Everything is INTERRELATED: Teaching Software Engineering for Sustainability

Birgit Penzenstadler
CSU Long Beach
Long Beach, USA
birgit.penzenstadler@csulb.edu

Stefanie Betz
Karlsruhe Insitute of Technology
Karlsruhe, Germany
stefanie.betz@kit.edu

Colin C. Venters
University of Huddersfield
Huddersfield, UK
c.venters@hud.ac.uk

Ruzanna Chitchyan
University of Bristol
Bristol, UK
r.chitchyan@bristol.ac.uk

Jari Porras
LUT
Lappeenranta, Finland
jari.porras@lut.fi

Norbert Seyff
FHNW
Windisch, Switzerland
norbert.seyff@fhnw.ch

Leticia Duboc
La Salle - Ramon Llull University
Barcelona, Spain
lduboc@salleurl.edu

Christoph Becker
University of Toronto
Toronto, Canada
christoph.becker@utoronto.ca

ABSTRACT

Sustainability has become an important concern across many disciplines, and software systems play an increasingly central role in addressing it. However, teaching students from software engineering and related disciplines to effectively act in this space requires interdisciplinary courses that combines the concept of sustainability with software engineering practice and principles. Yet, presently little guidance exist on which subjects and materials to cover in such courses and how, combined with a lack of reusable learning objects. This paper describes a summer school course on Software Engineering for Sustainability (SE4S). We provide a blueprint for this course, in the hope that it can help the community develop a shared approach and methods to teaching SE4S. Practical lessons learned from delivery of this course are also reported here, and could help iterate over the course materials, structure, and guidance for future improvements. The course blueprint, availability of used materials and report of the study results make this course viable for replication and further improvement.

CCS CONCEPTS

• **Software and its engineering**; • **Applied computing → Education**; • **Social and professional topics → Sustainability**;

KEYWORDS

Sustainability, software engineering, pedagogy, sustainability design, sustainability education

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on thefi rst page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICSE-SEET'18, May 27-June 3 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5660-2/18/05…$15.00
https://doi.org/10.1145/3183377.3183382

ACM Reference Format:
Birgit Penzenstadler, Stefanie Betz, Colin C. Venters, Ruzanna Chitchyan, Jari Porras, Norbert Seyff, Leticia Duboc, and Christoph Becker. 2018. Everything is INTERRELATED: Teaching Software Engineering for Sustainability. In *ICSE-SEET'18: 40th International Conference on Software Engineering: Software Engineering Education and Training Track, May 27-June 3 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3183377.3183382

1 INTRODUCTION

Sustainability is a major concern to humanity as a result of the consequences of the rapid consumption of the planetsfi nite natural resources, combined with exponential economic and population growth [32, 33, 52]. It is suggested that to achieve a sustainable future it is highly dependent on the creation of a professional workforce knowledgeable about sustainable practices and processes to optimize resource management and influence human activity on the environmental, economic and social aspects of sustainability [2]. While it is postulated that societal change begins with the individual, evidence suggests that students often appear to be unable to align their demonstrated unsustainable behavior with their values related to sustainability [38, 47]. As such, there is a need to create a *cognitive dissonance* to bridge the *commitment gap*, which it is argued can be achieved through *educational interventions* to encourage individuals to effectively balance the self-knowledge that motivates intentional personal development towards more sustainable behavior [12, 46].

Thefi eld of computing also plays a critical role in addressing sustainability given its high societal leverage [14]. However, undergraduate computing education often fails to address our social and environmental responsibility [30]. Despite long standing calls [26], computing education has been slow to act towards a shift in adopting sustainability education resulting in a deficit and misalignment in knowledge in how existing basic software engineering theory and practice relates to sustainability [17]. While the concept of sustainability has gained worldwide mainstream traction in the higher

education sector at an operational level [23, 24, 51, 53], few have addressed education for sustainability in a holistic, multidisciplinary, and systematic manner [29]. However, in a crowded computing curriculum [31], software engineering students have little chance to learn about the concepts needed to analyze sustainability beyond the technical scope of systems they develop in class. This requires interdisciplinary courses that combine a wider perception and understanding of sustainability with software engineering projects through reflective practice. Here sustainability[1] is interpreted as the capacity of a socio-technical system to endure [4]. It is a systemic concept, as sustainability dimensions (which are social, individual, environmental, economic, and technical) are both interwoven with functions and constraints of any given socio-technical system, and mutually interdependent, and contextualized by the ethical and legal norms and social practices [50]. Despite the emergence of a number of initiatives, little guidance exists on which subjects and materials to cover in such an endeavor, on the interdisciplinary challenges of mixed groups confronted with sustainability theory and systems design practice, on teaching practices and experiences in this space, and a lack of reusable learning objects for the wider software engineering community to utilize.

This paper describes a blueprint for such an interdisciplinary intensive summer school course on Software Engineering for Sustainability (SE4S) targeted for an audience of mixed (both SE and non SE) students. We provide a blueprint for this course, in the hope that it can help the community develop a shared approach and methods to teaching SE4S. We also outline our course evaluation approach, which combines a pre- and post- course survey with an independent review of artifacts resulting from the course and participant reflections, which enables a qualitative assessment of the course and its outcomes. After a discussion of the background in Section 2, this article will present the design of the course and the evaluation study in Section 3; summarize the one-week summer school course that presents the case we study in Section 4; and discuss the implications of ourfi ndings in Section 5. Section 6 summarizes ourfi ndings and discusses further implications.

2 BACKGROUND

A number of commentators have considered the issue of how to integrate sustainability into the computing curriculum in higher education [5, 43]. Currently, integration of sustainability happens sporadically through a number of avenues such as: (i) Developing new courses, which cover selected sustainability and green computing topics [1, 20]; (ii) Designing and developing independent green computing learning modules and projects, which can be plugged into the existing computing courses [5, 28, 43, 56]; (iii) An integrative and transformative approach where computing courses are completely re-designed with sustainability at their heart [35, 44]; (iv) A topic-centered approach [1]. For example, Sherman [48] suggested a number of steps to achieve integration of sustainability concepts into any curriculum: (i) identify some big ideas within the discipline; (ii) identify a link between one or more of these ideas and the elements of sustainability; (iii) design a class component that integrates the discipline with sustainability. However, it is also suggested that regardless of the pedagogic approach adopted for teaching sustainability, an epistemic transformative and learning response that is able to facilitate a *transformative learning experience* is required to ensure that students are fully immersed in the topic of sustainability [9, 57].

