For example, if the requirement to make the GUI user-friendly is not met (Figure 9), then the risks would not include a focus on adherence to sustainability goals but rather a more primary focus on business opportunities. In the case of this requirement, the risk of opening up the market for other competitive products would be encountered.

<table>
<thead>
<tr>
<th>NFR</th>
<th>GUI is user friendly on both desktop and mobile applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale</td>
<td>System is easy to use on both ends</td>
</tr>
<tr>
<td>Satisfaction Criterion</td>
<td>Approval from beta testers</td>
</tr>
<tr>
<td>Measurement</td>
<td>It takes only a few (&lt;4) taps/clicks on the UI to navigate to desired page.</td>
</tr>
<tr>
<td>Risk</td>
<td>Few customers will want to use the application. All large-scale goals (sustainability goals) will not be met if application is not adopted. Opens up market for other apps with similar functionality and more user friendliness.</td>
</tr>
</tbody>
</table>

Figure 9: A Quality Requirement concerning the GUI with a business risk if the requirement is not met.

V. CONCLUSION

When following the RE4S approach [3] for RE, the consideration of sustainability has more effect on some artifacts than others. Earlier artifacts, such as the Business Case Analysis, the Goal Model, and the System Vision are impacted the most with the consideration of sustainable development. Later artifacts, with the exception of the Non-functional Requirements, are impacted less with the consideration of sustainability, but may have to consider earlier sustainability requirements as technical constraints.

This case study adds another validation element to the groundwork for future projects that utilize RE for sustainability. When considering each artifact in future software projects, especially in the medical domain, the consideration of sustainability should parallel that of Project Cognatio. As software systems affect most aspects of our daily lives, enabling software engineers to align the objective of environmental sustainability with the other objectives for the software system under development could considerably decrease the impact of people in the industrialized world on our environment.

The application of RE4S – explicitly considering sustainability during the requirements engineering phase of the project – systematically integrates sustainability goals and requirements with other functional and non-functional requirements, all of which should then be refined into software-specific constraints and considered during design and implementation of the system. This requires additional effort to demonstrate that the best design decisions are made leading to the resulting system satisfying the sustainability goals. Once Cognatio is implemented we can measure some of the sustainability effects due to specific consideration during RE.

Future work with regard to Project Cognatio is to implement the application following the sustainability guidelines and put the system into practice, thereby improving patient medication adherence. Future work with regard to the RE4S requirements engineering approach is to transfer the guidance to various industry partners and collaboratively apply it on a larger scale. Furthermore, we plan to integrate it into teaching activities, thereby influencing the next generation of software engineers.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the involvement of the following individuals for their assistance and support in contributing to this project:

- Dr. Don Mehrabi and his BHSkin staff for their feedback as stakeholders in the project.
- Dr. William Tomlinson for providing different perspectives of sustainability and for attending the SURF-IT presentation for the project.
This work is partly funded by the DFG EnviroSiSE project under grant number PE2044/1-1 and by the SURF IT program 2014 by the California Institute for Telecommunications and Information Technology at the University of California, Irvine.

References


Part V

Education
The Education part contains the publications with regard to the dissemination of the presented research in education and teaching.

These publications were partially developed in collaborations. The following list sets the context for the elaboration of each of the publications.

- **Teach Sustainability in SE? [66]**
  This contribution was developed in collaboration with Andreas Fleischmann at TUM and describes the concept of integrating the concept of sustainability in its different understandings step-wise into the software engineering curriculum. It was published at the International Conference on Software Engineering Education and Training in 2011.

- **Jumpstart Sustainability in Seminars [59]**
  This contribution was elaborated in collaboration with Veronika Bauer and describes experiences from a seminar series conducted at TUM on sustainability in software engineering. It was published at the International Computer Science Education Research Conference in 2012.

- **Case Study-based RE Education: Evaluation and Lessons Learnt [69]**
  This contribution was developed in collaboration with Martin Mauhaux and Patrick Heymans from the University of Namur in Belgium. It describes the lessons learned from two requirements engineering courses with case studies on sustainability-impacting systems and was published at the International Conference on Software Engineering Education and Training in 2013.

- **Using Non-Profit Partners to Engage Students in RE [74]**
  This contribution reports on results from a collaboration with Beth Karlin from UC Irvine and Allison Cook from the Story of Stuff project and reports on the insights from conducting a requirements engineering case study on an online learning system for sustainability causes. It was published at the International Workshop on Requirements Engineering Education and Training.
20 Teach Sustainability in SE?
Teach Sustainability in Software Engineering?

Birgit Penzenstadler
Institut für Informatik
Technische Universität München
penzenst@in.tum.de

Andreas Fleischmann
ProLehre, Carl von Linde-Akademie
Technische Universität München
fleischmann@prolehre.tum.de

Abstract

Sustainability is becoming an important topic in IT—as contribution of IT to safeguard our future, and as evolving market segment. IT’s high productivity in combination with short life cycles and, on the other hand, growing resource problems of our planet, lead to a necessity that software engineers take their share of responsibility for sustainability. Therefore, we need to include the concept of sustainability into the university curriculum of computer science.

The challenge is to motivate and interest students (and lecturers) for sustainability, to identify spheres of activity for software engineers, to build up competence fields for solutions, and to incorporate the topic into the syllabus.

This paper presents a strategy for integrating the concept of sustainability into a degree course scheme across three stages: find a core of interested people by offering a seminar, then broaden the awareness for sustainability by offering a lecture series, and finally establish the topic by offering teach-the-teacher seminars and integration into software engineering lectures.

I. DOES SUSTAINABILITY MATTER? YES!

With a constantly growing world population and constantly decreasing resources, sustainability has become an important research and application topic (see, e.g., [1], [2]), including research fields such as energy efficiency, material efficiency, waste efficiency, co2 efficiency.

More and more domains acknowledge sustainability as a contribution to safeguard our future—and as an evolving market segment. Within academia, some domains have taken action and offer sustainability in separated curriculums, e.g. in economics, as Master of Sustainability Economics and Management [3] or Master of Arts in Responsible Management [4], and as field of activity in research, e.g. in vocational education [5] and logistics [6]. In computer science, green IT is one of the new buzzwords in industry (e.g., [7], [8], [9]), consulting (e.g., [10]), and at conferences (e.g., [11]). Unfortunately, the term green IT is sometimes also used for greenwashing and does not necessarily support sustainability from all perspectives.

At first glance, with software being a nonmaterial construct, it seems that sustainability is of no importance in software engineering. However, there are a lot of scenarios where software does have an impact on resources, especially energy: efficient software consumes less energy and requires less hardware; software intelligently controlling a machine’s behaviour reduces it’s energy consumption; also the software engineering process itself can save resources (e.g. paperless offices, model checking reducing the number of test cases or prototypes). Furthermore, the products have to be manufactured and delivered, and, by the end of the day, they have to be disposed of. As IT contributes to producing more systems in less time, it is also an important lever for promoting sustainability.

There are two reasons why sustainability matters to us as software engineering educators: First, we need to learn and teach how to make efficient use of resources in our specific domain. Second, we want to take into account emerging markets in order to attract both students for choosing an enrollment at our university and industrial investors and partners for research collaborations which involve the students’ more practical education. Therefore, we argue that sustainability is an important future topic in software engineering education [12].

We need yet to define what sustainability means for software engineering. Sustainability is the capacity to endure and, for humans, the potential for long-term maintenance. Therefore, we need sustainable development with a responsible consumption of resources. Sustainable development includes the aspect “to develop a sustainable product” as well as the aspect “to develop a product using a sustainable development process”.

Given a faculty and students that might not be familiar with sustainability as a matter of software engineering, it is quite a challenge to incorporate sustainability into the syllabus, and even more a challenge to actually motivate and interest the students (and teachers) for that topic: Why would our students be interested in learning about sustainability? If the students are not interested and motivated, they will not learn anything and what we try to get across will simply pass by unnoticed.
So an important part of teaching sustainability is raising awareness and motivating the issue [13]. The goal is to convince the students with our enthusiasm and credibility (“walk your talk”) without annoying them by becoming a missionary.

Sustainability is one more central quality attribute in a row with the standard quality attributes of correctness, efficiency, and so forth. In order to be able to measure sustainability, we require an awareness for the potential starting points of sustainability. Therefore, we need to investigate spheres of activity for software engineers where sustainability can be incorporated, as well as competence fields for developing solutions.

For integrating the concept of sustainability into a degree course scheme, we propose a strategy across three stages: a master’s seminar, a lectures series, and a systematic integration into SE lectures with teach-the-teacher seminars.

Outline: Section II explains the strategy and its steps and Section III gives an overview of important related work. Section IV summarises and concludes the paper with an outlook on future work.

II. EDUCATIONAL STRATEGY

Sustainability is a cross-cutting concern within software engineering. No matter whether we build compilers, robots, mobile applications, or business information systems, it can be done energy-efficiently. No matter whether we lecture about cloud computing, architecture design, or software project management, we can consider energy-efficiency and responsible resource consumption. There are many more aspects that have to be considered for sustainability in software engineering, for example, concerned with the question “How can we make the software development process itself more sustainable?”

The students shall get a general view on all aspects of the topic and be encouraged to train themselves in the direction of systems thinking [14].

Our concept for teaching sustainability consists of three stages:
1) Finding a core of interested people by offering a seminar.
2) Broaden the awareness for sustainability by offering a lecture series.
3) Establish sustainability as topic by integrating it into the syllabus of appropriate software engineering lectures with teach-the-teacher seminars.

An overview is also depicted in Fig. 1. The following subsections detail each of the stages and present concrete ideas for their realisation.

A. Getting Started: Seminar

The objective for the first stage is to start building a small community of students who are interested in sustainability. We are looking for first reactions and early feedback as input for the further shaping of the concept. Furthermore, there are no main stream of solution strategies yet for sustainability in software engineering. The means for receiving first feedback on students’ interest is a master’s seminar for graduate students. At the same time, a master’s seminar allows researchers to collaboratively explore a relatively new field with advanced students, thereby creating a win-win situation.

The concept of a seminar is to find interested students and to get them engaged to actively investigate a specific topic within current research interests. Thereby, they learn to organise themselves and elaborate their individual results, as compared to exercises and projects where the outcome is mostly determined. For the seminar, the only fixed outcome is that all students shall learn about the topic of a particular (group of) students during a 60 to 90 minute session in class.

Interpreted in a liberal way, advisors and students have room to play and creatively find new ways of passing knowledge on to the other students. According to our experience in other seminars, the more freedom we give them to explore and present their topic, the more engaged most of them are.

The seminar is organised in three sessions:
1) Introductory session: Warm-up with the Cooling Down Game [15] to introduce the topic in a playful manner. We give a general overview of the area and engage the students in various group discussions where we can explore different perspectives on sustainability. Then we let each of the students (or small teams, depending on the overall number of participants) choose their topic and give a range of examples of how to approach them.
2) Reporting session: This session can be replaced by individual appointments with the students or student teams. Each of them reports on their progress with the topic and how they are going to present the contents to the audience during presentation session.

3) Presentation session: The students or student teams are “on stage” and present their work to the audience. We wrap up the session with feedback in both directions (advisors to students and students to advisors) and identify future work and next steps. The results of the seminar are published on the web.

The following list is a collection of ideas for seminar topics. Our plan is to discuss them with the students, and encourage them to make innovative approaches, to play, and to investigate on their own. According to the well-known learning effectiveness curve from perception to reproduction to personal experience [16], the students have to find ways to make the topic tangible (first for themselves, then for the other students).

1) Develop an analysis of aspects where sustainability can have an effect in software engineering, considering both the development process and the resulting product itself.
2) Evaluate the first model for green and sustainable software engineering [17] or the proposed sustainable software engineering method evaluated at NASA [18] with a self-conducted small software project.
3) Compare different governmental approaches to handle sustainability [19] and find out how well they are realised in your city or region. What is the potential in other countries? [20]
4) Greenwashing ([21], [22])—how is it done? How can greenwashing be detected? Perform a little case study workshop with the goal to greenwash a product.
5) What could a “sustainability footprint” for software engineering look like? First ideas are given by the Greentracker tool [23].
6) Climate killer internet? Energy-efficient nets and systems have a notable impact [24].
7) A critical application domain: climate change research and software engineering for climate research [25].
8) Software-engineering without computers—how far could we go? This is an analogy to the paperless office. The idea is that lack of communication is a problem in software engineering practice and it can probably be solved by first communicating intensively and then documenting the contents, thereby saving paper and energy.
9) Marketing for sustainability—how can I make it matter for software engineers? This can be supported, for example, with a small survey study in software development companies.
10) Algorithmic efficiency (e.g. average google query: 0.2g co² [26], [27]) to reduce energy consumption in computing centres—is it worth the effort?
11) Intelligent energy grids—what can software contribute? A small project gives hands-on insights.
12) Excursion to a local energy provider to get hands on energy production and consumption, and to meet software engineers who work in the field of energy management.
13) Energy efficiency of different machines across the decades, and where is IT today compared to this development? (e.g., computers, laptops, displays, washing machines, cars, …)
14) Think tank for new solutions: Traditional approaches like improving efficiency will not suffice to battle the climate change. Therefore, we encourage students to think out of the box and come up with whatever seemingly crazy idea crosses their minds. An open-minded, supportive brainstorming space is crucial before starting to discuss concrete steps to evaluate the feasibility of a specific idea [28].

