Infusing Green: Requirements Engineering for Green *In* and *Through* Software Systems

Birgit Penzenstadler
University of California, Irvine
Irvine, CA, USA
bpenzens@uci.edu

***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 effect [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, Arnfalk, 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 Systems[1] [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.

[1]http://www.ics.uci.edu/~bpenzens/2014re4susy/

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 i*-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 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].

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

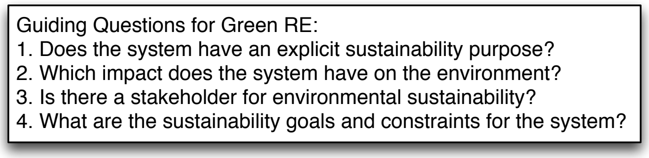
Guiding Questions for Green RE:
1. Does the system have an explicit sustainability purpose?
2. Which impact does the system have on the environment?
3. Is there a stakeholder for environmental sustainability?
4. What are the sustainability goals and constraints for the system?

Fig. 1. Guiding Questions for Green Requirements Engineering

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.

[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

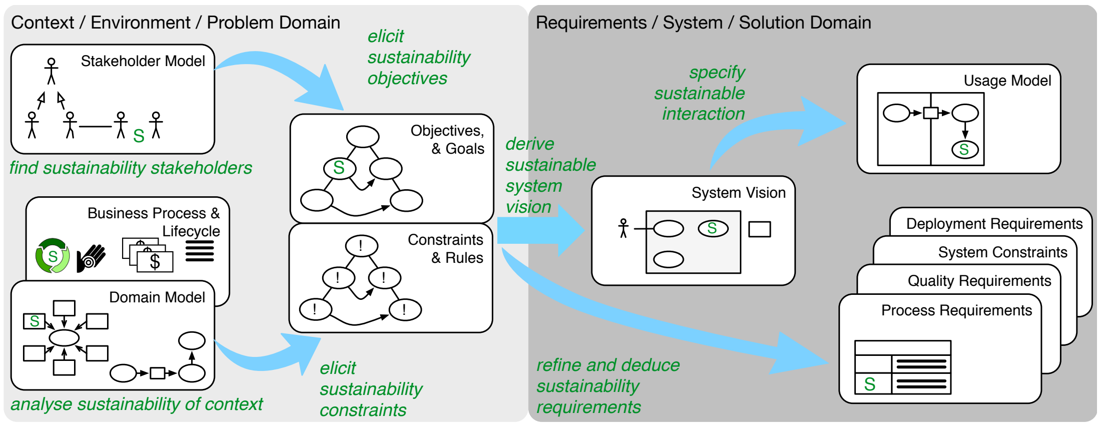
Context / Environment / Problem Domain
Requirements / System / Solution Domain
Stakeholder Model
S
find sustainability stakeholders
elicit sustainability objectives
Objectives, & Goals
S
Business Process & Lifecycle
S
$
Domain Model
S
analyse sustainability of context
Constraints & Rules
!
elicit sustainability constraints
derive sustainable system vision
specify sustainable interaction
Usage Model
S
System Vision
S
Deployment Requirements
System Constraints
Quality Requirements
Process Requirements
S
refine and deduce sustainability requirements

Fig. 2. Overview of Content Items and Information Flow for Green Requirements Engineering

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 *eliciting 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 activites *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 this 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.

[2]http://coolclimate.berkeley.edu/carboncalculator
[3]http://storyofstuff.com/
[4]http://www.zipcar.com/
[5]http://www.drive-now.com/

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.

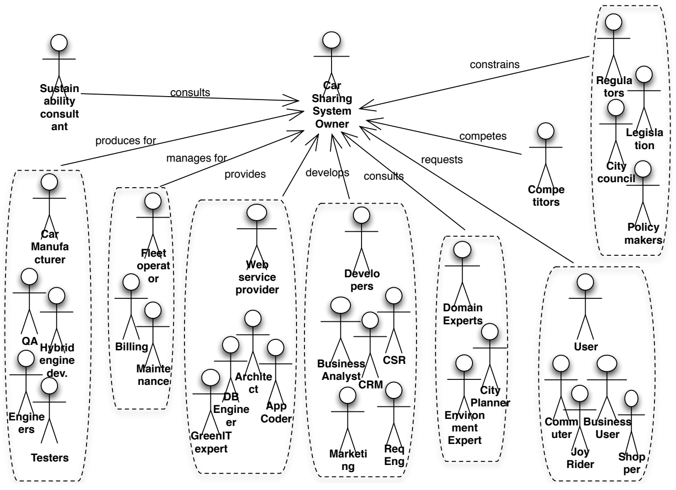

Fig. 3. Stakeholder Model of the Car Sharing System

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* 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.