Although limited in their generalizability, the results of feedback from the course evaluation suggest that students responded positively to their courses. However, despite the emergence of a number of pedagogical approaches their reproducibility to the wider software engineering community is both limited and in most instances take a narrow view of the concept of sustainability.

It is also suggested that the type of course that students take significantly impacts the way in which students conceptualize this term; the number of courses taken has no statistically significant impact [15]. This suggests that mere exposure to a particular theme in a class, rather than continued exposure to courses related to sustainability, is more important in shaping students' perceptions. In addition, Heeren et. al., [21] demonstrated that while knowledge had a significant bivariate correlation with behaviour, their results revealed that as students are educated about sustainability, fostering behaviour change will require education not only about how actions affect sustainability but also about social norms, attitudes towards sustainable behaviours and the level of self-efficacy in doing those behaviours.

In addition, a number of studies [5, 10, 13] have also identified a range of barriers and challenges to the integration of sustainability into the computing education curriculum including: a fundamental lack of interest; staff training; a lack of tradition; and a lack of priority; colleagues' scepticism; students' expectations of the course; an absence of policy; syllabus constraints; lack of leadership; an unfavourable view of the role of education for sustainability; the siloing within faculties of education; and a lack of a framework for sustainability education [18, 36].

This raises the question of *what concepts and topics we should teach for software engineering for sustainability within the computer science and software engineering curriculum?* Particularly as we observe that: (i) software development is often undertaken by non-specialists (i.e., other developers beyond trained software engineers); (ii) a software system is normally engineered for, and must conform to the requirements of "a client". In the following sections, we describe an approach that lays the foundation for the development of a global educational framework for teaching software engineering for sustainability.

3 GOALS AND STUDY DESIGN

3.1 Research objectives

The overall goal of this initiative is to establish a systematic approach for teaching software engineering for sustainability. The discussion above suggests that this intersection of areas requires an interdisciplinary perspective and an attention to reflective practice in design and in teaching. The following overall questions arise in the development of such an approach.

(1) **Content**. Which subject areas and modules constitute an effective baseline for an SE4S course curriculum?

[1]The notion of sustainability has been discussed extensively in a number of publications [3, 4, 8, 54], and readers are directed to these for an in depth treatment of this topic.

(2) **Structure**. How can they be scaffolded and applied in a project-based course?
(3) **Process**. How can we establish a reflective teaching practice among the diverse community of educators in SE invested in teaching SE for sustainability?

To address these questions, we developed a week-long summer school course and describe its content and structure below; and we evaluated and reflected on the teaching approach and the challenges encountered through multiple perspectives.

As a keyfi rst step, the overall objective of this study [45] is thus to understand, through this particularfi rst case, how to teach a group of students with mixed backgrounds such that they get an understanding of software engineering for sustainability and can apply this knowledge to a (local) project. The research questions addressed within this article thus are:

RQ1 Which subject areas and modules constitute an effective initial baseline for an SE4S course **curriculum**?[2]
RQ2 How can a project-based course be effectively focused on SE4S to synthesize diverse **backgrounds**?
RQ3 How difficult is it for students with varied backgrounds and knowledge of sustainability and software engineering to establish a shared working knowledge of this **intersection** of subjects?
RQ4 What types of **challenges** arise and how can they be addressed?

3.2 Study Data collection

The unit of analysis in this study is the design and teaching of an SE4S summer school course, which includes multiple embedded units of analysis and their relations. We structure these as follows.

The **course design** itself covers the *team of educators*; the *syllabus* produced and its content; the *weekly plan*; as well as lecture materials. As the content of the module represents the views of the educators team, the team is characterised by its subject-specific background and views on SE for sustainability.

The **participants** of the course are characterized in terms of their educational *background* and prior *experience*; the *learning outcomes* as evaluated by themselves and the educators; and the *reflections* they provided on the learning experience and change of perceptions.

The **practice** of teaching the course is characterized through observed *events and difficulties*, *challenges* and *strategies of adaptation* that were used to address identified challenges during the course.

3.3 Data Collection Methods

We triangulated the analysis of each unit of analysis using multiple sources of evidence, as illustrated in Fig. 1.

We used a pre- and post-survey and a report for the students' self-assessment and learning perceptions, and for the external assessment in form of artifact analysis we used a criteria catalogue.

Pre-Survey. A short pre-survey [41] evaluates whether our (informal and internal) hypothesis about the characteristics of the student population would hold, and to be able to better tailor the

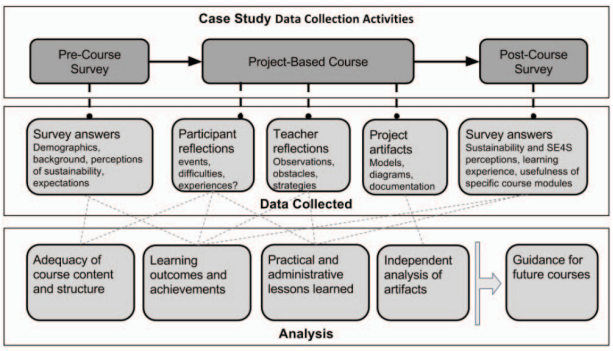

Figure 1: Data collection and analysis during the SE4S course

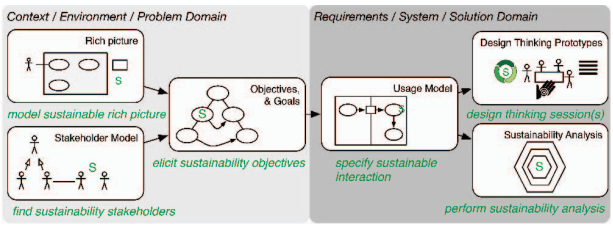

Figure 2: RE4S artefact model used in the course

course to the student population that decided to register for the course in that instance. The survey contained questions on their familiarity and experience with the general software engineering process, requirements elicitation and modeling, UML diagrams, SysML, IFML (Interaction Flow Modelling Language), Attribute-Driven Design (ADD), user interface development, rapid prototyping, design thinking, systems thinking, and computational thinking.

Post-Survey. We used a survey [41] that consisted of several sections, one part dedicated to the SE4S course, one part on ethics perceptions, and one part on value ratings.[3] The part specific for the SE4S course was composed by a background section, the motivation for the course selection, a section on their notions of sustainability, software and software engineering, their perceptions of the course content, and reflections on the course.