All these ideas are preliminary and, so far, have no explicit storyline relating them and leading from one to the other. The reason for this is that we want to discuss the ideas with the students and give them maximum freedom in choosing and shaping their specific topic. Even if the students come up with totally different topics by themselves that is perfectly alright with us. Intrinsic motivation is the best way to get them interested and stick to the seminar. Given that the field is not yet too well covered in literature, students might develop new research ideas and pursue them with their supervisors. That way, we want to build and gather a small community that serves as basis for the further steps outlined in the following, because to achieve changes in conscience and action, a critical mass needs to be motivated.

The idea of a seminar can be realized with either bachelor or master students. We are holding the seminar during the upcoming summer term, so we will be able to report on first experiences during the conference. The results of the seminar will subsequently be published as technical report.

B. Building up: Lecture Series

The seminar gives us a better understanding of the topic as well as the related interests of our students. The objective of this stage is to expand the small initial community to a critical mass. The means for achieving this are to give a general overview of the area of sustainability and explore different aspects and perspectives of the topic. For that purpose, we involve the students from the seminar by building up on their favourite topics. Maybe we can invite experts for the most promising topics from the seminar and let the respective students who worked on them introduce the experts in their lecture.

Ideas for giving a general overview include:

...
• Taking the audience for a virtual walk through the daily life of a software engineer highlighting the effects of their behaviour in a long term view.
• Let the students from the seminar conjointly hold a lecture to present the insights gained and results produced during the seminar.
• Presenting potential improvements that can be achieved through sustainability and their effects in long term perspective.
• Explaining the green IT reach-richness matrix [29] that provides an overview of the reach dimensions of green IT (creation, sourcing, operations, disposal) related to the richness dimension (policies, practices, technologies).
• Introducing the (country-specific) governmental agenda for sustainability to show that the topic is recognised and given importance.

Aspects that are relevant to consider for designing and realising sustainable solutions include:
• Energy (electricity, heat, cooling, …)
• Raw materials (metal, silicon, …)
• Human resources (developers, managers, …)

Perspectives that represent the most important stakeholder classes in the different potential application domains include:
• Manufacturing: development, production, maintenance, and disposal
• Consulting: productive sector and service sector
• Consumer: day to day life and long term perspective (return on investment)

These aspects and perspectives can be integrated into the talks by taking examples from the different domains or by inviting speakers specialised on these topics, for example sustainability consultants.

C. Teach the Teacher and Integration into SE Lectures

At the third stage of the concept, the idea is to mention and refer to sustainability during software engineering lectures whenever appropriate. That objective requires us to convince the lecturers of integrating the topic into their courses.

Such integration does not necessarily take a lot of up-front investment but can be introduced gradually according to the interest of the lecturer. At first, they only mention that they prefer the more energy-efficient solution when talking about different solutions for a specific problem. For example, the lecturer presents different options for an embedded system design and points out the most sustainable solution. The important factor is that the students get the notion of “This topic matters to my professor.” as a by-product during their education of software engineering. This hopefully triggers the second part of “I should give it some thought, too.”, even more so if the lecturer is popular and therefore accepted as kind of role model.

Later on, they (optionally) include a short chapter on sustainability supported by a small exercise into their lecture and tutorial. One idea for an exercise is to reuse an example system that has been designed during an earlier exercise and let the students reconsider the design under the aspect of sustainability. Another option for further integration is to invite a special guest for the topic and let them give an expert talk on current best practices for sustainability in industry or/and in consulting. This approach to integrating new topics into a curriculum is considered as the most sustainable way in terms of making knowledge last in students’ heads [30].

To offer the lecturers the possibility to enhance their knowledge about the current state of practice and state of research in sustainability, it is necessary to provide supporting teach-the-teacher seminars. Depending on the previous knowledge of the audience, these can include:
• Invited expert talks from either industry or other research institutions and faculties.
• The results from the master’s seminar.
• Lessons learned during the master’s seminar.
• Experiences from the lecturers who have integrated sustainability into their SE lectures, e.g. [31].
• Further group work as described in Sec. II-A.
• Resources for teaching material as listed, for example, in Mann et al. [32].

Such seminars can be held for SE lecturers, and later on computer science lecturers from different areas of research. Further more, they can be extended and adapted for lecturers from different faculties, for example mechanical engineering, architecture, …

In parallel to the idea of spreading the paradigm of sustainability all over the campus, we also envision an integration of the ideas of sustainability in school education as soon as computer science is taught within the children’s regular curriculum. Also addressing school teachers increases the potential audience for the teach-the-teacher seminars.

III. RELATED WORK

The topic of teaching sustainability in software engineering has been addressed by Samuel Mann and Yu Cai. Mann advocates computing for sustainability in his blog [33] and Mann et al. [32] propose a framework to evaluate resources for the integration of sustainability in the computing curriculum. The “CE4S” framework shall be used by educators to assess potential resources and,
in the long run, to create an open inventory of resources. Cai [31] describes how he integrates sustainability into undergraduate computing education in a course called “Green Computing & Network Services”.

Several universities are approaching sustainability in computer science education with a seminar where students work on dedicated topics as proposed in step 1 of our strategy, for example [24], [6]. The aspect of sustainability is also relevant for a global model for software engineering education as proposed by Gotel et al. [34] or in the context of education in developing countries [35].

IV. CONCLUSION AND OUTLOOK

In this paper we presented our concept for including sustainability into software engineering education. Sustainability is one important quality attribute in the row of standard quality attributes like correctness and efficiency for which we still need to gain insights on spheres of activity and solution fields. The sustainability education concept consists of three major steps: getting started in a master’s seminar, building up with a lecture series, and integrating with general lectures on software engineering after teach-the-teacher seminars.

We consider it vital to include sustainability into software engineering education as ambassador for promoting the value of sustainability to our students. The first step is the seminar, scheduled for spring 2010. With the gained experience, we plan a first edition of the lecture series before we move on to general integration and specialised seminars.

REFERENCES

[28] [Online]. Available: www.openplanetideas.com
[33] [Online]. Available: http://computingforsustainability.wordpress.com/
21 Jumpstart Sustainability in Seminars
Jumpstart Sustainability in Seminars: 
Hands-on Experiences in Class

Birgit Penzenstadler, Veronika Bauer 
Technische Universität München, Germany

Sustainability in its different aspects is hardly addressed in software engineering education, neither as quality objective of system development nor in business process design. Consequently, students tend to be unaware of this concept and are not considering sustainability as an important aspect of systems development. This results in the development of (environmentally or socially) suboptimal solutions—although information and communication technology systems could offer great support in promoting and enabling sustainability in our society. To unlock this potential, we are sensitizing students for the issue by gradually introducing the concept in the curriculum.

In this paper, we report on experiences in establishing sustainability in the software engineering curriculum of Bachelor and Master students by means of designated interactive seminars. A guideline on how to establish similar teaching activities concludes the paper.

Categories and Subject Descriptors: K.3.2 [Computer Science Education]: Sustainability, Software Engineering

General Terms: Sustainability, Software Engineering, Requirements

Additional Key Words and Phrases: Education

ACM Reference Format:

1. MOTIVATION AND BACKGROUND

Sustainability is becoming an important topic in IT—as a contribution of IT to safeguard our future, and as an evolving market segment. The most cited definition for sustainable development is that it “meets the needs of the present without compromising the ability of future generations to meet their own needs” [United Nations World Commission on Environment and Development 1987].

The rapid development of information and communications technology (ICT) is contributing a considerable share to excessive usage of resources by having improved our working capabilities by orders of magnitude during the past century. ICT’s high productivity in combination with short product life cycles and growing resource problems lead to software engineers needing to take their share of responsibility for sustainability. This is also acknowledged by various software engineering conferences, for example, this year’s ICSE slogan “Sustainable software for a sustainable world”.

However, sustainability is not yet a first class citizen in the family of quality attributes, currently taught to future software engineers, as, e.g., the ISO criteria [International Standardization Organization 2001]. In [Penzenstadler and Fleischmann 2011], we presented a concept for integrating sustainability into the software engineering (SE) curriculum as well as the first steps taken to implement our strategy at the Technische Universität München (TUM). We proposed a combination of introducing the topic of sustainability in dedicated seminars and courses with infusing existing courses on software engineering. Seminars catch the attention of students who want to focus on the topic, whilst integration into existing material increases the awareness of students who are not actively concerned with sustainability.

Problem: Sustainability is hardly addressed in current software engineering education. It does not appear in the list of quality objectives of software, and the sustainability of usage processes in the application domain is also neglected. As a result, students are unaware of this aspect—they are not considering sustainability in their development activities. The challenge is to attract the interest of students (and lecturers) to sustainability, to identify spheres of activity for software engineers, to build up competence fields for solutions, and to incorporate the topic into the syllabus.

Contribution: This paper reports on the experiences from three completed seminars, one for Bachelor and two for Master students, at two universities, the TUM and the Universitat Polytècnica de Catalunya (UPC). We present the selected topics and our didactical strategy. Furthermore, we report on the seminars’ realization, results, feedback, and the lessons learned. In addition, we present planned student projects, and our strategies to trigger multiplier effects. For other researchers and educators, this paper serves as a starting point for own seminars and activities, and as a basis for discussions on how to integrate sustainability from a software engineering viewpoint into the student curriculum.

Outline: We first report on related work (Section 2). Then we present the selected topics (Section 3). Section 4 introduces our didactical strategy. Section 5 describes the context of the seminars and their realization. This is followed by an assessment of the seminars’ outcome, and the lessons learned (Sec. 6). Furthermore, we describe ongoing work and give an outlook on future work (Sec. 7).
Finally, we conclude with a guideline to integrating sustainability in the curriculum (Sec. 9).

2. RELATED WORK

Research on teaching sustainability in software engineering has been done by Samuel Mann [Mann et al. 2009], Yu Cai [Cai 2010], Lorenz Hilty [Hilty 2010], and us [Penzenstadler and Fleischmann 2011].

Mann et al. [Mann et al. 2009] propose a framework to evaluate resources for the integration of sustainability in the computing curriculum. The “CE4S” framework shall be used by educators to assess potential resources and, in the long run, to create an open inventory of resources. They do not state how to perform the integration.

Cai [Cai 2010] describes how he integrates sustainability into undergraduate computing education in a course called “Green Computing & Network Services”. The approach is focused on integration of green IT into a lecture on networking while we aim at a broader understanding of sustainability and its integration into software engineering education.

Hilty [Hilty 2010] has a research group on Informatics & Sustainability in Zürich where lectures on informatics & sustainability are already an established part of the curriculum. He has not yet reported on how to proceed for successfully integrating sustainability into the informatics curriculum. In [Hilty et al. 2011], he emphasizes the importance of the decoupling of economic growth from natural resource usage as the most important means to support sustainability by using ICT.

Our search in journals on sustainability in education and environmental science revealed no work related to software engineering education; not even to computer science education in general.

Boyle [Boyle 2004] sketches challenges of and requirements for educating engineers in sustainability, and derives a topic list for the curriculum to suit these needs. There is no report on plans to implement the sketch activities. Kurland [Kurland 2011] reports on the evolution of a campus sustainability network. She looks at how to improve the university’s sustainability and what impact their efforts had, but computer science skills are referred to only as “needed” by physical plant management projects [Kurland 2011, p. 409].

In the journal “Environmental Science and Technology”, Mihelcic et al. [Mihelcic et al. 2003] make a case for sustainability science and engineering as a new “metadiscipline” but restrict themselves to engineering and make no reference to informatics or computer science.

In [Penzenstadler and Fleischmann 2011], we propose a strategy for faculties that don’t have a distinguished research group for sustainability. In the following, we present the first experiences in implementing this strategy at TUM.

3. TOPICS

The topics we analyzed with our students were chosen according to the students’ experience and degree of specialization in software engineering. Therefore, we present the topics specified for each seminar type: one Bachelor seminar and one Master seminar at TUM, and one Master seminar abroad at the Universitat Polytècnica de Catalunya (UPC) in Barcelona. As foundation for the work in the seminars, we conducted a systematic literature survey on sustainability in software engineering [Penzenstadler et al. 2012].

<table>
<thead>
<tr>
<th>Title</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is Sustainability?</td>
<td>History, definition, and related fields of sustainability</td>
</tr>
<tr>
<td>Sustainability in Legislation</td>
<td>Which role does sustainability play in politics?</td>
</tr>
<tr>
<td>Greenwashing</td>
<td>What is greenwashing, how can it be unveiled?</td>
</tr>
<tr>
<td>Sustainable Software Development Products</td>
<td>How can software development be more sustainable?</td>
</tr>
<tr>
<td>Sustainable Software Products</td>
<td>Can software products be sustainable?</td>
</tr>
<tr>
<td>Case Study Green Energy</td>
<td>What is green energy? What is the contribution of computing?</td>
</tr>
<tr>
<td>Case Study Internet</td>
<td>(How) does the Internet influence sustainability?</td>
</tr>
<tr>
<td>Green Car IT</td>
<td>Can IT help to produce greener cars?</td>
</tr>
</tbody>
</table>

3.1 Bachelor Seminar: Sustainability in Software Engineering

This Bachelor seminar was the first seminar on this topic to take place at the faculty of Computer Science of TUM. Starting from scratch allowed for a considerable freedom. We scanned a wide range of computing-related sustainability issues for scientific activity as well as their suitability for further inspection in the field of software engineering research. The list of topics is shown in table I.

3.2 Master Seminar TUM: RE for ICT-Systems for Environmental Sustainability

In the winter term 2011, we offered two seminars for Master students, one at TUM and one at UPC in Barcelona. The TUM seminar is specifically on “Requirements Engineering for ICT-Systems for Environmental Sustainability” and the UPC seminar on “Environmental Sustainability in Software Engineering”.