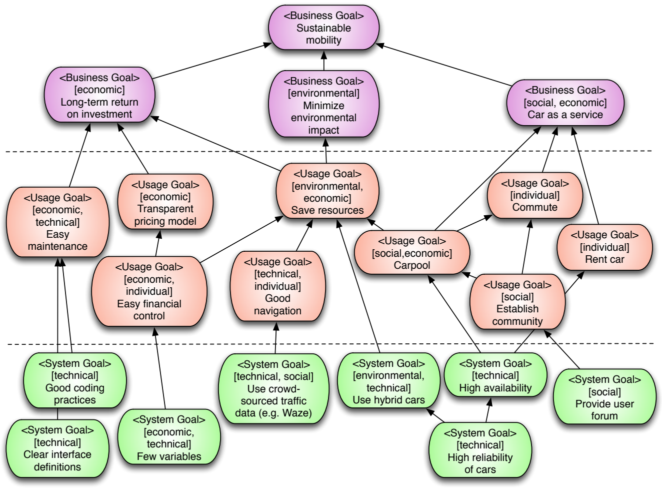
<Business Goal> Sustainable mobility
<Business Goal> [economic] Long-term return on investment
<Business Goal> [environmental] Minimize environmental impact
<Business Goal> [social, economic] Car as a service
<Usage Goal> [economic, technical] Easy maintenance
<Usage Goal> [economic] Transparent pricing model
<Usage Goal> [environmental, economic] Save resources
<Usage Goal> [individual] Commute
<Usage Goal> [social,economic] Carpool
<Usage Goal> [individual] Rent car
<Usage Goal> [economic, individual] Easy financial control
<Usage Goal> [technical, individual] Good navigation
<Usage Goal> [social] Establish community
<System Goal> [technical] Good coding practices
<System Goal> [technical, social] Use crowd-sourced traffic data (e.g. Waze)
<System Goal> [environmental, technical] Use hybrid cars
<System Goal> [technical] High availability
<System Goal> [social] Provide user forum
<System Goal> [technical] Clear interface definitions
<System Goal> [economic, technical] Few variables
<System Goal> [technical] High reliability of cars

Fig. 4. Sustainability Goals of the Car Sharing Case Study

In the car sharing example, the objectives in the three categories were elicited by considering the dimensions of sustainability and how they can be reflected with regard to the system. Figure 4 depicts an excerpt of the goal model for the car sharing system. The overall goal of *Sustainable mobility* is broken down into subgoals (e.g. *Minimize environmental impact*) and these are refined into usage goals (e.g. *Save resources*), which are again broken down (e.g. *Carpool*) and/or refined into system goals (e.g. *High availability*). The square brackets denote the sustainability dimensions of the goal.

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).

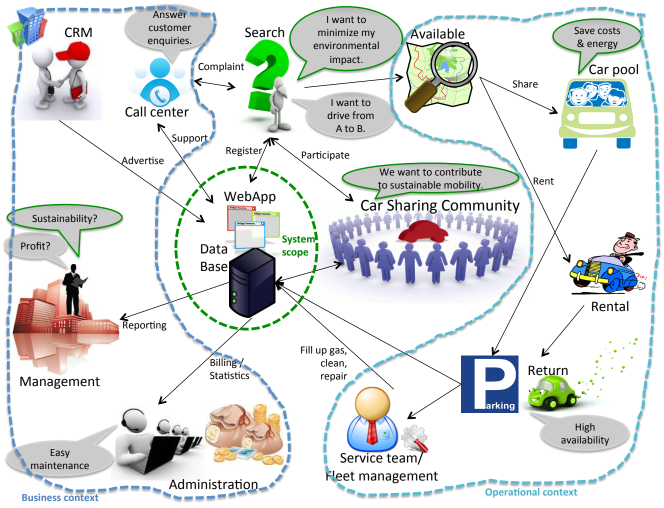

Fig. 5. The Car Sharing System Vision

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.