Artifact Analysis. Over the course of the week, the objective was to develop a specification according to a small requirements artifact model as well as some prototypes or mock-ups. An overview of the artifact model to be produced is given in Fig. 2. The artifacts are a rich picture, a stakeholder model, a goal model, a use case overview model, design thinking prototypes, and a sustainability analysis diagram. For the artifact analysis, we used a list of jointly elaborated quality criteria to structure their analysis (see [41]). For each artifact, there is a number of questions and criteria to be considered by the evaluators.

Analysis and validation procedures. Two external experts performed the artifact analysis. They were not part of the team that designed and carried out the study and who were not involved

[2]While we are highly interested in the entire SE process, RE has the biggest impact on sustainability [3], and the limited time for the course required us to focus on RE, some design activities, and the hackathon.

[3]This paper only analyzes the parts of the survey dedicated to the SE4S course due to space limitations. The additional parts on the wider ethics and value perceptions are forthcoming in a longer journal article.

with the course itself. They received the artifacts after the course had ended and used a list of jointly elaborated quality criteria to structure their analysis.

4 A COURSE ON SE4S

The week-long course is designed as part of a summer school held at the Lappeenranta University of Technology (LUT) in Finland. LUT has been arranging these international summer schools since 2012 and the event has attracted 150-200 students on approximately 15 different courses per year. One third to half of the students are international exchange students and the rest local students from various Finnish universities. For the year 2017, we proposed a course on Software Engineering for Sustainability that included five faculty, four lecturers and the local host. The course featured roughly seven hours of face-to-face time with the students per day, distributed in four sessions of 90-120 minutes each.[4] This sums up to a total of 35 hours of face-to-face time during that week, plus about 10 hours that the students invested in their time outside of class to prepare thefi nal reports and to do their reflections.

In its inception this course was driven by the convictions and background for the educator team. The team comprises three software engineering academics and an information systems researcher all with a key research interest in requirements engineering. The team holds a strong conviction that the key to engendering sustainability through software is in treating sustainability as an integral concern in software requirements engineering [3]. It is because the impact that a software system will have on its (natural, social, economic, and technical) environment is primarily determined by how the software engineers set out in the software requirements. Essentially, the requirements prescribe which system to build, whose interests to support through the system (i.e., who are the relevant stakeholders) and how will a success of this system be evaluated [3]. To briefly illustrate this: consider a weather update app. The resultant software and it socio-economic impact will drastically differ depending on such requirements as will it have to be informative for illiterate users (e.g., farmers orfishermen in developing countries), be multilingual and voice operated, or run on old hardware/software platforms? Thus, the syllabus of the course was strongly weighted towards tools/techniques that enable engineering sustainability into software system requirements.

Furthermore, the team is driven by conviction that all software design professionals need to engage with the sustainability design [4]. Thus, the initial syllabus design was aimed at a diverse population of software professionals: from HCI practitioners to those working on mathematics foundations of computer science. Yet, it held a silent assumption of some computer science background. We were expecting a mix of undergraduate (less) and graduate (more) students from different years with a focus on computer science and possibly a few from other disciplines. The software engineering knowledge that these students had would be limited but basic knowledge. We also expected little previous knowledge on sustainability, but knew that may not be the case for all students. The team, however, was aware that this assumption may not hold, and made an effort to identify the actual background of the intended audience, as discussed below.

We had prepared a brief list of readings ([4, 22, 39, 55]) to go through, but we also knew from past experience with summer schools that there is a limited likelihood of students following through on such a reading list before the start of a course.

The learning outcomes for the course are detailed in Tab. 1 along with how they are assessed.

Table 1: Learning outcomes

Goal / Objective	*Assessment / Measurement*
Sustainability concepts and principles: Develop an understanding of the concept of sustainability and its different dimensions and orders of effect and an ability to transfer these concepts to other application domains.	Students will demonstrate their mastery of sustainability concepts in demonstrating the transfer to a different application domain in their team project documentation.
Requirements Engineering: Students develop an understanding of the basics of requirements engineering, they understand and are able to apply stakeholder modeling, goal modeling, process modeling, use case modeling, and SysML.	Students demonstrate their mastery of requirements engineering by developing a consistent specification that includes stakeholders, goals, process model, use cases, and SysML diagrams.
Systems thinking: An understanding of the mindset of and the general principles of systems thinking, including holistic viewpoints and iterative development. Understand and be able to reason about long-term effects that a system under development may have on the environment and on society.	Students demonstrate their knowledge in taking a bigger picture perspective and holistic viewpoint in their rich picture. Demonstrate the reasoning in providing an analysis to that regard for the system under development and pointing out risks that may or may likely occur in the future given certain conditions.
Design thinking: Understand and be able to apply design thinking on (complex) problems, which requires alternating between narrowing down and opening up the perspective.	Demonstrate the application of design thinking on a local problem to demonstrate understanding and transfer of the concepts of iterative development in innovation in their project.

The subject areas and modules are detailed in Tab. 2. For each, we provide content, example key references, and rationale for their inclusion in the course. For more details, please refer to [41].

Table 2: Subject areas and modules

Module	*Content*	*Key refs.*	*Rationale for inclusion*
Sustainability foundations	Dimensions of sustainability, orders of effect, application domains, Sustainable Development Goals	[4, 19, 22]	Provides a common basis for scoping sustainability within this course and puts the concept into a larger perspective.
Principles of sustainability design	Principles of substitution, decoupling, and dematerialization; Software Engineering for Sustainability examples	[7, 22, 39]	Introduces the principles that can be used for thinking of system ideas for the projects to be developed during the course.
Rich pictures	Rich picture method for scoping and high-level domain modeling	[34]	Rich pictures are a simple, non-technical method to illustrate the vision for a system or scoping of a problem in its surrounding application domain and operational context.
Stakeholder and goal models	Introduction to concepts, roles, reference models, analysis, and notations for both of the models.	[40, 42]	Forms the basis for eliciting requirements for a chosen project idea. The stakeholder model makes sure all relevant roles are included, the goal model provides a basis for consensus, conflict identification and trade-offs.
Process modeling	Introduction to concepts of and notation for business process modeling	[27, 37]	Provides the transition from high-level goals to the operationalization of these objectives in executable processes.
Software Engineering	Overview of the general Software Engineering process phases and introduction to use cases	[25]	Complements the technology-agnostic processes with the technology-aware perspective by describing the interaction between user and system.
SysML	Introduction to SysML for analyzing and designing complex systems	[16]	The Systems Modeling Language is a widely used general purpose language for modeling and verifying software systems.
Design Thinking	Hands-on crash course tutorial in design thinking	[49]	The d.school's highly interactive workshop on design thinking facilitates transition into rapid prototyping.
Sustainability analysis	Introduction to the analysis and estimation of a system's impact according to the sustainability dimensions and order of effect	[3]	The overview of all impacts of a system from short-term to long-term enables software engineers to take a wider perspective and better judge the consequences of their choices during development.