For our second seminar at TUM, we chose Master students in order to explore the full range of possibilities of integrating the topic into the curriculum. The topic of the seminar is more specific with a focus on requirements engineering and an explicit specification of which kind of systems we want to analyze. We are interested in systems that reduce man’s impact on the environment by new approaches in ICT that explicitly take sustainability into account. We call these systems “ICT for environmental sustainability” (ICT4ES). A prerequisite for the seminar is experience in requirements engineering by either having attended the requirements engineering lecture or through industrial practice.

ICT4ES systems especially account for ecologic aspects, for example the resource consumption during development but also the implications of the system for future business processes. At the same time, in order to have practical significance, they also need to be socially and economically sustainable. Therefore, all three aspects need to be taken into consideration when exploring the suitability of specific requirements engineering (RE) methods and techniques from related disciplines.

Table II lists the topics of the seminar. The first two topics set the scope for the seminar by analysing present definitions and delimiting the boundaries of the context. Three topics assess usage analysis and prediction of deltas between user expectations and actually supported system interaction. Three more topics investigate
Table II. Topics of the TUM Master Seminar: Requirements Engineering for ICT-Systems for Environmental Sustainability

<table>
<thead>
<tr>
<th>Title</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyses, definition and synthesis of the terms sustainability and RE</td>
<td>What is their relation and what is their potential?</td>
</tr>
<tr>
<td>Delimitation of the terms Green IT and sustainability</td>
<td>Which aspects are emphasized in Green IT and what are the implications on RE and available methods?</td>
</tr>
<tr>
<td>which analyses methods are suitable to detect how a system changes the interaction with the user?</td>
<td>Investigate how information systems change the world in which they are applied.</td>
</tr>
<tr>
<td>How to detect in RE how the user might act differently in using the system from what was expected during systems development?</td>
<td>How can be predicted in which form the user will actually eventually interact with the system?</td>
</tr>
<tr>
<td>How to determine discrepancies between the user expectations and the modified processes that a new information system might involve?</td>
<td>How to perform a delta analysis between requirements and implementation?</td>
</tr>
<tr>
<td>which stakeholders have a direct interest in sustainability and how can they be classified?</td>
<td>Which methods are suitable to elicit and document the requirements of these stakeholders?</td>
</tr>
<tr>
<td>which aspects of the business context to capture and how to analyze their impact on business processes and their modeling?</td>
<td>How to analyze and model regulations, value chains, vertical integration, and life cycle of the business objects?</td>
</tr>
<tr>
<td>Which existing methods are suitable for context modeling for ICT4ES?</td>
<td>Analyze a number of methods, for example, Quasar Enterprise [Engels et al. 2008] and B-SCP [Bleistein et al. 2006].</td>
</tr>
<tr>
<td>which aspects of evolution of an information system have to be considered with respect to sustainability and what are the consequences for the business context?</td>
<td>How does evolution of a system interfere with sustainability and how can the resulting impact be analyzed?</td>
</tr>
<tr>
<td>At which point of the life cycle of an information system should metrics for environmental sustainability be measured and assessed?</td>
<td>How to define quality assessment for environmental sustainability and respective quality gates?</td>
</tr>
<tr>
<td>which metrics can be used to measure and assess the environmental compatibility of a business process?</td>
<td>How to perform impact analysis of ICT4ES system processes?</td>
</tr>
<tr>
<td>Which aspects of maintainability (as defined in [Deissenboeck et al. 2007/])</td>
<td>Evaluation of suitability of the TUM quality model for maintainability for environmental sustainability.</td>
</tr>
<tr>
<td>have a direct impact on environmental sustainability and how can their compliance be assessed?</td>
<td></td>
</tr>
</tbody>
</table>

Table III. Topics of the UPC Master Seminar: Environmental Sustainability in Software Engineering

<table>
<thead>
<tr>
<th>Title</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systematic Literature Review of ICT4ES</td>
<td>Analysis with setup and complete protocol according to [Kitchenham et al. 2009]</td>
</tr>
<tr>
<td>Survey of potential ICT4ES systems and classification of domains</td>
<td>Which domains are relevant for ICT4ES and how to classify them?</td>
</tr>
<tr>
<td>Interview study with practitioners on values behind sustainability</td>
<td>Which stakeholders have a direct interest in sustainability and what are their values?</td>
</tr>
<tr>
<td>Survey of goals for sustainability and their classification</td>
<td>What are the goals for supporting sustainability and how to classify them?</td>
</tr>
<tr>
<td>Survey of metrics for sustainability and their applicability to software</td>
<td>What are the KPIs in use for environmental sustainability and how to apply them to software?</td>
</tr>
<tr>
<td>Suitability of domain modeling techniques</td>
<td>Which modeling techniques in software and in environmental sciences are suitable to perform domain modeling for ICT4ES?</td>
</tr>
<tr>
<td>Suitability of artefacts for sustainability requirements</td>
<td>What documentation artefacts are suitable to elicit and capture sustainability requirements?</td>
</tr>
<tr>
<td>Suitability of life cycle analysis to an ICT4ES System</td>
<td>How to apply life cycle analysis for the impact of a software system’s usage processes?</td>
</tr>
<tr>
<td>Suitability of risk assessment techniques for an ICT4ES System</td>
<td>How to apply risk assessment techniques for environmental sustainability?</td>
</tr>
<tr>
<td>Suitability of impact analysis techniques for an ICT4ES System</td>
<td>How to apply impact analysis techniques for environmental sustainability?</td>
</tr>
</tbody>
</table>

stakeholders and context modeling for ICT4ES. The last four topics explore different aspects of quality assessment and evolution.

3.3 Master Seminar UPC: Environmental Sustainability in Software Engineering

The scope of the seminar is similar to the TUM Master seminar, albeit a little more general. Due to the organizational frame, the UPC students have less time to prepare their presentations than the TUM students. Therefore, the topics are concrete tasks like small surveys and case studies to analyze the applicability of a specific method or technique.

Table III lists the topics of the seminar. In half of the topics, the students perform surveys on relevant domains, stakeholders and values, goals, and metrics. In the other half, they assess the suitability of specific methods for ICT4ES systems. For the latter, all students use the same case study in the domain of eMobility — this case study combines information systems and embedded systems and includes direct user interaction as well as background processes. By using the same case study, the students can base their work on a common set of goals and constraints and work together for the specifications. On the one hand, this might introduce a slight bias with respect to assessing the suitability of the modeling techniques. On the other hand, the goal is an illustrative case study that
exemplifies the techniques’ usage rather than an empirically strict study.

4. DIDACTIC STRATEGY

This section presents our overall didactic goals and general rules as well as specifics for the advanced seminars.

4.1 Goals

We want the students to engage proactively in the seminar, to be critical in researching and analyzing their topic, and to try new ways of interaction when presenting it. To achieve this we took the following actions: According to new findings in didactics, we created an interactive seminar setting which was embedded in a workshop format. We offered extensive tutoring as well as coaching on interactive presentation methods and techniques. Additionally, we invited researchers from other disciplines to give input during the seminar. Furthermore, we provided students with the opportunity to specify own topics, thereby supporting creativity and/or the possibility to better align the seminar work with their other assignments.

4.2 General Rules

There are a couple of rules that we communicated in the first meeting to set the context:

—“We expect you to work independently and to show active interest for your topic.”

—“You can always ask for advice at any time. We want to use the results of the seminar in research, so it is our concern that you know what to do and how to do it.”

—“Perform proper research: literature search, original sources, correct citations, all references, good writing style — if the results are good, we can publish them and your work is of use for other researchers as well.”

—“Make your presentation interactive. Let the audience participate instead of lecturing them — this creates lasting effects that stick to people’s memory. Illustrate your content.”

4.3 Deliverables

The students were required to participate in all meetings. Their deliverables were a presentation with an interactive part and an essay. The length of the essay was 5000 words +/- 10% for the BSc seminar, and 3000 words +/- 10% for the MSc seminars (as we focused on the analysis instead of presenting background knowledge). As further incentive and because the results may be interesting for other researchers, student seminar essays are usually published as a technical report by the end of the term if their quality was good enough.

4.4 Bachelor Seminar Specifics

Schedule/Format. We chose a two day workshop format for the seminar, rather than a once a week seminar of two hours, to ensure an intensive immersion in the topic. The setting also gave the students equal time to prepare their sessions. The time slots, which in the beginning seemed overly generous, in the end were completely filled and needed to give students the necessary space to explore their ideas. Also the scheduled long breaks were vital to regenerate and to keep the concentration at the required level. Students were participating actively and enjoying the discussions. They also acknowledged that, after the two days, they were very tired.

The interactive workshop format does not only challenge the students to prepare differently for the seminar and become more creative. It also requires the tutors to get involved more into the supervision by not only giving advice related to the topic, but also by coaching each student on didactical methods and techniques. Consequently, the tutors need to receive training themselves. Furthermore, they need to get to know their students to be able to propose interaction strategies which suit their personalities, as well as helping them to overcome their insecurities with respect to this open format. Otherwise, students will not feel at ease during their workshop sessions, incurring the risk of demotivating the group. Instead, if the students are comfortable in their new role as interactive presenter, this greatly improves the atmosphere and effectiveness of the seminar.

Infrastructure. To support students with their research, each advisor scheduled several meetings to discuss the students’ topics as well as providing guidance and feedback concerning the ideas for interaction. Furthermore, a wiki was set up containing initial reading material for each topic and the students were encouraged to add the results of their literature review to share them with the others.

Evaluation. We decided on grading the students’ workshop session according to three separate aspects: quality of content and research, interaction, and participation during the seminar. Attendance was obligatory at all seminar sessions to ensure a more predictable environment for the selection of suitable interaction techniques. The essay was graded separately following the university criteria for written coursework with respect to quality of the content (compliance to the topic, technical accurateness, originality, independently acquired knowledge, citation of relevant literature) and quality of the exposition (writing style, scoping, structure, conclusion, appearance).

4.5 Master Seminar TUM Specifics

The students shall understand how sustainability can be addressed within requirements engineering. Furthermore, they shall explore the options for sustainability analysis and how they can be supported by specific modeling concepts and techniques. We expect the students to have background knowledge in requirements engineering.

Interdisciplinary Collaboration. We initiated contact with other disciplines that also conduct research on sustainability in order to gain insights from their point of view. The following fields are investigating sustainability in their research at TUM: Forest Science, Mechanical Engineering, Civil Engineering, Philosophy of Science, Sociology and Politics. Research assistants of forest science, mechanical engineering and quality of the exposition (writing style, scoping, structure, conclusion, appearance).

4.6 Master Seminar UPC Specifics

Since this seminar was held at a host university, the schedule differs from the two seminars held at TUM.

The extent of the seminar is two consecutive weeks with a total of 12h of presence at class with lecture and active participation in

2Slahova et al. [Slahova et al. 2007] report on creative activity in conception of sustainable development education.

3https://sustainability.wiki.tum.de/
Jumpstart Sustainability in Seminars

• group work, followed by a homework phase with a written essay to be handed in 4-6 weeks later.

The learning goals for the seminar are:

— “Develop an understanding of what sustainability is.”
— “Get an overview of what aspects contribute to sustainability inside and outside of software engineering.”
— “Understand how sustainability can be supported within requirements engineering.”
— “Acquire methods and techniques to support the analysis of sustainability.”
— “Gain experience with interactive teaching.”

5. SEMINAR REALIZATION AND RESULTS

This section reports on the actual implementation and the results of the seminars.

5.1 Bachelor Seminar: Sustainability in Software Engineering

As our first step to introduce sustainability in the software engineering curriculum, we offered a pilot seminar on “Sustainability in Software Engineering” at TUM during the summer term 2011. In the following, we describe the context, our didactic strategy, and the seminar’s realization.

Table IV sums up the organizational facts.

<table>
<thead>
<tr>
<th>Table IV. The Seminar in Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
</tr>
<tr>
<td><strong>Target Group</strong></td>
</tr>
<tr>
<td><strong>Number of Students</strong></td>
</tr>
<tr>
<td><strong>Number of Advisors</strong></td>
</tr>
<tr>
<td><strong>Deliverables</strong></td>
</tr>
<tr>
<td><strong>Number of Meetings</strong></td>
</tr>
<tr>
<td><strong>ECTS credits</strong></td>
</tr>
<tr>
<td><strong>Duration student talk</strong></td>
</tr>
</tbody>
</table>

Realization. The seminar was structured in three phases: a pre-seminar session in which we presented background information and topic proposals, an introductory session in which the more general topics were grouped to lay the foundation for the more specialized ones, and a final two-day workshop. Students were assigned a 75 minutes slot, which they could fill freely under the condition of not exceeding 30 minutes of lecture-style presentation and using at least 45 minutes for interactive teaching. The allocated slot might seem extensive, but since interactive elements tend to be underestimated with respect to their time consumption, we pre-cautiously decided on a generous time frame. Our experience from other seminars had indicated the interactive format to be very effective in terms of learning. However, the setting is demanding a lot of activity and energy from the participants. For this reason, we scheduled no more than five topics per day, which were separated by regenerative breaks of 20 to 30 minutes. As the students formed a rather heterogenous group from different semesters and specializations within computer science, it was important to introduce them to each other and inspire a group feeling. To this end, we started each phase of the seminar with an interactive warm-up session to break the ice, unite the group, and prepare the way for their participation in the interactions.

Seminar Results. The results of the Bachelor seminar are available as a Technical Report [Penzenstadler et al. 2011].