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.

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.

Example *Process Requirement for the car sharing system*

- *Develop the system according to the guidance provided by Software Engineering for Sustainability (SE4S) and as described in the book "Green Software Engineering".*

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.

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.*

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.

Example *System constraint for the car sharing system*

- *The web application shall use Waze as navigation system on handheld devices.*

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].

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)*

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 *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 I
REQUIREMENTS CONTENT ITEMS AND THEIR SUSTAINABILITY DIMENSIONS

Content Item	Sustainability Dimension
Stakeholder Model	all
Goals	all
System Vision	mainly environmental and social
Usage Model	social and economic
Quality Requirements	technical
Deployment Requirements	technical
System Constraints	environmental and technical
Process Requirements	environmental, social and technical
Risk List	all

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 exists 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+20[6], 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

[6]http://www.uncsd2012.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 sustainability[7] 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].

[7]http://www.se4s.org

Acknowledgements

I would like to thank Ankita Raturi, Debra Richardson, Daniel Mendez, Henning Femmer, Alejandra 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 EnviroSiSE project under grant number PE2044/1-1.

References

[1] Henrikke Bauman and Anne-Marie Tillman. *The Hitch Hiker's Guide to LCA: An Orientation in Life Cycle Assessment Methodology and Applications*. Professional Pub Serv, 2004.

[2] Simon Bell and Stephen Morse. *Sustainability indicators: measuring the immeasurable?* Earthscan, 2008.

[3] Berntsson Svensson, R. et al. 18th Intl Working Conf on Requirements Engineering: Foundation for Software Quality. Proc of the Workshops RE4SuSy, REEW, CreaRE, RePriCo, IWSPM and the Conference Related Empirical Study, Empirical Fair and Doctoral Symposium. ICB Research Reports 52, University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB), 2012.

[4] Uwe M Borghoff and Johann H Schlichter. *Computer-supported cooperative work*. Springer, 2000.

[5] Paul Burger and Marius Christen. Towards a capability approach of sustainability. *Journal of Cleaner Production*, 19:787–795, 2001.

[6] J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, and J.-N. Mazon. Integrating sustainability in decision-making processes: A modelling strategy. In *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 207 – 210, May 2009.

[7] Coral Calero and Marco Piattini, editors. *Green in Software Engineering*. Springer, to be published 2014.

[8] Marius Christen and Stephan Schmidt. A formal framework for conceptions of sustainability — a theoretical contribution to the discourse in sustainable development. *Sustainable Development*, page DOI: 10.1002/sd.518, 2011.

[9] A. Classen, P. Heymans, and P.-Y. Schobbens. What's in a Feature : A Requirements Engineering Perspective. In J. Fiadeiro and P. Inverardi, editors, *Proceeding of the 11th International Conference on Fundamental Approaches to Software Engineering (FASE 08) in Conjunction with ETAPS 08*, number 4961 in FASE/ETAPS, pages 16–30. Springer-Verlag Berlin, 2008.

[10] A. Cockburn. *Writing Effective Use Cases*. Number ISBN-13: 978-0201702255. Addison-Wesley Longman Publishing Co., Inc., 2000.

[11] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner. Software Quality Models: Purposes, Usage Scenarios and Requirements. In *Proceedings of the 7th international workshop on Software Quality (WoSQ 09)*, page N/A. IEEE Computer Society Press, 2009.

[12] Premkumar T. Devanbu and Stuart Stubblebine. Software engineering for security: a roadmap. In *The future of software engineering*, pages 227–239. ACM Press, 2000.

[13] Andrew Dobson. Environment sustainabilities: An analysis and a typology. *Environmental Politics*, 5:401–428, 1996.

[14] Oliver Feldmann. Sustainability Aspects in Specifying a Car Sharing Platform. Bachelor's thesis, Technische Universität München, 2012.

[15] Martin Glinz and Roel J. Wieringa. Guest editors' introduction: Stakeholders in requirements engineering. *IEEE Software*, 24(2):18–20, 2007.

[16] R. Goodland. *Encyclopedia of Global Environmental Change*, chapter Sustainability: Human, Social, Economic and Environmental. Wiley and Sons, 2002.