4.1 Project Assignments

Students develop the following artefacts throughout the course: a rich picture, a stakeholder model, a goal model, a use cases model, a sustainability analysis diagram and physical prototypes. The assignments to develop the artifacts in Fig. 2 are given in class at the end

[4] 90 minutes per session are sufficient, 120 minutes just allow for a longer and more detailed feedback discussion.

of the lecture module that introduced a specific artifact [39]. The instructions are given orally as the students use the consecutive hours to develop that artifact in their team. Students also receive five sample reports from different domains from an earlier course. The instructors are present during the working sessions and available for guidance, questions, or preliminary feedback. In addition, there is a hackathon planned for the weekend, where students code prototypical implementations of their projects. As the focus is more on the conceptual understanding of how to integrate sustainability considerations into software, students are asked to focus on concepts rather than specific precise diagram notations. Students deliver their reports after thefi nal presentation, and therefore have the chance to incorporate the lecturers' feedback.

4.2 Learning Perception

At the beginning of the course we provided the students online space to write down their reflection and their learning perception in an online journal. At the end of each working session we provided the students 5 minutes time tofi ll in their experiences in the journal. In addition after each day we reminded them tofi ll out the journal to gather their experiences. While we still think this is a good teaching instrument, due to the fact that most students participated without a laptop, this diary was not followed through.

5 STUDY RESULTS

5.1 Participants

Which students did end up attending? Our assumption about students' backgrounds turned out to be off. While we had expected that most students would have a computer science background, a small pre-survey, which was answered by half of the students who had signed up, revealed that we could expect hardly any software engineering knowledge, and little knowledge about sustainability as well. We had 13 students from various disciplines, namely computer science, business and marketing, industrial engineering and environmental engineering, and they were mostly Bachelor students. On the second day of the course, it turned out the students had not received any information about the course that we had tried to get to them, so our preparation reading was rendered irrelevant as none of the students was enrolled on the learning platform that was set up about the course. The learning platform included, in addition to the preparation reading, a detailed course plan and the objectives of the course. Thus, the students did not know what to expect from the course.

5.2 Teaching the course

The planned schedule of modules of the course and the how we modified it according to circumstances are depicted in Fig. 3.

Day 1. *Introduction to sustainability, and Software Engineering for Sustainable Systems*: The key expected outcomes from this day were the common ground on the perception of sustainability, the role of software systems in addressing sustainability challenges, and a choice of a project that a group would work with through the rest of the course, including a rich picture of it. The basic introduction to the notions of sustainability is a necessity in such a course, as part of the class had no previous systematic knowledge on this subject. To keep the whole class engaged (including those with previous knowledge of sustainability), the sessions were intersected with a number of interactive exercises, such as hands folding to get away from established thinking patterns [33]. On the whole, the sessions were well received and the class was fully engaged in all activities. However, for the system vision activity, it quickly became apparent that there was some genuine resistance to the broader system view idea by some engineering students. These students perceived the visioning and broader system view activities as irrelevant. One of the students repeated several times "We just need to implement this system". Such much-too early and narrow implementation-focused attitude is one of the core symptoms of inadequacy of the current software developers' education, whereby no or little consideration is given to the longer-term impact that the technical systems have upon their situated environments. We note that tackling such practices and attitudes will necessarily be one of the hurdles in any SE for Sustainability curricula.

Day 2. *Stakeholder and Goal Models*: The key expected outputs that day were stakeholder and goal models. Both types of models were presented to the class and illustrated with generic and sustainability specific examples. The notion of stakeholder models was well understood and realised by the groups for their own systems. However, the groups had difficulties with the goal modelling activity. The groups: (1) did not understand how the high-level goals (e.g., support environmental sustainability) relate to specific software systems; (2) struggled with identifying appropriate goals; (3) found it difficult to represent the complex interrelationships and interdependencies due to quick growth of the model elements; (4) did not quite know when to terminate the modelling activity. Clearly, many of these issues are well known with respect to the goal modelling technique, and some are exacerbated by the complex nature of sustainability concerns, e.g., to support goal identification and decomposition/termination goal, generic models and catalogues can be used [42]. We returned to the exercise of the goal decomposition with a step-by-step illustration of one of teams' project, demonstrating how generic sustainability goals can be operationalised into specific tasks and functions of a software system. We observe that this specific example-based exercise was productive in conveying the key notions of this technique to the students. Yet, this may lead to blind "copy-paste" approach in problem analysis, rather than specific and detailed consideration (see Sec. 6).

Day 3. *Systems and System Boundary, Introduction to Software Engineering, Use Cases*: The key expected outcomes of Day 3 initially were design models. However, we chose to teach the use case models only because (1) the class continued on the goal modelling task for the 1st part of the day, (2) we observed a lack of modelling experience on students, and (3) a considerable amount of new notations had already introduced to them. Due to time constraints, we chose not to undertake the textual description of use cases. The flexibility in including or excluding specific techniques into the course is quite important to help adapting the pace to the learning capacity of a given class at a given time. Given the mix background of the students, the general lack of familiarity with SE techniques, and the short duration of the course, the introduction of further models and notations would likely have overwhelmed the students.