The following paragraphs present examples of the outcomes of the seminar sessions, as well as the employed interaction techniques: During the interactive part of the session on “Green Energy”, participants were handed out worksheets, which had to be solved in small groups with the objective of computing the energy currently used versus the energy which could be potentially produced by sustainable technologies. Consequently, the participants had to get involved into making realistic estimates with regard to the current energy situation as well as the possibilities of renewables. To increase the awareness and put the single results in context, the results were visualized in form of paper strips being pinned on top of each other, in the attempt to match the current energy consumption with sustainable energy production (see Fig. 2). Furthermore, quiz breaks presenting interesting facts about green energy were scattered through the session. To vote for a particular answer, the participants had to place themselves in the suitable corner of the room (Fig. 3). The movement across the class room as well as the surprising answers to the questions led to a motivated and active climate.
The session on “Greenwashing”5 highlighted the creation strategy for a seemingly green image for companies and products by having the students develop their own “green” product. Teams of three to four students had to select a classic climate polluter, such as transatlantic flights or fuel-hungry cars, and promote them to be particularly “green”. The most promising advertising campaign developed in this setting coined the slogan “Go green, go Porsche!”. “Sustainability in Legislation” assessed how politics are addressing the matter - by enforcing norms, as well as sponsoring campaigns lobbying for a more sustainable lifestyle. One example, which was presented as a video was the sustainability rap “Guten Appetit” (“Enjoy your meal!”) by the group Rapucation, shedding a critical light on the practices of the food industry as well as on consumers’ attitude towards nutrition.

In the session “Sustainable Products”, participants were asked to form teams and to develop a software which would render a coffee machine more sustainable. When discussing, the students noticed that sometimes sustainability as requirement caused conflicts with other goals, such as a discount price, and trade-offs had to be made.

5.2 Master Seminar TUM: RE for ICT-Systems for Environmental Sustainability

The MSc seminar “RE for ICT-Systems for Environmental Sustainability” was also conducted as a block seminar, but with introductory talks by members of other faculties who also focus on sustainability in their research.

Table V sums up the organizational facts of the TUM Master seminar.

Table V. The TUM Seminar in Numbers

<table>
<thead>
<tr>
<th>Target Group</th>
<th>Master Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Students (actual/max.)</td>
<td>3/12</td>
</tr>
<tr>
<td>Number of Advisors</td>
<td>3</td>
</tr>
<tr>
<td>Deliverables</td>
<td>Presentation, Interactive Session, Essay</td>
</tr>
<tr>
<td>Number of Meetings</td>
<td>5</td>
</tr>
<tr>
<td>ECTS credits</td>
<td>4</td>
</tr>
</tbody>
</table>

Results. The talk by forester Andreas Hahn gave insights into a domain very far from traditional software engineering application domains, the domain of forestry, which led to an interesting discussion session on different viewpoints for sustainability.

The work on modeling sustainability in business processes was the best result. The student chose a vilification plant as case study and sketched the different interests of the stakeholders related to the aspects of sustainability they concerned and how that potentially influences the business processes.

The presentations can be accessed via the seminar’s webpage6.

5.3 Master Seminar UPC: Environmental Sustainability in Software Engineering

The MSc seminar “Environmental Sustainability in Software Engineering” emerged from a collaboration with Xavier Franch at the UPC and was held in November 2011. It is designed for two consecutive weeks of presence classes and, subsequently, a written essay as homework that is supervised and discussed via email and Skype. Table VI sums up the organizational facts. One detail worth noticing was that 3 professors took part in the seminar as it reflected some of their current research interests.

Table VI. The UPC Seminar in Numbers

<table>
<thead>
<tr>
<th>Target Group</th>
<th>Master Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Students (actual/max.)</td>
<td>7 (incl. 3 professors)/12</td>
</tr>
<tr>
<td>Number of Advisors</td>
<td>1</td>
</tr>
<tr>
<td>Deliverables</td>
<td>Presentation, Interactive Session, Essay</td>
</tr>
<tr>
<td>Number of Seminar Sessions</td>
<td>6 (2 h)</td>
</tr>
<tr>
<td>ECTS credits</td>
<td>3</td>
</tr>
</tbody>
</table>

The work plan was split into 4 phases:

— Phase 1: Week with three sessions of 2 hours, each including 45-60 min lecture and 30-45 min discussion and group work.
— Phase 2: Week with three sessions of 2 hours, each including short presentations by the students followed by discussion and group work.
— Phase 3: Written essay on an agreed specific topic with an assessment of related literature and analysis of potential future work and solutions.
— Phase 4: Mutual feedback for students and lecturer and assessment of the seminar and its results.

Results. Given the short time the students had to prepare their talks, they did very well and showed a lot of motivation in presenting topics they had just learned about. In addition, they had to present in a foreign language although questions could be clarified in their mother tongue.

The students performed a survey of potential ICT4ES systems and a respective classification of ICT application domains, analyzed life cycle assessment for its applicability to ICT4ES systems, compared the existing goal models for sustainability, and elaborated a small survey of metrics for sustainability and their applicability to software.

The presentations can be accessed via the seminar’s webpage7.

5The term “greenwashing” describes the behavior of interest groups, such as companies or lobbyists, advertising environmentally neutral or harmful products or services as being particularly “green” and sustainable.

6http://www4.in.tum.de/lehre/seminare/WS1112/RE4Green/

7http://www4.in.tum.de/lehre/seminare/WS1112/UPC-EnviroSiSE/
Jumpstart Sustainability in Seminars

6. DISCUSSION

Overall, we are satisfied with the results of the seminar. In the following paragraphs, we assess the outcome in more detail and present our lessons learned.

Scope. The topic “Sustainability in Software Engineering” has a wide scope and encompasses a variety of aspects from sustainability as quality objective in software design to sustainability in the processes that a software triggers in its application domain. We chose different scopes for the seminars, a wide one for the BSc and a more specific one for the MSc ones. Consequently, the essays ranged from an overview of related topics to an in-depth analysis of sustainability in software engineering.

Lesson Learned: The better defined the scope is, the more concrete the results will be. However, to explore a new topic in its full breadth, a wider scope might be necessary.

Content and Information Sources. There are many publications on the topic of sustainability. However, a large amount of the material is not suitable for scientific research. Occasionally, students had difficulties to filter for relevant content. One example we encountered during the BSc seminar was the “Blackle Hoax,” which was taken seriously by some participants. This incident stresses the necessity to train students in their attitude towards responsible research by advising them to question plausible arguments and verify their validity before seconding them. Regarding the topics (see Section 3), we came to realize that the level of abstraction required for an analysis of some of the topics was hard to achieve for some students. This resulted in partially superficial syntheses of the students’ analyses. On the other hand, some presentations and analyses exceeded our expectations.

Lesson Learned: Undergraduate students are not critical enough in judging retrieved information and we need to teach them this additional skill with more rigor.

Methods and Techniques. The participants appreciated the interactive didactical approach and found the strategy motivating. Even though the interactive workshop format was new to all of them, they dealt with it very well. The students were creative in their interaction techniques, integrating surveys, quiz breaks, visualizations, music, video, and group work in their workshop units. All participants were actively involved in discussion rounds and their general participation. Apart from the seminar supervision, we offered extensive feedback for each student, in which we reflected on their research as well as their didactical skills. This offer was met with great interest by all of them.

Lesson Learned: We will keep experimenting with interactive teaching methods. Both students and supervisors enjoyed it, it fostered a lively participation, and a good learning atmosphere during the sessions. Hopes are that the interactions will also create long-lasting learning effects.

Infrastructure. The wiki was not used as much as we had expected. Students referred to it only in order to look up the reading lists and to assess the scope of their topics in comparison to their colleagues. They did not show interest to add their own literature or to edit their theme page. One reason for this behavior might have been that literature had to be listed in their seminar essays anyway, so wiki participation might have seemed like extra effort.

Lesson Learned: A wiki is an online alternative for collaboration that is less necessary when the group is located at the same university. Students preferred communicating directly to their peers and supervisors instead of putting effort into their wiki pages. For distributed collaboration, a wiki might prove more helpful.

Target Group. To establish sustainability as a value in software engineering, we want to familiarize students with the concept in an early stage of their studies, which is why we targeted the first seminar at undergraduate students. This way, we attempt to lay the foundation for a further interest in the topic, as well as creating a general awareness of the impact of computing on sustainability. In the following term, we targeted the seminars at graduate students. In this way, we offered students from the first seminar a follow-up to further specialize on the topic. Additionally, we met the demand from graduate students, who requested a sustainability seminar for their course.

Lesson Learned: Undergraduate and graduate students can benefit from the seminar and be beneficial for their supervisors. Depending on their status in the curriculum they will have either little knowledge about independent research but more time left at the faculty to continue their work in subsequent assignments, or they will already have accumulated more knowledge but less time left at the faculty.

Effectiveness. Assessing the effectiveness of our seminar is not an easy task, as it is too early to see the desired long term effects such as continued interest in the topic, or a more thorough adoption in other courses. However, some factors suggest that a lot remains to be done to establish sustainability as an accepted value of software engineering. All students attending the seminar were highly motivated and interested: one part of the group considered sustainability as the “next big thing” whilst the other part was concerned about the environmental impact of the current technological development. In this regard, the seminar implicitly targeted intrinsically motivated students, which were already convinced of the relevance of the issue.

Lesson Learned: Motivation for the topic is essential—indeed of the specific target group. Therefore, we consider it necessary to integrate sustainability also into more established lectures in order to generate awareness also with students which have not yet discovered their interest in the subject. To achieve this, however, lecturers must be convinced of the relevance of sustainability in software engineering, which requires action also on the top level of academia, e.g. by the dean of the faculty.

7. CURRENT AND FUTURE WORK

Seminars are a good start to introduce the topic of environmental sustainability into software engineering education. The achieved impact is, however, comparatively small considering the total number of students in the department. Therefore, we attempt to trigger multiplier effects to spread the word and raise attention by means as presented in the following paragraphs.

Lesson Learned: A wiki is an online alternative for collaboration that is less necessary when the group is located at the same university. Students preferred communicating directly to their peers and supervisors instead of putting effort into their wiki pages. For distributed collaboration, a wiki might prove more helpful.

8 The Blackle Hoax claims that changing the background color of Google’s starting page from white to black could cause significant energy savings. However plausible this might sound at first, it is incorrect as the significant share of energy used by a search query is in the infrastructure needed to provide the response in a timely manner. http://www.blackle.com/
RE Case Study with Industrial Participants. We integrated an industrial case study with focus on sustainability in our requirements engineering lecture this winter term. A stakeholder from a big automotive company agreed to serve as customer for our students. The case study was on eMobility and one focus was on the requirements for environmental and social sustainability. The study is now in use in two Bachelor theses and a Guided Research.

Further Student Projects. To perpetuate the results of the seminars, we envision to offer student projects to develop products, such as smart phone applications for increasing everyday sustainability, newsletters, or short films on sustainability in software engineering, for subsequent use in advertising environmental sustainability within the department.

Increasing Visibility. To increase visibility in the faculty, we plan to trigger multiplier effects by, e.g., inviting SE lecturers to the seminar sessions, or filming the seminar presentations and showing excerpts on other occasions. Furthermore, we gather the results of the seminar to present them at a lectures series, or a guest lecture of an existing course.

Symposium. In July 2012, we are holding an interdisciplinary symposium on sustainability for faculty with a varied background, including informatics, sociology, business studies, philosophy, pedagogy, and political science. The focus is on the relation between informatics and sustainability and their mutual effects in an interconnected society.

8. GUIDELINE
To initiate similar activities at other universities, we suggest to:

(1) Consider our proposed plan for integration as first inspiration for own implementations [Penzenstadler and Fleischmann 2011].

(2) Choose a student target group to start a seminar. Examples for topics are given in Tab. I, II, and III.

(3) Find interested stakeholders from industry and ask them to be customers for a student project (Sec. 7).

(4) Organize a workshop to connect different disciplines working on sustainability throughout the campus.

9. CONCLUSION
This paper reports on our experiences with introducing sustainability into the software engineering curriculum. We have organized three seminars for Bachelors and Masters and there are further plans for integration into the software engineering lecture syllabus. The topic was met with great interest by our students, who also appreciated the workshop format of the seminars. Interactive didactical methods and techniques were positively taken on by the students and fostered active participation as well as constructive learning. The seminar format required extensive tutoring regarding the seminar topic as well as the didactical skills. This necessitates, in turn, that teaching staff is trained accordingly. Furthermore, sustainability as a major quality goal in software engineering needs to be integrated into established lectures to create awareness also in students who do not yet consider it as relevant. For this, lecturers need to be convinced to introduce sustainability in their courses over time.

REFERENCES

SPECIAL ISSUE SECTION: Distributed Software Development.


22 Case Study-based RE Education: Evaluation and Lessons Learnt
University Meets Industry: Calling in Real Stakeholders

Birgit Penzenstadler (co)  
Systems and Software Engineering, Technische Universität München, Germany  
penzenst@in.tum.de  

Martin Mahaux (co) and Patrick Heymans  
PRéCISE Research Centre, University of Namur, Belgium  
{martin.mahaux|phe}@fundp.ac.be

Abstract

Teaching the discipline of requirements engineering (RE) is slowly establishing at universities within the software engineering curriculum. While several studies have shown that case study-based education was more efficient in RE, many teachers are still reluctant to change their teaching style, and stay with classical lectures and complementary exercises. These courses often fail to relate the different steps and stages of RE to each other and do not address crucial communication and project management issues that are common in industrial RE practice. They also miss the chance for using the classroom as a near-to-real-settings research lab, and won’t show students the stakes existing in doing engineering in our society. We describe our experiences in teaching RE with a case study in two universities, achieving a triple-win: putting students in contact with real stakeholders, showing students their responsibility towards a sustainable world and doing empirical research in the classroom. We report on the course design, the evaluation, the lessons learned, and the potential success factors for such courses. We conclude that case study-based approaches to teaching RE considerably improve skills valued by industry, are feasible at a reasonable cost, and are enjoyable for the students, the teachers and the stakeholders. With this paper, we want to encourage RE educators to implement such courses in their setting.