[17] Qing Gu, Patricia Lago, and Simone Potenza. Aligning economic impact with environmental benefits: A green strategy model. In *First International Workshop on Green and Sustainable Software*, 2012.

[18] Richard Heeks. Ict4d 2.0: The next phase of applying ict for international development. *Computer*, 41(6):26–33, 2008.

[19] Lorenz Hilty, Roland Hischier, Thomas F. Ruddy, and Claudia Som. Informatics and the Life Cycle of Products. In *iEMSs 2008: International Congress on Environmental Modelling and Software*, 2008.

[20] Lorenz Hilty, Wolfgang Lohmann, and Elaine Huang. Sustainability and ICT — an overview of the field. In *Proceedings of the EnviroInfo 2011*, 2011.

[21] Lorenz Hilty, Wolfgang Lohmann, and Elaine Huang. Sustainability and ict—an overview of the field. *POLITEIA*, 27(104):13–28, 2011.

[22] Lorenz M. Hilty, Bernard Aebischer, Göran Andersson, and Wolfgang Lohmann, editors. *ICT4S 2013. Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, ETH Zurich, February 14-16*. ETH Zurich, University of Zurich and Empa, Swiss Federal Laboratories for Materials Science and Technology, 2013.

[23] Lorenz M Hilty, Peter Arnfalk, Lorenz Erdmann, James Goodman, Martin Lehmann, and Patrick A Wäger. The relevance of information and communication technologies for environmental sustainability. *Environm. Modelling & Software*, 21(11):1618 – 1629, 2006.

[24] International Standardization Organization. ISO 14000 - Environmental management, 2004. http://www.iso.org/iso/home/standards/management-standards/iso14000.htm.

[25] International Standardization Organization. ISO 26000 Guidance on social responsibility, 2010. http://www.iso.org/iso/home/standards/iso26000.htm.

[26] S. Islam. Software Development Risk Management Model - A Goal Driven Approach. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundation of Software Engineering (ESEC / FSE)*, pages 5–8, New York, NY, USA, 2009. ACM Press.

[27] S. Islam, S. Houmb, D. Mendez Fernandez, and M. Joarder. Offshore-Outsourced Software Development Risk Management Model. In *Proceedings of the 12th IEEE International Conference on Computer and Information Technology (ICCIT 09)*, pages 514–519, 2009.

[28] Michael Jackson. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Addinson-Wesley, 1995.

[29] William Stanley Jevons. *The coal question*. Macmillan and Co. London, 2nd edition, 1866.

[30] Björn Johansson, Anders Skoogh, Mahesh Mani, and Swee Leong. Discrete event simulation to generate requirements specification for sustainable manufacturing systems design. In *Proceedings of the 9th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '09, pages 38–42, New York, NY, USA, 2009. ACM.

[31] Tom Kelley and David Kelley. *Creative Confidence: Unleashing the Creative Potential Within Us All*. Crown Business, 2013.

[32] Susan Krumdieck. The Survival Spectrum: The key to Transition Engineering of complex systems. In *Proceedings of the ASME International Mechanical Engineering Congress & Exposition*, 2011.

[33] Robyn Lutz. Software engineering for safety: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, pages 213–226. ACM, 2000.

[34] Georgina Mace. The Limits to Sustainability Science: Ecological Constraints or Endless Innovation? *PLoS Biol*, 10(6), 2012.

[35] Martin Mahaux, Patrick Heymans, and Germain Saval. Discovering Sustainability Requirements: an Experience Report. In *17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2011.

[36] Donella Meadows. Leverage points — places to intervene in a system. http://www.donellameadows.org/wp-content/userfiles/Leverage_Points.pdf, 1999. A shorter version of this paper appeared in Whole Earth, winter 1997.

[37] Donella H. Meadows. *Thinking in Systems - A primer*. Earthscan, 2008.

[38] Desta Mebratu. Sustainability and sustainable development: historical and conceptual review. *Environmental impact assessment review*, 18(6):493–520, 1998.

[39] Daniel Mendez and Birgit Penzenstadler. Artefact-based Requirements Engineering: The AMDiRE Approach. *Requirements Engineering Journal*, to appear 2014.

[40] Andrew Monk and Steve Howard. Methods & tools: the rich picture: a tool for reasoning about work context. *interactions*, 5(2):21–30, 1998.