Day 4. *Design Thinking and Sustainability Analysis*: The key expected outputs from Day 4 were the design thinking artefacts helping to review the project ideas, as well as the sustainability

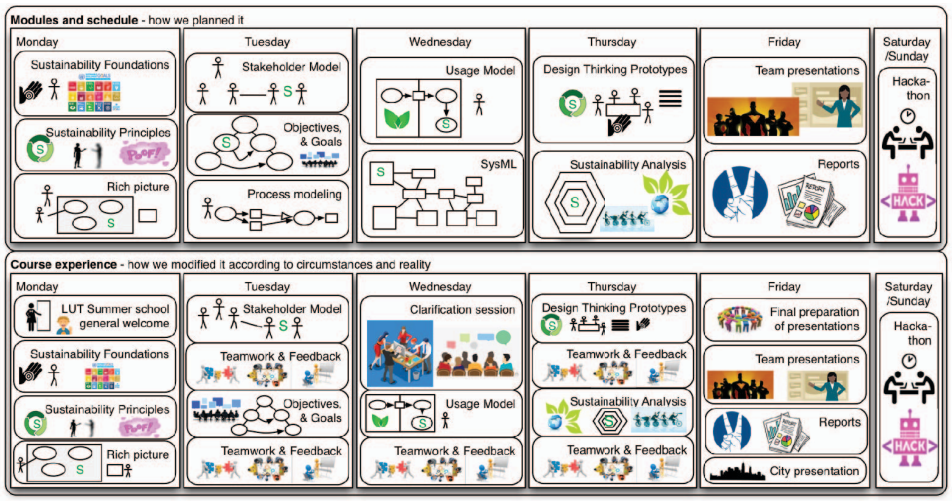

Figure 3: SE4S course planned schedule and actual experience

analysis diagrams of their projects decisions. The students were paired with project members from other groups for the design thinking exercise. This resulted in a number of radical ideas and reviews of the projects. However, here too (as observed in Day 1) a few students were unwilling to listen to external input, as they had already made their mind up about their own "system implementation". All the students commented on the usefulness of the sustainability analysis via those diagrams, as they were able to observe direct, indirect, and systemic effects of their decisions upon the sustainability dimensions.

Day 5. *Group Presentations and Reports*: On Friday, the students gave theirfi nal presentation and gathered feedback regarding their artifacts as well as the learning outcomes. The last two sessions of the week were working sessions so that the students couldfinalize their reports. That way, they had the chance to incorporate the lecturers' feedback, which some addressed effectively.

For the projects, the students were prompted to think of a sustainability challenge in their direct environment in Lappeenranta and/or in campus life. They came up with the following sustainability challenges:

- Consumers should be informed about the **impact of their choices**, as often choices are made without appreciation of their sustainability impact;
- Lappeenranta is a northern town, requiring a lot of **energy** to be generated and used for heating and lighting; and
- Lappeenranta suffers from **youth migration** away to the larger cities, as there is not enough to do for workforce, which can cause economic and social sustainability issues.

We had four teams in total. Students organized themselves into groups of 3 to 4 people with mixed backgrounds. The project ideas and backgrounds were as follows:

3-D printer: The on-demand fabrication of products supports personal choice and can facilitate product reuse and recycling. This project proposes a system that enables customers, CAD designers and 3-D manufacturers to exchange blueprint models of products and have them printed in 3-D. The system gives the user the opportunity to recycle products (or parts) and to request the use of recycled material in the manufacturing of their products (see Fig. 4). Backgrounds: mechanical engineering (2 students) and computer science.

Trading and sharing energy platform: Household heating takes up a lot of energy and results in high CO2 emissions. This project proposes a trading and sharing platform of energy for local communities, in which households are both consumers and producers of energy. The system encourages the use of local renewable energy, facilitates the energy sharing among neighbors, and reduces the environmental impact of energy production and consumption. Backgrounds: electrical engineering (with some previous knowledge of software engineering), sustainable production, and innovation for sustainable development.

Sustainable heating system: This project suggests a heating system that, in addition to the usual temperature and timing setup, provides users with information about the environmental impact and energy cost of their heating usage patterns. The aim is to raise awareness and change consumption patterns (see Fig. 5). Backgrounds: mechanical engineering, industrial engineering & management, and sustainability.

Food educator app:This project aims to raise awareness about sustainability by creating an app that informs its users about the "sustainability meat index" of different types of meat; an index that measures how much environmental, social and economic impacts has produced a specific type of meat during its lifecycle. Backgrounds: sustainability, computer science, management, and industrial engineering & sustainable management.

5.3 Analysis of the course

We evaluate our course to understand whether mixed-background students can develop a shared understanding of the intersection between sustainability and software engineering, and to apply this

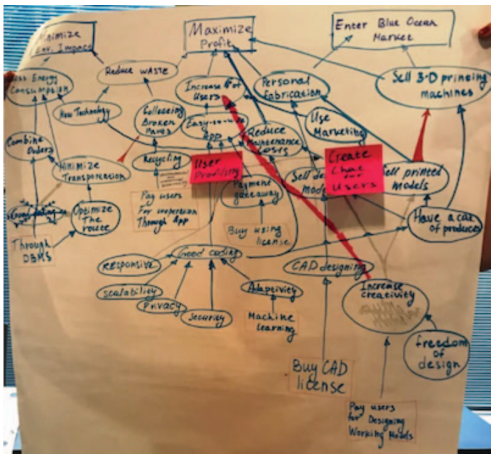

Figure 4: Goal Model of the 3-D printer team

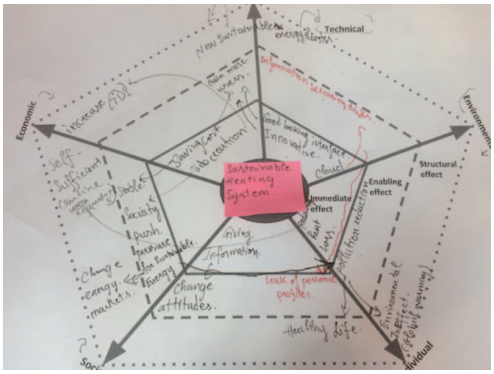

Figure 5: Sustainability Analysis Diagram of the Sustainable Heating System team

understanding to design a software that impacts the application environment in a sustainable way.

The input for our analysis were the reports delivered by the students at the end of the course as well as the post-survey. As described in Sec. 4.1, the reports included the artifacts produced during the course. For example, Figure 4 shows the goal model of the 3-D printer group and Figure 5 illustrates the sustainability analysis diagram for the Trading and sharing energy platform.

5.3.1 Artifact analysis. In order to analyse the artifacts themselves, two evaluators (also authors of this paper) who were not involved in the summer school independently analysed the reports following the criteria shown in the appendedfile *Future Online Supplementary Material*. Evaluators analysed all artifacts, with the exception of the 3-D prototypes. The latter was part of a design thinking exercise and were only meant to serve as internal feedback mechanism to help iterate solution ideas [49]. Next, they compared their thoughts on each project for the different criteria, consolidating their impressions and going back to the original reports when conflicts occurred. The artifact analysis indicates that the students have understood the concept of sustainability and orders of effects, and most importantly, that software-intensive systems can have an effect on sustainability. The artifact analysis revealed that despite the very short training period and their mixed background, all groups were able to create diagrams that showed they grasped the main ideas of the models. However, if compared with diagrams normally created by software engineering students during a regular semester course, the diagrams had more conceptual and notational mistakes. In terms of group composition, there was no notable differences in the general quality of the reports produced. One possible exception was the 3-D printer group, whose sustainability challenge and how it would be achieved was not so clear. Notably, this group did not have a member with sustainability background. In general, groups favored free-form sketches rather than adopting predefined notations. This is not surprising, as lectures were more concerned with the concepts than notation. Additionally, students mixed the descriptions of the artifacts with their own reflections on the artifacts and the learning process; and the former were rather poor. All this made models more difficult to understand.