1. Motivation: Case Study based Learning

According to Armarego and Minor [2], practitioners see shortcomings in formal education, particularly with respect to more generic skills such as communication, team skills and affective attributes of novices. Consequently, Armarego et al. provided a case-based Software Engineering (SE) course concept [1]. The results were mostly positive, showing that “students placed in an environment that enables them to model professional practice (...) should be much better prepared for the workplace”. Very recently, Hermann [12] analyzed over 200 job advertisements and confirmed that recruiters were more looking for soft skills (92%), in particular team work and communication skills (about 57% each) than for RE specific knowledge (37%).

Another positive aspect of case-studies with real stakeholders in the classroom is their ability to provide a near-to-real-settings lab for researchers, making winners for both education and research [15]. Li et al. report that, for the researcher, students feedback provides empirical evidence and opportunities to improve. For students, they underline the potential reflection-in-action
provided by experimental work, and Wong’s study confirmed that “an early exposure to actual research practice can help these students more quickly and effectively grasp the principles and goals of research in general” [23].

Finally, case studies with real-life problems have the ability to inform the students of their future responsibility as engineers. If the problem is well chosen, it may give the student a greater feeling of the impact of his/her work on the society as a whole.

**Problem:** Despite these very positive characteristics, it seems that such case-based courses have not been widely adopted in universities or practitioner’s courses. To confirm this intuition, we ran a quick poll through the RE-online mailing list, reaching several hundreds of RE teachers. We collected only nine answers, so could not conclude anything statistically. However, only four respondents reported running a case-study for more than half their course, and only one of the respondents reported doing so with real stakeholders. The number of students did not seem to have a strong impact on teachers decisions, as some teachers with big audience did use case-study (with role-played stakeholders) and some teachers with small audience did not. While we ignore the reasons for this low adoption, one can imagine that many teachers fear that such a course might be complex to design and implement, and costly in general, or even impossible given their own settings.

**Contribution:** In this paper, we report on running three instances of such a case-based RE course, at the Technical University of Munich (TUM) and the University of Namur (UN). The courses integrate RE lectures and/or readings with direct application in a real project, where students are expected to gather, model and write requirements for a to-be-defined system satisfying the needs and constraints of real stakeholders. The courses were used as a research ground for various research objectives. And the topic chosen was apt to show the responsibility of designing software for the society. The paper describes the courses in detail, compares them with one another and with other ways of teaching RE, and discusses the pros and cons for students, teachers, stakeholders and future employers. It offers insights on what we believe were success factors of the courses and on feedback by all parties.

**Impact:** The approach of teaching RE using real-life stakeholders and a real case study happened as described in the literature (for example, [1]), and we therefore expect that the gains for all parties involved was as described by this literature. Seen the relatively low cost of doing such courses, we strongly advocate for broader application of this concept in teaching. This paper offers three more positive experiences, and guidance for other RE teachers willing to implement case-based RE courses.

**Outline:** We report on related work (Sec. 2), describe the study design (Sec. 3) and the implementation (Sec. 4), and discuss the results (Sec. 5).

### 2. Related Work

There has been ample work showing that active or reflexive teaching modes are more desirable than passive ones [6, 14]. Furthermore, there are case-studies showing that case-based course are particularly desirable and effective in SE. Armarego and colleagues have proposed and analyzed the SE Design Studios mentioned above in an important body of work [1, 2]. Ghezzi and Mandrioli [11] consider the contextualization of SE courses an important challenge, as the difference between “learning by studying” and “learning by doing” is bigger in SE than in other disciplines. Richardson *et al.* [20] describe how problem-based learning can be used to accomplish both pedagogical and research-related goals in a single course module, but report the challenge of finding problems interesting to industry and manageable for students. Others report on successfully
including research into the classroom, e.g., Li et al. [15] and Wong et al. [23]. Furthermore, there are some experience reports with running case-based courses with real stakeholders [10, 13], while others describe how it is possible to cope with more students thanks to virtual, or role-played, stakeholders [9, 22, 4]. Most of these experiences perform the case study as a final year project, where the project is a test case for the acquired RE skills, and may also involve skills that can actually hardly be taught, for example unrealistic customer expectations and changing requirements as described by Barnes et al. [3]. In RE, course-related experience reports exist, covering various aspects of the discipline, such as requirements prioritization in the course proposed by Nancy Mead and implemented in class by Port et al. [19], or requirements negotiation, in courses described by Fernandes et al. [8] or Callele and Makaroff [5].

3. Study Design

Objectives: This course has objectives at several levels. It should firstly be a place for students to learn about RE. Secondly, it would offer the chance for a near-to-real-settings lab for our research. Finally, and perhaps in this objective we differ from previous similar initiatives, we took the chance to let students understand their responsibility in shaping tomorrow’s socio-technical systems.

For the educational purposes, we derived our objectives from the description of the industry’s expectations for the requirements engineer, as given in Armarego’s [1] and Macaulay and Mylopoulos’ studies [16]. The seven extracted key competencies and their learning objectives are listed in Tab. 1.

<table>
<thead>
<tr>
<th>Competencies</th>
<th>Learning Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical skills</td>
<td>Be able to articulate a problem and separate it from its solution, while understanding that both are interrelated and influence each other.</td>
</tr>
<tr>
<td>Abstraction skills</td>
<td>Be able to distinguish between various levels of abstraction in a requirements specification, and travel from one level of abstraction to another consciously.</td>
</tr>
<tr>
<td>Phrasing skills</td>
<td>Be able to write understandable requirements artifacts.</td>
</tr>
<tr>
<td>Communication skills</td>
<td>Be able to manage an efficient communication with a team of peers and with stakeholders.</td>
</tr>
<tr>
<td>Method competency</td>
<td>Understand the main steps of a requirements process, be able to follow and adapt a plan.</td>
</tr>
<tr>
<td>Reflective skills</td>
<td>Be able to reflect on one’s own work and adapt where needed.</td>
</tr>
<tr>
<td>Sensitivity for customer problems</td>
<td>Be able to listen truly to the customer.</td>
</tr>
<tr>
<td>Creativity</td>
<td>Be able to solve conflicts creatively and think out of the box.</td>
</tr>
</tbody>
</table>

Above these competencies, we also wanted our students to get an understanding of what RE is, and why it is important. They had to gain knowledge of some important terminology in RE, a sense of the fundamental steps of the RE process and the fundamental artifacts to be delivered. Finally, it is important that they get an initial mastering of some basic elicitation, analysis, specification, documentation, and quality assurance techniques.

For research purposes, our objective was to run experiments for the initial evaluation of techniques developed in our fields of research: creativity in RE and sustainability in RE, respectively.

Finally, for the responsibility aspect, we wanted to let students with a feeling of how socio-technical systems could affect the sustainability of human behaviour, and how RE was crucial in shaping this impact.

Study Plan: Similar to Design Studio [1], the core of our course is the execution of a relatively large case study in near-to-real-world settings. For this, we have invited industrial stakeholders into the classroom to present a high-level problem and its domain as starting point for the case study. Student teams were then given the task to write the requirements for a sub-problem that they
would define themselves. During the course, the students went through all the basic steps until the elaboration of specifications, including interviews, workshops and/or presentations to stakeholders. The main steps of the course are described in Tab. 2. Each step takes one to two weeks, at a rate of 3 to 4 hours a week, to reach a total of 50 to 60 hours in 15 to 16 weeks, representing a classical optional course during a semester at our universities.

Table 2. Study Plan for the “Sustainable Mobility” RE Case Study

<table>
<thead>
<tr>
<th>Step</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The students receive introductory lectures on RE by the professors plus reading material.</td>
</tr>
<tr>
<td>2.</td>
<td>The students participate in a team-building event (at UN only).</td>
</tr>
<tr>
<td>3.</td>
<td>The real stakeholders (local domain experts) give a one hour presentation on the challenge for the case study.</td>
</tr>
<tr>
<td>4.</td>
<td>The students receive their task assignment for the case study and are asked to form teams of 4-5 students.</td>
</tr>
<tr>
<td>5.</td>
<td>The students brainstorm and form ideas on potential systems useful for the challenge as they understood it.</td>
</tr>
<tr>
<td>6.</td>
<td>The students perform an interview with the main stakeholder(s) to get a deeper understanding of the problem.</td>
</tr>
<tr>
<td>7.</td>
<td>The students elaborate high-level requirements artifacts, with feedback from tutors.</td>
</tr>
<tr>
<td>8.</td>
<td>The students present the results in a workshop to the stakeholder(s) for feedback and initiate new artifacts with them during the workshop.</td>
</tr>
<tr>
<td>9.</td>
<td>The students rework their initial artifacts and develop additional artifacts to draft the complete requirements specification. Tutors provide feedback again.</td>
</tr>
<tr>
<td>10.</td>
<td>The students give a final presentation and receive feedback from the stakeholders and the professor (at TUM only).</td>
</tr>
<tr>
<td>11.</td>
<td>The students participate in the exams.</td>
</tr>
</tbody>
</table>

**Evaluation Plan:** In order to assess if the pedagogical objectives described above would be reached, we planned to combine several artifacts: the RE deliverables, the individual exams, and a reflexive “project diary” on the case study development that describes how the students proceeded and how they managed the various arising challenges. The RE deliverables had to be assessed for their closeness to real-world setting, as well as the quality of the delivered artifacts. This had to be done in conjunction with an analysis of the project diary. Indeed, students were allowed to make mistakes and be far away from real-world practice, as long as they were conscious of these lacks, that they would describe in the project diary to indicate their understanding of what would have been an ideal result. Those documents were cross-checked with informal feedback by the main stakeholders and by the students, as well as by tutors involved. Ideally, we would have liked to survey the employers of students who finally got hired as requirements engineers after our course. We could not practically do this, but managed to interview briefly some students working as requirements engineers after following the course. Finally, exams were planned to bring an individual level to this assessment, as the other measures were mostly at group level. Furthermore, we had to assess if we could run fruitful experiments for our research, and assess whether the topic chosen was apt to let students observe a potential impact of their work on a sustainability challenge.

4. Study Implementation & Results

Table 3 provides an overview of the course settings at both universities. In the following, we describe them in further detail, highlighting the peculiarities of the course instances at either university when needed. The main difference lies in the amount of time dedicated to the case-study versus lectures, where TUM had a much more balanced distribution and UN a stronger focus on the case.

**Course Deliverables:** Deliverables included requirements artifacts and a project diary, all available online1. The structure of the requirements artifacts differed slightly in both universities, but

1[http://www4.in.tum.de/~penzenst/sources/RealStakeholdersRECaseStudy.zip]
Table 3. Summary of the settings

<table>
<thead>
<tr>
<th>University</th>
<th>TUM</th>
<th>UN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsible</td>
<td>Prof. Manfred Broy</td>
<td>Prof. P. Heymans</td>
</tr>
<tr>
<td>Designer</td>
<td>Dr. Birgit Penzenstadler</td>
<td>Martin Mahaux</td>
</tr>
<tr>
<td>Tutors</td>
<td>Birgit Penzenstadler, Mario Gleirscher, Max Junker</td>
<td>Martin Mahaux, Nicolas Genon, Patrice Caire</td>
</tr>
<tr>
<td>Trainees</td>
<td>22 Master students Software Engineering</td>
<td>18/22 Master students, half Information Systems, half Management Sciences</td>
</tr>
<tr>
<td>Time frame</td>
<td>60 hours in class (28 lecture + 32 case-study)</td>
<td>50 hours (4 lecture + 46 case-study)</td>
</tr>
<tr>
<td>Course type</td>
<td>Optional course</td>
<td>Optional course</td>
</tr>
<tr>
<td>Real Stakeholder</td>
<td>Initiative Leader and Manager for “DriveNow” from BMW AG</td>
<td>HR Director from UN, Experts in Mobility, in Mobile Application Design, in Business Development</td>
</tr>
<tr>
<td>Project</td>
<td>case study for system already implemented</td>
<td>system possibly intended to be built</td>
</tr>
</tbody>
</table>

they converged in demanding various major RE artifacts, covering both problem and solution, from high to low level of detail. The requirements-related deliverables structure at TUM is represented in Fig. 1. In the case of UN, students were allowed to chose the template for the requirements artifacts, inspired from existing templates such as Volere [21]. Given the short deadlines, students were not required to be exhaustive in their deliverables, but to show a certain level of mastering various modeling and writing techniques. For example, rather than poorly writing all requirements, they were asked to efficiently write some of the requirements.

The project diary describes the experience of running the project as lived by the students. It highlights what was perceived as well done, and what should have been done differently. The project diary enables them to reflect on the effectiveness and efficiency of any activity undertaken, and document particular insights. For example, several teams had been conducting their interview with the stakeholders not listening to them but defending their vision of the problem. This experience was debriefed with tutors, and students indicated in their project diary where they failed and how to improve. The project diary also included raw student notes from interviews and workshops, plus the documents to prepare those exercises.

**Informal Assessment of Experience and Lessons Learned:** In this section, we report informally on how the course did reach its triple objective. We report in particular on the learning made by the students, as the evaluation plan permitted. While more formal analysis might have been desirable, the size and complexity of both the experiments and the deliverables, along with a short time for tutors dedicated to meta-analysis, did not allow for delivering any more significant results than a report by the tutors. Consequently, most of what follows reflects tutors perception only, however backed with as many objective statements as we could in an *a posteriori* analysis.