[41] Stefan Naumann, Markus Dick, Eva Kern, and Timo Johann. The greensoft model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, 1(4):294 – 304, 2011.

[42] B. Nuseibeh and S. Easterbrook. Requirements Engineering: A Roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46, New York, NY, USA, 2000. ACM.

[43] Birgit Penzenstadler. Supporting sustainability aspects in software engineering. In *3rd International Conference on Computational Sustainability (CompSust)*, 2012.

[44] Birgit Penzenstadler and Henning Femmer. A Generic Model for Sustainability with Process- and Product-specific Instances. In *First Intl. Workshop on Green In Software Engineering and Green By Software Engineering*, 2013.

[45] Birgit Penzenstadler, Henning Femmer, and Debra Richardson. Who Is the Advocate? Stakeholders for Sustainability. In *2nd International Workshop on Green and Sustainable Software (GREENS, at ICSE, San Francisco, USA)*, 2013.

[46] Birgit Penzenstadler, Martin Mahaux, and Camille Salinesi, editors. *Second International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)*, 2013. http://ceur-ws/vol-995.

[47] Birgit Penzenstadler, Ankita Raturi, Debra Richardson, Coral Calero, Henning Femmer, and Xavier Franch. Systematic Mapping Study on Software Engineering for Sustainability (SE4S). In *18th Intl. Conf. on Evaluation and Assessment in Software Engineering*, 2014.

[48] Birgit Penzenstadler, Ankita Raturi, Debra Richardson, Coral Calero, Henning Femmer, and Xavier Franch. Systematic Mapping Study on Software Engineering for Sustainability (SE4S) — Protocol and Results. Technical Report UCI-ISR-14-1, Institute for Software Research, University of California, Irvine, 2014. http://isr.uci.edu/publications.

[49] Birgit Penzenstadler, Ankita Raturi, Debra Richardson, and Bill Tomlinson. Safety, security, now sustainability: The non-functional requirement of the 21st century. *IEEE Software Special Issue on Green Software*, 2014.

[50] Birgit Penzenstadler, Bill Tomlinson, and Debra Richardson. Re4s: Support environmental sustainability by requirements engineering. In *International Workshop on Requirements Engineering for Sustainable Systems*, 2012.

[51] Ankita Raturi, Birgit Penzenstadler, Bill Tomlinson, and Debra Richardson. Developing a sustainability non-functional requirements framework. In *accepted for GREENS'14*, 2014.

[52] K.-H. Robert, B. Schmidt-Bleek, J. Aloisi de Larderel, G. Basile, J.L. Jansen, R. Kuehr, P. Price Thomas, M. Suzuki, P. Hawken, and M. Wackernagel. Strategic sustainable development — selection, design and synergies of applied tools. *Journal of Cleaner Production*, 10:197–214, 2002.

[53] David Stefan, Mark Barrett, Emmanuel Letier, and Mark Stella-Sawicki. Goal-oriented system modelling for managing environmental sustainability. In *International Workshop on Software Research and Climate Change (WSRCC)*, 2010.

[54] Joseph Tainter. Social complexity and sustainability. *Journal of Ecological Complexity*, 3:91–103, 2006.

[55] Joseph Tainter. Sustainability. Personal communication, February 14, 2014.

[56] The Club of Rome. 40 years 'limits to growth'. http://www.clubofrome.org/?p=326, 2012.

[57] Mark Thompson. Ict and development studies: Towards development 2.0. *Journal of International Development*, 20(6):821–835, 2008.

[58] Bill Tomlinson. *Greening through IT*. MIT Press Association, 2010.

[59] United Nations. Report: Our Common Future. In *Conference on Environment and Development*, 1987.

[60] K. Wiegers. *Software Requirements*. Number ISBN-13: 978-0735618794. Microsoft Press, Redmond, WA, USA, 2nd edition, 2003.

[61] World Business Council for Sustainable Development. Vision 2050: A New Agenda for Business. http://www.wbcsd.org/WEB/PROJECTS/BZROLE/VISION2050-FULLREPORT_FINAL.PDF, 2010.

[62] Pamela Zave. Classification of research efforts in requirements engineering. *ACM Computing Surveys*, 29(4):315–321, 1997.