5.3.2 Post-Survey results. The survey was answered by all participants. One participant apparently lost interest halfway through and so we couldn't use part of that participant's answers. However, as the number of respondents is too small for any statistical significance, we are limited to qualitative analysis and report on our findings therein. Students chose the course for learning either about coding, sustainability, software engineering, or software engineering for sustainability. They expected to learn how sustainability can be tied into software engineering as well as to get an overview of thefi eld. Their understanding of sustainability before starting the course varied considerably. Some students simply stated they had a background in it, some based their understanding on the three pillars or on the idea to preserve for future generations, and some narrowed it down to reducing resource consumption. Of the students that didn't have a background in sustainability, most reported a widening of their understanding through the course. Before the course, about two thirds of the students thought software could help with sustainability, and after the course all of them did. The element of the course that changed their perception varied quite a bit, but mainly from rich picture to goals to design thinking prototypes. Moreover, their perception of how software could help with sustainability widened exceptionally, as one student put it: "Now I feel that everything is interrelated and every task can have a vision of sustainability". Software engineering, before the course, was perceived as being mainly coding by half of the respondents but after the course received a more encompassing perception. While, before the course, the majority thought software engineering could not help with sustainability, afterwards all of them thought it could. Therefore, their understanding of what software engineering for sustainability means broadened considerably over the course, including process aspects, the use of models, and wider scoping including the society.

What helped in linking the notions of software engineering and sustainability were the rich picture, goals, sustainability analysis, and the systems thinking activities. Overall, these answers show the different expectations and backgrounds of the students. However, they also show that the students with a lack of knowledge with

regard to sustainability think they gathered knowledge with regard to sustainability and that the ones with less knowledge in Software Engineering gathered knowledge in Software Engineering.

The key sustainability learning experiences were around the people aspect of sustainability (stakeholders, society), the immediacy of sustainability ("not abstract", "sustainability is real", overshoot day, "everyone can take part"), and that software can help with innovating. Also, as most students did not know how Sustainability and Software Engineering relate — this was the key learning outcome, to see that and how they actually relate. The key SE learning experiences were that SE is more than coding, the process (flow) and some guidelines ("user must be studied", "be patient"), and the techniques we taught (rich picture, goals, use cases). For their plans of what to do next with the acquired knowledge, the majority wanted to apply their new skills, some mentioned integrating it in other areas, and study more. They would like to continue their learning in this area, get to know the next steps, and transfer their knowledge to other areas. About the course, they disliked that there was no big picture given at the beginning of the course and that lead to some confusion. They did like the teamwork, activities (interaction) and tools.

6 DISCUSSION

6.1 Benefits for students

The students in general benefitted from multidisciplinary teamwork, hands-on practice sessions with feedback rounds and iterations, and developing a project that they can build upon for their portfolio. They learned to appreciate the "wider" sustainability - to look beyond their own discipline (e.g., software or environment only), observing that various notions, which are normally not considered relevant for software itself, are linked with sustainability concerns. They experienced this to show in multi-level and multi-artefact influences. They received an introduction to and practice with tools that set out the analytical framework for reasoning about sustainability-related impact of decisions, choices, and actions. Note that goals, stakeholders, rich picture, spider web can be applied not only for software requirements analysis, but also for any other requirements analysis, as argued in [6]. They can now take an abstract notion of sustainability and refine it to specific actionable steps/objectives.

6.2 Lessons learned

Content adequateness. Provide adequate level of content in a mixed-background group. Given the dramatic variety of backgrounds, it was difficult to teach the class at the level that was right for all students. Some students, perceived that it was too much content for a one week course, making it challenging for them to properly internalise the presented materials. For instance those who did not have previous knowledge of software engineering, found the number of analysis techniques presented somewhat difficult to handle. Yet, those with previous SE degrees complained that the SE content was "basic". From this we note that the following runs of such mixed background classes would benefit from a period of small-group topic-specific teaching, where the topics not familiar to a given group are discussed, aiming to bring the overall class level to some common reasonable grounding level. From that point onwards the class can be taught as one again. But the initial differentiated group teaching will save repeated introduction of basics to more experienced students for each group category.

Templates. The teaching team had discussed whether to prepare templates or the common notions of sustainability modeled in various artefacts. While some samples were presented to the students before their own hands-on modelling activities (e.g, sample stakeholder and goal models), it was agreed not to provide any template solutions, in order not to limit students' creativity and allow for clear emergence of domain specific characteristics. For example, teams from the food domain and the energy domain need very different rich pictures and hardly have overlapping elements. Also, while in general we are in favor of templates, there might even be a limitation to their learning if they received templates of such a degree of detail and preparedness that the assignment almost turns into afi ll-the-blank exercise.

Scoping sustainability. What is *enough* to look at? People who were familiar with systems thinking were able to handle that better. We even drew a picture of how to go through development steps and artefacts iteratively and where you have a wider scope (rich picture) and then narrow the perspective (use cases) and then widen it again (sustainability analysis). Also, students resisted to open up their scope and changing perspectives, especially the ones having an engineering background. Moreover, once they came to the point of opening up and thinking about the system from a holistic perspective, taking sustainability effect overtime into account, it was quite difficult for the students to focus on the system to develop again. Overall, this concept of alternating between narrowing down and opening up the perspective when designing sustainable software systems was difficult for them.

Administrative issues. There was a lack of direct contact during preparation and setup for the summer school between students and teaching team: Do not rely on hosting institutions getting all information to external participants without double-checking they actually receive it. While the hosting institution was timely in issuing local email addresses to students and providing them with access to the local learning platform that contained all course materials, most students did not have access to that local email address or were not aware of it until the second day, after the course had started. Consequently, they felt left in the dark about the content and preparation for the course and did not get a chance to go through the recommended readings beforehand. This could have been circumvented easily by us sending a direct email to the registered students under the email address they registered with and provide them access to, e.g., a sharedfi le folder.