**Students:** According to feedbacks from both students and tutors, the three instances of this course were an enjoyable experience. Most students showed motivation, as indicated by the additional work put into the exercise by most of the students compared to what was expected to simply
succeed. We had the chance to interview four students from Namur that we knew had started their career as requirements engineer in the industry. Their answers gave the course positive ratings with regard to its usefulness for their career pursuit. In particular, students strongly agreed that the course encouraged them to start a requirements engineer career. They also agreed that the course helped them in getting the job. At TUM, none of the students who responded had yet started their professional career, but the answers indicated that some of them considered a career in requirements engineering and the majority thought the acquired skills were going to help them in their future career. The formal evaluation of the course showed an improvement from 2.3 to 1.8 on the rating scale (where 1 was best and 5 was lowest).

Artifact Quality: The final deliverable exhibited a considerable quality for the short timespan for their elaboration. In particular, the students creativity, i.e. their ability to produce useful and novel work, was explicitly high. For example, two years after the first requirements documents were written in Namur, we are observing start-ups and authorities developing applications that are mostly in the scope of what was then a students’ work. In other words, they were visionaries in developing dynamic car-sharing applications for solving mobility problems. Some other artifacts were clearly not of industrial quality, nor of the quality that tutors would expect at the end of an exercise session in a classical lecture & exercise setting. However, 32 to 46 hours to fulfill the given task would certainly be judged impossible by any industry person. Students were aware that this kind of assignment would require several months of full time work for an analyst, as the individual exams confirmed (this was one question at Namur’s exam). Students efficiently distributed the work among them, spontaneously using various online collaboration tools. Finally, they reached a deliverable exhibiting many of the characteristics of a valid industry requirements work, in a fraction of the time. The quality of individual artifacts sometimes suffered from this, but students were able to identify low quality. Indeed, a discussion with the teacher during the exam would in general reveal that they were able to enhance the poor artifact by themselves.

Success Rate: All students passed the final exam with good grades, meaning tutors and professors had a positive perception of students’ ability with regard to the fixed didactic objectives. It has to be noted that the involved professors have a longer experience in giving RE courses than the tutors having designed and managed the course. They consequently could compare to what was done before, and this is an indication, while certainly subjective, of the efficiency of the course.

Competencies: With regard to the analytical skills, method competency, and abstraction skills, students were “on the right way” in the case study deliverables, i.e. they made mistakes but learned from them and achieved overall reasonably good results in the final exams. Their phrasing skills and communication skills were quite diverse at the beginning, and we noted a significant improvement between the first and the following workshops and presentations with the stakeholders. Naturally, there were different levels of how serious students took the case study. One team showed up in suits and another one spent days on elaborating an animated video pitch for the product while others just did the most necessary. However, all of the teams engaged in active discussion with the stakeholders and received good feedback. Their reflective skills were enhanced over the time of the case study and they understood how the communication had worked and which were the critical factors for writing a good requirements specification (e.g., more feedback questions for stakeholder instead of assumptions for fuzzy requirements). The communication practice over the course of the workshops also increased their sensitivity to customer problems, which was reflected in the oral exams at Namur, when students were put into typical RE situations and had to act accordingly.

Stakeholders: According to their own statements, stakeholders did not perceive they had lost time in participating in interviews and workshops. As far as the topics were chosen to their best
interest, the only chance for not getting them to at least learn something in the process was to have students behaving in a really non-professional way. In case this was observed, tutors could quickly intervene to avoid such negative situations. In general, stakeholders were happy to have participated in this experience and learned about current RE methods as well as about potential business ideas.

**Research:** Various techniques under experimentation at our lab were tested during the course. In Namur, students spent two sessions doing team building using improvisational theatre techniques [17]. As this technique is supposed to have an influence on how stakeholders can be creative within a group, we filmed group work before and after the team building. This way, we attempted to assess if the team building had an impact. While those laboratory settings are not perfect, they are certainly interesting as this kind of process is very hard to do in real industry settings.

In Munich, students were asked to specifically focus on the aspect of the environmental sustainability of the project. This work formed the basis for a case study for a generic sustainability model [18]. The case study is available online [7].

**Responsibility:** This exercise is one of the first time that students are facing a design problem that looks like what real life can offer. We believe this is one of the first time that they can really feel their potential impact on the world as solution makers. Consequently, we consider important that this is not ignored by the teachers when they choose the subject. Including the sustainability dimension in this exercise was an important point for us, and we think it also helped in motivating the students, who generally consider this as an important element, and were probably happy to see that engineers and managers do not necessarily have to work for banks. Probably few among the students had really thought about the impact of socio-technical systems on sustainability. The case-study forced them to realize this impact, and this is certainly an interesting learning for the coming generation of solution designers.

**Effort & Costs:** The main effort that needs to be conducted in comparison to other forms of teaching requirements engineering is to find one or several industrial stakeholder(s) willing to participate. However, tutors don’t have to prepare unitary exercises, and the reduced number of classical lectures reduces the number of slides to be prepared. In our cases, we got access to the stakeholders by using our extended personal networks. We found it quite easy to find in our networks someone with an IT-related challenge that required the design of a relatively undefined solution, and that included elements of sustainability. In Namur, we also invited IT-related stakeholders. There are plenty of these available in our faculties. One expert in business creation from regional authorities had also spontaneously declared interest to be a stakeholder.

The effort for stakeholders is relatively small. In our cases, students require an initial presentation that is made in a plenary session (1h). Then each group needs to have one or several interviews, preferably two or three, with different kinds of stakeholders if possible. If all groups go to see the same people, those people might end up with N hours of interview, with N being the number of groups. As stakeholders are also invited to a 1 hour workshop per group, this adds N hours of work for each stakeholder. This limits the possible number of groups to 4 to 8, to limit the total workload to 8 or 16 hours for most stakeholders (one more for the sponsor doing the first presentation). In the second instance of the course in Namur, however, we left it over to the students to decide who they would interview, and to arrange the interviews themselves, inviting a tutor for observation. This had a reasonable success, all groups found at least one person to interview, and some found
five of them. This way of doing reduces the work for tutors and distributes the stakeholders workload. This might be a way to sustain this course model in bigger classrooms. A good option to acquire partners might be former students who are now in industry and are applying requirements engineering themselves.

Before the course, the tutors have to make an initial planning, as if they were to do the requirements engineering themselves for the project at hand. This requires some understanding of RE in practice, that we hope any teacher has. During the course, tutors must gather every week to assess students advancement in the project, fine tune the coming week’s schedule and prepare corrective actions when needed. There is important communication involved with students as to guide them on a day-to-day basis. Clarifying and communicating expectations as the experience go is crucial in these settings. During sessions, tutors must be available for answering student’s questions, delivering just-in-time knowledge when needed. This requires that tutors master their subject quite well, as questions that arise from practicing in near-to-real settings might be far from what tutors may expect in classical exercise settings.

5. Discussion

Benefits: Although relying on informal evaluation, we believe that the main expected benefits from the case-based teaching style have been reached in our course instances. Students had a nice course and developed an idea of their future responsibility as designers, while tutors could use the setting for valuable research. Further, the costs were clearly under control, and to our opinion comparable to preparing an in-depth theoretical course with exercises.

Drawbacks: These courses are heavily dependent on various factors. While the outcome of classical courses is more controlled, here it is highly dependent on students, stakeholders and tutors. In particular the latter have to have an idea of RE in practice, and should be able to handle tough questions. They have to have some contacts in the industry, and be influential enough to get them accept an invitation to work with students. They have to have some project management skills, and flexibility in particular. Stakeholders, on their side, have to be interested in what is passed on to students, as they are not as effective as experienced requirements engineers. The TUM stakeholders’ interest in participating was also driven by finding new business ideas.

One issue is how close to industry settings we can actually get in such a case study. The stakeholders are real, but they certainly do not perceive the project as crucial for success as their regular projects. Further, the short time frame does only allow for a discovery of the main steps of a requirements engineering, not for an in-depth real requirements work. However, our students demonstrated impressive abilities to work in teams under short time frames, which is a frequent challenge in the industry.

The availability of stakeholders is a limiting factor, to both the number of students that can be handled in such settings, and the time spent by each student in contact with stakeholders. While we highlight a possible way to scale up, by allowing students to find their stakeholders themselves (some stakeholders are not rare, such as users having a smart-phone and potentially participating in car-sharing), we don’t know if this would be successful for a high number of students (over 40). However, the lack of stakeholders availability also reflects a real project situation.

Finally, we believe our course would not be as effective as a classical course for teaching more deeply specific hard techniques for the requirements engineer, such as UML or other modeling. In Namur, this is the subject of another course, that we reuse here without putting the emphasis on correctness of models.
**Success Factors:** Retrospectively, we tried to understand what made this course a success. The following represents our consolidated perception of the main success factors: 1. **Freedom** Students were encouraged to think by themselves about what to do and how to do it. There was no pre-defined solution, and students could mostly choose their techniques and tools, or even not to use any of them. The matter of importance was to criticize the process and the outcome. 2. **Feedback** All along the exercise, students were encouraged to submit partial deliverables. Tutors gave constructive feedback on deliverables without grading them. Tutors were available during sessions, and on other days as well, in case of any questions. Errors were more than permitted; students were judged less on the quality of the final deliverable than on what they could understand from their errors. 3. **Real problem** The cases were based on real-life problems, including trendy challenges (sustainability), required real creativity to find new solutions. While the problem and stakeholders are real, ensuring realism, there was not such a high pressure on the results as the real life would confer, allowing for safe learning. The emphasis is not on the details of some specific technique but rather on the ability to use them in a realistic setting. 4. **Teams** Teams of 3-5 worked well in that everyone can participate, and collaboration is rich as teams benefitted from mixing backgrounds. Team building session at the beginning of the course are a plus, certainly if students don’t know each other (coming from different faculties in Namur). 5. **Fun** We had fun during the exercise, as tutors tried to setup a relaxed atmosphere for allowing spontaneous interaction.

**Threats to Validity:** When we started designing our courses, each at our university, our objective was to set up a nice course for our students; it was not to make an experiment we could write a paper on. Incidentally, discussing together about RE teaching at a conference, we decided to jointly replicate and report about the course. After these three iterations, we believe that we have reached a certain level of consolidation, allowing to present this paper as a valuable experience report. However, as reporting was not our primary objective, even in the following iterations of the course, we lacked resources to collect the needed evidence for a formal analysis. Consequently, the report is mainly informal, representing the most important threat to validity of this study. We reported our perceptions, basing them on as many objective facts as possible. To limit the potential researcher bias arising from this, we stress this paper does not claim any personal invention from the authors. We simply report on using concepts from others that worked for us, and had no personal interest in showing a particular success in this paper.

6. **Conclusion**

In this paper, we describe our experience from three RE courses where students at two universities performed case studies in cooperation with real stakeholders. We confirm that this type of course challenges and improves skills and competencies in the students that are highly valued by industry. Both students and involved stakeholders enjoyed the course and benefitted from its results, as did researchers using the course as a near-to-real-settings lab. As, on top of that, the involved costs are reasonable, we strongly encourage all RE educators to consider including such a case study in their courses. Future work is, inter alia, to demonstrate the impact of RE on sustainability.

**Acknowledgment**

The authors would like to thank Prof. Manfred Broy for his support, the participating students and industry partners, and Marco Kuhrmann for helpful feedback on earlier versions of this paper.
References

Using Non-Profit Partners to Engage Students in RE
Using Non-Profit Partners to Engage Students in RE

Birgit Penzenstadler, Debra Richardson
School of Computer and Information Sciences
University of California, Irvine, CA, USA
bpzenzs@uci.edu

David Callele
Experience First Design Inc.
Saskatoon, Canada
dcallele@experiencefirstdesign.com

Beth Karlin
School of Social Ecology
University of California, Irvine, CA, USA
bkarlin@uci.edu

Allison Cook
Story of Stuff Project
Berkeley, CA, 94709, USA
allison@storyofstuff.org

Krzysztof Wnuk
Blekinge Institute of Technology
SE - 371 79, Karlskrona, Sweden
Krzysztof.Wnuk@bth.se

Abstract—To improve requirements engineering education and training, experience reports serve as guidance on how courses can be taught and which methods and approaches work with specific types of audiences.

One problem when teaching undergraduate audiences is often a lack in motivation for the course because of misconceptions about requirements engineering as well as simple work overload by the curricula or other imposing constraints with regard to the students’ time budget.

A way to overcome this motivational problem and to make students want to actively contribute to a requirements engineering class project is to let them perform a case study on a socially relevant system in collaboration with an industrial partner.

This paper provides an experience report of such an in-class project on an online-learning platform for civic engagement in cooperation with the Story of Stuff Project. We provide the structure of the course as well as the assignments, excerpts from the students’ results, and observations made throughout the course, all of which may serve as input for other instructors.

Index Terms—requirements engineering; requirements education; environmental sustainability; civic engagement;

I. INTRO

Requirements Engineering (RE) as a discipline provides the foundation to making successful software systems by eliciting the appropriate information from the relevant stakeholders and by offering the methodological means to analyze and document the findings such that they can be incorporated throughout design and implementation.

In the undergraduate course on RE at the University of California, Irvine, we want to integrate the theory in class with as much real-world experience as the classroom constraints allow for. Therefore, we integrate case studies and examples from ongoing software system development and collaborate with industry and/or non-profit partners to give the students an impression of RE in practice.

Web content has become an increasingly common way of disseminating important information to people and engaging them in social causes [12]. The challenge of educating people online in civic engagement, as developed by the Story of Stuff Project, provides a well-suited example case study for students to learn these techniques and to experiment with different ways of achieving a consensus among a variety of stakeholders from a non-engineering background.

With the majority of our students being interested and responsible citizens (as emerging data suggests that Millennials are interested in doing more purpose-driven work or just being “socially-conscious” [5]), we predicted that using a case study on empowering citizens would engage them more thoroughly with the responsibilities of requirements engineering.

Aims and Outline: This paper reports on experiences gathered with using the Citizen Muscle Boot Camp (CMB), an online learning program currently under development at the Story of Stuff Project for environmental activism, as student project in a requirements engineering course for undergraduate students at the University of California, Irvine. The goal of the Boot Camp is to train a subset of the 500,000 members of the Story of Stuff community into citizen activists to work toward social change on sustainability issues. After introducing the project, we present the course details, how the students performed the assignments, and the observations from their results.

II. BACKGROUND

A. The RE Undergraduate Course at UCI

The undergraduate course in RE at the University of California, Irvine, has been taught by a number of different lecturers with each slightly adapting the style and focus to personal preferences and current research and practice. In the spring quarter of 2014, it was taught by the first author of this paper.

One of the definitions for RE that is used in the course is the one by Nuseibeh and Easterbrook [13]: Requirements engineering is concerned with interpreting and understanding stakeholder terminology, concepts, viewpoints and goals. Hence, RE must concern itself with an understanding of beliefs of stakeholders (epistemology), the question of what is observable in the world (phenomenology), and the question of what can be agreed on as objectively true (ontology). Such issues become important whenever one wishes to talk about validating requirements, especially where stakeholders may have divergent goals and incompatible belief systems [13].

1 http://www.storyofstuff.org
Furthermore, RE facilitates the process of consolidating different stakeholders’ concerns and agreeing on a system vision [11]. The major learning goals that arise from this view and shall be incorporated in the course are the following:

1) Knowledge areas: RE terminology RE contents, RE principles and methods for eliciting, analyzing, specifying, validating, verifying requirements, and quality assurance
2) Competencies and skills: analysis, abstraction, phrasing, communication, sensitivity for customers, method competencies

The course occurs over 10 weeks, with two lectures per week. Student evaluation is performed by a mid-term and final exam as well as a number of team assignments. The assignments are designed to add up to a complete requirements specification, with students working in a team distributing the work and synthesizing and consolidating individual parts as well as taking responsibility for quality assurance.

B. Student Body

The course is mandatory for undergraduate students from three different major subjects, namely Informatics, Computer Science, and Business and Information Management. Due to the fact that it is a required course, there are many students each year who have to participate, and this made for a large course of 106 enrolled students.

All have had programming experience in their prior coursework, while some have had more extensive experience in software development either in their own projects or in internships.

C. Lectures and Course Materials

The lectures were structured along an outline that followed the paradigm of artifact-oriented requirements engineering, i.e. that used the artifact model presented in Sec. II-D as a backbone to elicit and analyze requirements and to organize the other tasks involved with requirements management. Each lecture was 80 minutes long and the lecture slides were available online beforehand.

Preparation materials for the case study were provided in the form of background information on the Story of Stuff Project as introduced in Sec. III-A and Sec. III-B as well as some general background on Massive Open Online Courses (MOOCs) [9] with examples2. Students were asked to familiarize themselves with the Story of Stuff material as well as with examples for MOOCs.

D. Artifact-based RE with AMDiRE

In winter 2014, we introduced AMDiRE [7], an artifact-oriented approach to RE, in the course. The basic idea of artifact orientation consists in defining a reference model of all relevant artifacts and their dependencies while leaving open the way of their creation. The focus thus lies on what to create rather than on how to create it. In RE, there exist several artifact models, such as the one of Berenbach et al. [4, ch. 2], who describe RE artifact modeling with the key components to be a measurable reference model and respective process guidelines. To tackle the problem of a blurry terminology and to foster the discussions about this paradigm, Mendez et al. introduced a meta model for artifact-based RE [10]. This meta model defines the basic concepts of artifact-based RE, i.e. which elements are necessary to define an artifact (structure, content), or how an artifact relates to further software process concepts like “method” or “role”. This supports the systematic creation of artifact-based RE approaches covering all elements of software processes, and thus the integration and customization of an artifact-based RE as part of a software process.

The Artifact Model for Domain-independent Requirements Engineering (AMDiRE) [7], [2], developed on the basis of that meta model, served as framework for the lectures and structured the team project into weekly assignments.

E. Related Work

In the field of sustainability education, Karlin et al. [8] report on a pilot study of the Guided Research Applied Sustainability Project (GRASP) model for sustainability education that provides students with a positive and engaging learning experience.

In RE education, Zowghi and Paryani [16] used role playing to bring real world experiences into the classroom. Penzenstadler and Callele [14] report on experiments with bringing a stakeholder from mechanical engineering into class. Barnes et al. [3] discuss how to foster an attitude of acceptance in their students versus resistance and fear of the unknown and unknowable in RE by including real-world examples and experience in class.

Penzenstadler et al. [15] report on bringing stakeholders from industry into the classroom. In contrast, the course described in the paper at hand did not include interviews with the stakeholders in class but instead information via documents and online research.

III. SOS CMB PROJECT

This section describes the problem domain, the study system, and the student assignments.

A. Problem Domain: The Story of Stuff Project

Story of Stuff started as a film project in 2007 designed to tell the story of “stuff (i.e. consumer goods) from its creation, through its sale and use, and eventually to its disposal” in a catchy and engaging manner. The single film, with an initial viewership goal of 50,000 views, quickly exceeded this goal and sparked the launch of The Story of Stuff Project as a 501(c)3 nonprofit organization. The Story of Stuff Project has since created 8 additional films and has also translated into other mediums, including television (interstitials for PBS), print (Story of Stuff book), online (website, social media, podcast), and curricula (K-12 and religious groups). Their combined films have been translated into 39 languages and

2https://www.coursera.org/
seen by over 40 million people and counting. This viewership has also translated into an active online community of nearly a half million people. The initial goal of the Story of Stuff Project was to create and provide media resources to environmental activists and educators as well as to the general public. A community of potential activists formed around the organization, who started to want to become more actively engaged in the issues discussed in the films.

B. Study System: The Citizen Muscle Boot Camp

To address this need for positive citizen engagement through education, The Story of Stuff Project created a new program in 2013 called the Citizen Muscle Boot Camp. The Citizen Muscle Boot Camp (CMB) is a six-week, online course designed to provide Story of Stuff Project community members with the skills, motivation and peer support they need to act on issues related to environmental sustainability. It was designed in response to inquiries and requests from the Story of Stuff community to help members learn, engage, connect, and act on the issues raised in their films. The CMB guides participants through a series of weekly trainings aimed at strengthening their civic activism skills, or “citizen muscle”. Each week of the program focuses on a unique skill:

1) Purpose: discovering your change making style and goal
2) Talk: learning how to communicate effectively about your issue
3) Grow: finding and developing a community of allies
4) Focus: getting strategic about how to accomplish your goals
5) Push: figuring out which tactics you’ll need to employ to effect change
6) Practice: putting your learning into action.

For a first pilot of the course, members of the Story of Stuff community were contacted via email and invited to register for the CMB free of charge. Those who registered were enrolled in the CMB and received an email with a link to a 2-3 minute video lesson, accompanied by additional resources (e.g., framing, tips, readings) to get them practicing their change-making skills and a homework activity to put their new skills into practice. The CMB is designed so that each weekly module can be completed in 1-2 hours per week, and it was chosen as study system for the students to complete their assignments and develop a requirements specification.

C. RE Course Assignments

This section provides the assignments for the students as well as the evaluation criteria that were used to grade the submissions. The students were divided into 26 teams of four to five students. They were free to choose their teams within the first week of the quarter and the students who had not found teams by then were assigned to one. Each team was serving as customer for one of the other teams, and as a developer for a different team. The timeline of the assignments is depicted in Table I. We used a reduced version of the AMDiRE artifact model as depicted in Fig. 1.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Percentage</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Case Analysis</td>
<td>5%</td>
<td>Jan 27</td>
</tr>
<tr>
<td>Stakeholder Model</td>
<td>5%</td>
<td>Jan 27</td>
</tr>
<tr>
<td>Goal Model</td>
<td>5%</td>
<td>Feb 3</td>
</tr>
<tr>
<td>System Vision</td>
<td>5%</td>
<td>Feb 3</td>
</tr>
<tr>
<td>Mid-term exam (20%)</td>
<td></td>
<td>Feb 6</td>
</tr>
<tr>
<td>Domain Model</td>
<td>5%</td>
<td>Feb 10</td>
</tr>
<tr>
<td>Usage Model</td>
<td>5%</td>
<td>Feb 10</td>
</tr>
<tr>
<td>Non-functional requirements</td>
<td>5%</td>
<td>Feb 17</td>
</tr>
<tr>
<td>Functional hierarchies</td>
<td>5%</td>
<td>Feb 24</td>
</tr>
<tr>
<td>Peer review</td>
<td>5%</td>
<td>Mar 3</td>
</tr>
<tr>
<td>Presentation of specification</td>
<td>5%</td>
<td>Mar 10</td>
</tr>
<tr>
<td>Final Exam</td>
<td>30%</td>
<td>Mar 20</td>
</tr>
</tbody>
</table>

TABLE I: TIMELINE OF ASSIGNMENTS AND GRADING SCHEME

1) Business Case Analysis and Stakeholder Model: For this assignment, the students had to perform an elicitation meeting with their team’s customer group to discuss the case study problem in their role as part of the development team. After the elicitation meeting where they discuss with the customer their understanding and vision of the Story of Stuff Citizen Muscle Boot Camp Online Course, the team develops a business case analysis and a stakeholder model, both as described and discussed in the lecture.

Evaluation criteria:

- Business Case Analysis [40 points]:
  - Is the analysis structured well?
  - Does it contain executive summary, problem statement, analysis, solution options, project description, cost-benefit analysis, risks and recommendations?
  - Is each of these elements described in sufficient detail?
  - Is the information consistent throughout the document?

- Stakeholder Model: [40 points]
  - Have all major stakeholder groups been considered?
Have they been analyzed and described to an adequate degree of detail?
Are the relationships between the stakeholders clear?
Is it clear which stakeholder has which knowledge, skills, priority, and responsibilities?

2) Goal Model and System Vision: For this assignment, the students had to ensure that they captured the critical goals from their elicitation meeting with the customer. If this is not the case yet, they have to further communicate with them and elicit more goals. On that basis, the team develops a goal model and a system vision, both as described and discussed in the lecture. They had to make sure that the goal models contain at least three to four levels of subgoals for more than half of the goals\(^3\) and that appropriate notations are used (goal categories, labeled relations, AND/OR alternatives). The task was not only to submit the diagram but also a textual explanation of how they did it and why they did it this way. This also gives them a chance to report on encountered challenges.

For the system vision, the scoping (system boundary) is important, as well as other systems in the context and the involved stakeholders. It was important to keep the intention of a system vision in mind: It shall communicate the idea of the project to all stakeholders in a way they can agree on and that is easily understandable without technical knowledge.

Evaluation criteria:
- **Goal Model [40 points]**
  - Is the goal model well structured?
  - Does it contain at least five business goals, five usage goals and five system goals?\(^4\)
  - Are they broken down and refined into subgoals where possible?
  - Is each of these elements related sufficiently to the other goals and are all notations used?
- **System Vision: [40 points]**
  - Is a clear system boundary / scope visible?
  - Is the vision described and illustrated to an adequate degree of detail?
  - Is it clear which systems are in the operational and business context?
  - Is it clear which stakeholders are involved?

3) Domain Model and Usage Model: For this assignment, the students use the input from the earlier content items (business case analysis, stakeholder model, goal model, system vision) including the feedback they received on those to develop a domain model and a usage model, both as described and discussed in the lecture. It was encouraged to make sure that the domain model contains the important concepts of the domain — business objects, real world objects, and events that transpire — in form of classes (at least seven), attributes (at least two per class), and associations with roles and multiplicities. Furthermore, it was again requested to not only submit the diagram but also a textual explanation of how they accomplished the task. For the usage model, they were asked to provide an overview diagram of all use cases, plus a detailed version of at least one use case, including the full information from the template in the lecture slides. The detailed version of one use case includes one scenario — first described according to the Cockburn template [6] (as main success scenario with extensions and/or variations)—and then choosing one diagram type (activity diagram or message sequence chart) to illustrate it. They were asked to provide a description of the rationale for the domain model and usage model, at least two paragraphs of how they did it and what they found difficult or the most challenging aspect of it.

- **Domain Model [40 points]**
  - Is the domain model well structured?
  - Does it contain at least seven classes and two meaningful attributes per class?
  - Are all classes connected by associations with roles and multiplicities?
  - Is the domain model complete w.r.t. the criteria discussed in class? Does it provide all important concepts?
- **Usage Model: [40 points]**
  - Is a use case overview diagram provided that is well structured and includes all important use cases?
  - Are all use cases that are important for the system depicted in that diagram?
  - Is a complete and correct description of one exemplary scenario provided for one central use case in either a UML activity diagram or a message sequence chart?
  - Is a complete and correct description provided for one central use case in a table (according to the Cockburn template)?

4) Non-functional requirements: For this assignment, the students again had to use input from the earlier content items (goal model, domain model, usage model) including the received feedback to develop a small set of non-functional requirements. This should be provided in the form of two examples of each of the following categories: process requirements, deployment requirements, system constraints, and quality requirements. These requirements shall be refinements of the earlier elicited goals, so they were asked to name the goals as rationale.