Signposting. Clear annunciation to the prospective students of the aims, objectives, and learning outcomes as well as of the module structure well ahead of the module run. Students had chosen this module expecting different learning outcomes (e.g., one was intending to learn "coding", some wanted to learn about "sustainability", others were only interested in "software engineering", and yet others wanted to learn about integrating both software engineering and sustainability). Consequently, the expectations of many students were not fully met, leading to disappointment. This could also be avoided if the participation had been chosen for the "right" objectives. Course organizers also had offassumptions

about the background of the course participants and thus different learning objectives were not expected.

6.3 Evaluation of validity

For the evaluation of the validity of the study, we report on construct, internal, and external validity as well as reliability [45].

Construct validity focuses on whether the theoretical constructs are interpreted and measured correctly [11]. To minimize the threat, we based our evaluation on the established concepts expressed in the requirements engineering artifacts and on a basic survey that asks about the students' state of knowledge before and after the course and their learning experience.

Internal validity focuses on the study design, and particularly whether the results really do follow from the data [11]. The evaluation of the artifacts was performed by two researchers who were not lecturers or participants of the course to increase the neutrality of the analysis. They conducted the analysis individually and conferred about their results in discussion until they reached joint conclusions. The evaluation of the survey results was performed by one researchers qualitatively coding the answers and a second researcher checking the coding and adding codes where found relevant. Concerns with regard to a couple of answers and how to interpret them were jointly discussed and resolution agreed upon. However, the most substantial student feedback data on the pre- and post-course knowledge were collected through questionnaire at the end of the course. Thus, it is possible that the responses on the pre-course knowledge questions (e.g., what did you think sustainability was before doing this course ...) have been biased either due to the students currently different level of knowledge, and/or due to their emotional state (e.g., if a student is disappointed that the module did not provide any coding opportunities, as he expected, he may say he already knew the other presented material). Such a bias could have been, to some degree, mitigated through reference to the submitted artefacts at group level. Yet, since in the present run of the course, each artefact had been iteratively improved through several feedback cycles, the artefacts cannot be objectively contracted to the validate the pre-module content related feedback claims.

External validity focuses on whether claims for the generality of the results are justified [11]. This is a qualitative study, reporting on our, so far, single experience of teaching a particular mixed group course on software engineering for sustainability. We report on our qualitative insights from this experience. In the following years, we are planning to replicate this study and establish a series of related evaluations to strengthen ourfindings.

Reliability focuses on whether the study yields the same results if other researchers replicate it [11]. We are aware that we will not be able to exactly replicate the results of the study because the setting of such a course will always be slightly different. As the course materials are available and reusable, and the set-up and implementation of the course as well as the analysis of the results are described in detail, the study is transferable and we provide a basis for a family of studies. Such a family of studies would provide reliable results.

7 CONCLUSIONS AND OUTLOOK

How can we integrate sustainability in a computer science / software engineering curriculum? This paper presented a course design and qualitative evaluation of a summer school course on software engineering for sustainability. The course was taught by four instructors to a group of students from mixed backgrounds at the Lappeenranta University of Technology in Finland. The paper summarized the content, structure and process of the course itself. Our qualitative study explores four research questions, which are answered in **summary** below.

(RQ1) Which subject areas and modules constitute an effective initial baseline for an SE4S course curriculum? The course blueprint articulated a candidate set of modules. The evaluation of the course suggests that together with the project-based course design, the modules presented a reasonable baseline to develop further.

(RQ2) How can a project-based course be effectively focused on SE4S to synthesize diverse backgrounds? Teaching the course in practice was met with challenges, but reflections and independent evaluations suggest that the learning experience proved valuable and that the project theme and support by the educators was successfully enabling the overall achievement of learning outcomes.

(RQ3) How difficult is it for students with varied backgrounds and knowledge of sustainability and software engineering to establish a shared working knowledge of this intersection of subjects? It is difficult, even with the excellent instructor to student ratio of a summer school course. However, the emphasis on reflective practice over notational accuracy showed promising initial outcomes as stepping stones for a continued engagement with SE4S; a strong outcome for a week-long course.

(RQ4) What types of challenges arise and how can they be addressed? Challenges included administrative hurdles, expectations on prior subject area expertise, and time. The experience report included in this paper should prove valuable for future attempts at similar courses.

Benefits noted by the students include: **(B1)** A wider perception of sustainability and discipline-independent understanding; **(B2)** A generic analysis framework using requirements engineering and design thinking; **(B3)** Multidisciplinary teamwork and project-based, hands-on practice sessions.

Lessons learned in teaching SE4S to mixed background student groups are to: **(L1)** Create more targeted interaction with the students before the summer school to better guide preparation; **(L2)** Have more consistent signposting towards the overall objectives for the course during the project week; **(L3)** Adapt the content for background-specific subgroups to bridge the gaps between the knowledge levels; **(L4)** Element overviews plus examples plus walking through their own drafts in a round of feedback works really well for understanding the application of these methods; **(L5)** Scoping sustainability requires a general understanding of systems thinking concepts.

The course provides a blueprint for future courses, and the study presents a baseline and design that supports future replications to get a deeper understanding of how to best teach a group of students with mixed backgrounds. Many opportunities to deepen and expand on thisfi rst experiment exist. We intend to replicate the study in various universities in a family of future studies.

ACKNOWLEDGMENTS

We thank the EU PERCCOM program for their support. The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 712949 (TECNIOspring PLUS), from the Agency for Business Competitiveness of the Government of Catalonia, NSERC RGPIN-2016-06640, and the UK EPSRC Refactoring Energy Systems fellowship (EP/R007373/1).