Evaluation criteria for non-functional requirements [80 points]:
- Is the template filled out completely and correctly for process requirements?
- ... for deployment requirements?
- ... for system constraints?
- ... for quality requirements?

5) Function hierarchy: For this assignment, the students again had to use the input from the earlier content items (usage model and scenarios). On that basis, the team develops a function hierarchy according to the example from the lecture,
structuring them into functions and subfunctions and adding the respective relationships between them.

Evaluation criteria for the Function hierarchy [80 points]

- Are there at least ten functions in the function hierarchy?
- Does the structure of the hierarchy make sense?
- Are at least three of the possible types of relations (precedes/follows, requires, interrupts, activates/deactivates) made explicit?
- Are all relations depicted that exist between the displayed functions or are relations missing?

6) Quality Assurance in Peer Review: For this assignment, students used the input provided by their development team, meaning that:

1) They send their complete specification to their customer in one PDF file (consolidated version of all your assignments in one document) and receive the one from their development team.

2) The team writes up a review of the specification of their development team according to the following IEEE-830 criteria: completeness, consistency, unambiguity, correctness, structuredness, traceability, changeability, understandability

Evaluation Review [80 points]: Is there a rating and rationale for the rating for the above criteria, namely Completeness, Consistency, Unambiguity, Correctness, Structuredness, Traceability, Changeability, and Understandability?

IV. RESULTS & OBSERVATIONS

All teams completed the assignments and handed in their solutions. They also provided the rationale on how the results were developed and which challenges had been encountered while performing the assignment. This section provides illustrative examples of good solutions (not all from the same team) and a discussion of observations made throughout the course and while reviewing the results.

A. Business Case

Figure 2 is a screenshot of the rather elaborately designed business case brochure one of the teams created. The hardest part in the business case was that the system was for a nonprofit organization. This led some of the student teams to inventing options for voluntary donations while taking the online course, but made it hard to develop an actual business case with convincing numbers apart from trying to keep the costs low. The business case in Fig. 2 is a good, concise solution. While the layout effort is quite impressive, that part of the effort could have gone to something more directed to the task at hand. Constraining the students to focus on content over presentation seems to be useful in such an early and short course. However, this is one of the three “fancy” examples out of 26 solutions, so most of the students did stick to a simpler presentation format. Apart from that, students put a lot of effort into researching some background statistics on the web that would back up their data in the business case.

The challenge of finding sufficient background information is not so much related to learning the actual techniques of requirements engineering, but very much in the day-to-day practice of business analysts who also perform requirements engineering.

B. Stakeholder Model

The stakeholder model in Fig. 3 is a simple yet sufficiently extensive representation of the major stakeholders of the CMB system. Students often neglected representing the relations between stakeholders, as for example in Fig. 3, where there...
are arrows to “hold together” the figure but no labels on the arrows that would indicate the actual relation between the stakeholders.

From the discussion on the solutions afterwards, the major challenge was to infer relations if the checklists and examples provided in class didn’t already include those relations.

C. Goal Model

The goal model in Fig. 4 depicts business goals (in red), usage goals (in green), and system goals (in blue) and their relations. The model is structured to support a goal hierarchy with respect to the scope of the goal. Business goals are more primary, relating to the actual purpose of creating the system, so they are located toward the top of the diagram. They are supported by usage goals which define the intent and function of the system and are thus supported by system goals which demonstrate the characteristics of the system. The model contains a multitude of antigoals, or constraints, as well as obvious goals. Anything that is not marked with the double minus sign is an uninhibiting subgoal of its parent goal. Each constraint is marked with a double minus sign (- -) meaning it can inhibit the functionality or achievement of its parent goals. Double plus sign (++) means the subgoals lead up to their parent goal.

Major challenges were to distinguish between business, usage, and system goals and to identify relations between the goals.

D. System Vision

Figure 5 is a pictorial representation of the system vision of the CMB that all stakeholders (in this case, customer team and developer team members) agree on and that serves as common reference point in discussions. The system vision was agreed upon by all stakeholders (i.e., each developer team and the respective customer team) in order to define the functionality and characteristics of the Story of Stuff CMB. The vision contains passive and active stakeholders centering around the CMB and a business and operational context, separated by a dotted line on the diagram. Most active stakeholders are highlighted by colored fields. Community leaders and users are among the active stakeholders, but they are not members of the Story of Stuff organization. Each group is highlighted according to the legend on the diagram. Passive stakeholders, along with community leaders and users, are not highlighted by any boundary.

The major challenge with the stakeholder model was to consider all categories of stakeholders and to determine influences between them.

E. Domain Model

The domain model in Fig. 6 is a class diagram that lists the most important business (real-world) objects that have to be represented in the system. The classes and attributes needed to not only follow what was written in the business case and the goal model, but also had to be consistent with the usage model. The classes are associated with each other when they have direct interaction with each other.

The students encountered many challenges including but not limited to: having consistent terms between the domain model and previous documents or models; determining whether classes were necessary or arbitrary; and figuring out if an attribute was part of the correct class. They resolved the terminology issue by discussing the model with their teammates and coming to a consensus as to which terms to be used in the model. As for the classes, they decided which ones should become classes and which ones should remain attributes while creating the diagram.

F. Usage Model

Figure 7 and Figure 8 depict excerpts of the usage model, representing different paths of how a user can interact with the website. In the activity diagram (Fig. 8), the user is placed in the middle as a way to symbolize their role in determining whether or not data would be transmitted between the system.
Goal Model

The model above is structured to support a goal hierarchy with respect to the scope of the project. The system vision for the SoS CMB involves creating a Massive Open Online Course. The implementation team writes the code and creates the MOOC itself. The number of registered users increases by 10% every year for 5 years. Good lectures are also available.

Legend
- Business Goal
- Usage Goal
- System Goal
- Primary Goal

Fig. 4. Goal Model for the SoS CMB

Fig. 5. System Vision for the SoS CMB
and the account database. The students also considered the failure condition, in which an account is not created. Including these two end conditions is necessary to demonstrate the destruction of data upon failure.

The most challenging aspect of designing activity diagrams was determining when and where data flow occurs, as the students had difficulty in defining the interaction between the system and the accounts database.

### G. Non-functional requirements

The non-functional requirements in Fig. 9 provide examples of a simple template specification form used to detail the NFRs and their validation. The non-functional requirements included quality requirements, process requirements, deployment requirements, and system constraints.

The students reported slight difficulty in eliciting non-functional requirements, along with phrasing them in a way that was coherent with the goals. They referred mostly to their goal models, breaking down each element in order to determine some non-functional properties their client prescribed for the system. The NFRs themselves were rather high-level quality goals broken down to a level that could be applied to the overall software system. The measurement (see Fig. 9, 4th attribute of every table) was often underspecified so that testers would have to make assumptions about how exactly to measure this satisfaction criterion.

#### H. Functional hierarchies

The functional hierarchy in Fig. 10 decomposes the user-perceived functionality from a system point of view and thereby facilitates the transition to design. While the students listed out each function, they tended to think too broadly and in general terms about the CMB, instead of the specific requirements that were requested by the clients. They found it difficult to determine the subfunctions of certain parent functions and figuring out whether activates, precedes, or requires was more suitable for describing how certain functions interacted with each other in the hierarchy. Another problem that they encountered was making the functional hierarchy diagram readable. The user was required to go through the “signin to account” function before being allowed to use the rest of the functions in the MOOC system. It was the subfunction to most of the other functions, so there were many arrows pointing to that function and many boxes surrounding it. As a result, they chose to make the “signin to account” function “activate” all the other functions, which solved the problem and made the diagram much easier to read.

Early design decisions like this were the most discussed point about function hierarchies and show where RE and design have to be integrated or may overlap.

#### I. Peer Review

In the peer review, students tended to be very generous to their developer teams. They commented on a few minor mistakes or how the specifications could have been extended, but in general they were satisfied with the work of their peers. Most of the teams had gone through the effort of reworking their specifications after the initial feedback from the teaching assistants, so those efforts apparently had paid off, as the specifications had improved considerably.

#### J. Presentation of the Specification

The teams presented excerpts of their solutions within five minute presentations during the last two sessions of the course before the final exam. At this stage of their curriculum, the students do have experience in presenting in front of a large
audience, but their presentation skills varied considerably. While some teams made an effort to make their presentation engaging, others only put in the minimum required effort to deliver an acceptable summary. Presentation skills are of critical importance to successful customer communication and should therefore be part of an RE course.

V. DISCUSSION AND LESSONS LEARNED

A. Overall Quality of Results

The range of solutions extends from bad to good, as well as from little effort to major effort that was put into the development of the specification.

The grades in the mid-term and the final exam were average for a RE course, not much difference was perceived from other years (comparison as perceived by teaching assistant) or from earlier teaching experience by the first author at other universities. However, this is not dependent on the teaching method but on the students’ population — badly designed course will generate failures but well designed courses may also generate failures if students are not motivated.

B. Team Dynamics

There is a strong correlation between the students level of experience and the quality of the work that they produced on the project. Students who had experience in software development projects and even worked in industry managed their teams in a very effective way and provided medium to high quality specifications. On the downside, they often strongly focused on the technical aspects of the system (solution-orientation) and neglected stakeholder communication (problem-orientation).

There was a perceivable difference between the groups that were self-selecting and the groups that were assigned in terms of the work they produced. Teams who had self-selected were usually composed of friends who therefore had established communication habits, while new teams first had to get to know each other and find common denominators.

C. Solution Space

The design space used by the students was rather limited compared to what will actually be implemented. Were the students simply not interested enough or not knowledgeable enough? Or were the re-documenting what they saw elsewhere without considering other options? This depends partially on their experience, but also on their creativity and motivation for coming up with new ideas in this course.

Having chosen a problem domain that would likely increase motivation (see Sec. III-B), we were interested in how much sustainability showed up in their results. The answer is: only in System Vision and Goal Model, if at all. Some even reduced it only to the economic sustainability of the business. It is hard to determine whether this is due to the lack of familiarity with requirements techniques, due to lack of interest in the specific concern for the system under development, or due to a lack of knowledge of sustainability issues. Other disciplines that are also trying to incorporate questions on sustainability into existing structures, like UC Santa Cruz’s engineering program [1] have reported a high motivation amongst students for sustainability causes, therefore we conclude that this is mainly due to inexperience of how to factor such a value into RE.

D. Lessons Learned

This section sums up the lessons learned of the experience report at hand.

Reduce Number of Assignments: Effort estimation in student projects is hard, but as first-timers every artifact takes more time to elaborate and team work requires a lot of discussion if done properly. The conclusion is to ask for less artifacts when
the course has only 10 weeks—it was too much stuff to deal with in such a short time—and to probably cut out function hierarchies because they are strongly design-oriented.

**Real Stakeholders Motivate:** While the students liked having a “real system” to work on, they would have preferred to talk to a real person involved in the project as opposed to having to rely on documents, online research, and future users. Consequently, for the next iteration of the course, we will again put a real stakeholder in class (compare to the BMW DriveNow case study [15]) for one or two interviews during elicitation and analysis, even if they are playing a role. According to our experience, a real person who is only available for that specific time, considerably increases the motivation to understand the problem domain and system under development.

**Choice of Problem Domain:** There are challenges when motivating the pragmatic topic of RE with a project that is, itself, motivated by what might be perceived as an ethical choice. Does the ethical choice resonate with the students (positive motivator), is it ignored (neutral, but not achieving your personal goal to enlighten them), or does it annoy them? When looked at from a software system perspective, RE is about the content to be represented in the system, not the system purpose. The question of whether or not the content of the project enhances or detracts from the mechanics of the course material is hard to answer without back-up by significant empirical data. It was perceived that the students were enjoying the topic of the problem domain but there might have been students who were negatively motivated by the topic without providing feedback on that. Next time we will include a question on this in the course evaluation survey.

**Value-based Design:** Sustainability is not a concern that visibly showed up in the solutions other than in the goal model and system vision. Environmental sustainability was used as example for a value that could be supported in value-based design. While it was not the intention of the course to promote any political agenda, this was a value that students would generally be willing to accept and that was a little easier to grasp than a value like “fun”. The fact that the value was not made very explicit in the solutions shows that for practicing any kind of value-based design, this needs to be put into an exercise more explicitly, for example by making different teams design for different values and comparing the differences amongst their solutions.

VI. CONCLUSION & FUTURE WORK

This paper reported on the experiences with using an online course, the Citizen Muscle Boot Camp, under development at the Story of Stuff Project, as study system for the assignments in an undergraduate RE course at the University of California, Irvine. The Story of Stuff Project used the results as additional input for the actual development of the CMB. The major lessons learned are that artifact-oriented RE works well to structure assignments, but also needs to be restricted to the given time frame of the course, that real-world systems do motivate students, but even stronger motivation and commitment to developing a good requirements specification can be achieved by inviting real stakeholders, and that the choice of the problem domain might influence students’ motivation. There are a few follow-up questions to be explored in future work, for example, whether value-based design should be included as a concept when teaching requirements engineering, and how significantly the choice of problem domain affects the motivation of the students in such a course. For the next iterations of the course at UCI, we will try to allow the students to perform interviews with real stakeholders for a system under consideration in practice.

Acknowledgements: This work is supported by the DFG EnviroSiSE project under grant number PE2044/1-1.

REFERENCES


[9] Rita Kop. The challenges to connectivist learning on open online networks: Learning experiences during a massive open online course. The International Review of Research in Open and Distance Learning, Special Issue-Connectivism: Design and Delivery of Social Networked Learning, 12(3), 2011.


Part VI

Bibliography
Bibliography