REFERENCES

[1] Ken Abernethy and Kevin Treu. 2014. Integrating Sustainability Across the Computer Science Curriculum. *J. Comput. Sci. Coll.* 30, 2 (Dec. 2014), 220–228.
[2] Robert E. Beck and Daniel T. Joyce. 2013. Sustainability Improves Student Learning (SISL) in Computing (Abstract Only). In *Proc. of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, 730–730.
[3] Christoph Becker, Stefanie Betz, Ruzanna Chitchyan, Leticia Duboc, Steve M Easterbrook, Birgit Penzenstadler, Norbet Seyff, and Colin C Venters. 2016. Requirements: The key to sustainability. *IEEE Software* 33, 1 (2016), 56–65.
[4] Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C Venters. 2015. Sustainability design and software: The karlskrona manifesto. In *ICSE-SEIS*, Vol. 2. IEEE, 467–476.
[5] Yu Cai. 2010. Integrating Sustainability into Undergraduate Computing Education. In *Proc. of the 41st ACM Technical Symp. on Computer Science Education*. 524–528.
[6] D. Callele, K. Wnuk, and B. Penzenstadler. 2017. New Frontiers for Requirements Engineering. In *Intl. Requirements Engineering Conf.* IEEE, 184–193.
[7] Ruzanna Chitchyan et al. 2015. Evidencing sustainability design through examples. In *4th Intl. Workshop RE4SuSy*.
[8] Ruzanna Chitchyan, Christoph Becker, Stefanie Betz, Leticia Duboc, Birgit Penzenstadler, Norbert Seyff, and Colin C Venters. 2016. Sustainability design in requirements engineering: state of practice. In *ICSE-SEIS*. ACM, 533–542.
[9] A. Desai. 2015. Hands on project experience in a core class focused on sustainability. In *Proc. of the IEEE Frontiers in Education Conf.* 1–4.
[10] Lorna Down. 2006. Addressing the challenges of mainstreaming education for sustainable development in higher education. *Int. J. of Sustainability in Higher Education* 7, 4 (2006), 390–399.
[11] S. Easterbrook, J. Singer, M.A. Storey, and D. Damian. 2007. *Selecting Empirical Methods for Software Engineering Research*. Springer.
[12] Richard Emanuel and J.N. Adams. 2011. College students' perceptions of campus sustainability. *Int. J. of Sustainability in Higher Education* 12, 1 (2011), 79–92.
[13] Thomas Falkenberg and Gary Babiuk. 2014. The status of education for sustainability in initial teacher education programmes: a Canadian case study. *Int. J. of Sustainability in Higher Education* 15, 4 (2014), 418–430.
[14] D. H. Fisher, Z. Bian, and S. Chen. 2016. Incorporating Sustainability into Computing Education. *IEEE Intelligent Systems* 31, 5 (2016), 93–96.
[15] P. Brian Fisher and Erin McAdams. 2015. Gaps in sustainability education: The impact of higher education coursework on perceptions of sustainability. *Int. J. of Sustainability in Higher Education* 16, 4 (2015), 407–423.
[16] Sanford Friedenthal, Alan Moore, and Rick Steiner. 2008. OMG Systems Modeling Language (OMG SysML) Tutorial. In *INCOSE Int. Symposium*, Vol. 18. 1731–1862.
[17] M. L. Gibson et al. 2017. Mind the chasm: A UKfi sheye lens view of sustainable software engineering. In *6th Intl. Workshop on Requirements Engineering for Sustainable Systems*. 15–24.
[18] Michael Goldweber et al. 2012. A Framework for Enhancing the Social Good in Computing Education: A Values Approach. In *Reports on Innovation & Technology in Computer Science Education Working Groups*. 16–38.
[19] David Griggs et al. 2013. Policy: Sustainable development goals for people and planet. *Nature* 495, 7441 (2013), 305–307.
[20] Margaret Hamilton. 2015. Learning and Teaching Computing Sustainability. In *Proc. of the 2015 ACM Conf. on Innovation and Technology in Computer Science Education*. 338–338.
[21] A. Heeren et al. 2016. Is sustainability knowledge half the battle?: An examination of sustainability knowledge, attitudes, norms, and efficacy to understand sustainable behaviours. *Int. J. of Sustainability in Higher Edu.* 17, 5 (2016), 613–632.
[22] Lorenz M Hilty and Bernard Aebischer. 2015. Ict for sustainability: An emerging researchfi eld. In *ICT Innovations for Sustainability*. Springer, 3–36.
[23] Michelle Horhota et al. 2014. Identifying behavioral barriers to campus sustainability: A multi-method approach. *Int. J. of Sustainability in Higher Education* 15, 3 (2014), 343–358.
[24] David H. Kaplan. 2015. Transportation sustainability on a university campus. *Int. J. of Sustainability in Higher Education* 16, 2 (2015), 173–186.
[25] Gerald Kotonya and Ian Sommerville. 1998. *Requirements engineering: processes and techniques*. Wiley Publishing.
[26] A. Kurkovsky. 2006. Educational Aspects of Sustainable Development Analysis: Computational Models and Software. *J. Comput. Sci. Coll.* 21, 4 (April 2006), 24–31.
[27] Selim Larsch, Stefanie Betz, Leticia Duboc, Andréa Magalhães Magdaleno, and Camilla Bomfim. 2016. Integrating Sustainability Aspects in Business Process Management. In *Business Process Management Workshops*. 403–415.
[28] D. Lopez et al. 2014. A methodology to introduce sustainability into the final year project to foster sustainable engineering projects. In *IEEE Frontiers in Education Conf.* 1–7.
[29] Samuel Mann. 2016. Computing Education for Sustainability: What Gives Me Hope? *interactions* 23, 6 (Oct. 2016), 44–47.
[30] Samuel Mann, Lesley Smith, and Logan Muller. 2008. Computing Education for Sustainability. *SIGCSE Bull.* 40, 4 (Nov. 2008), 183–193.
[31] Andrew McGettrick et al. 2005. Grand Challenges in Computing: Education—A Summary. *Comput. J.* 48, 1 (2005), 42–48.
[32] D. Meadows, D. Meadows, and J. Randers. 1992. *Beyond the Limits: Confronting Global Collapse, Envisioning a Sustainable Future*. Chelsea Green Publishing Co.
[33] D. Meadows, D. Meadows, and J. Randers. 2004. *Limits to Growth: The 30-Year Update*. Chelsea Green Publishing Co.
[34] Andrew Monk and Steve Howard. 1998. Methods & tools: the rich picture: a tool for reasoning about work context. *interactions* 5, 2 (1998), 21–30.
[35] L. Morell et al. 2012. An engineering curriculum track for IT for sustainability. In *Proc. of the IEEE Frontiers in Education Conf.* 1–6.
[36] Simon O'Rafferty, Hannah Curtis, and Frank O'Connor. 2014. Mainstreaming sustainability in design education – a capacity building framework. *Int. J. of Sustainability in Higher Education* 15, 2 (2014), 169–187.
[37] Martyn A Ould and MA Ould. 1995. *Business Processes: Modelling and analysis for re-engineering and improvement*. Vol. 598. Wiley Chichester.
[38] J. Pappas and E. Pappas. 2015. The Sustainable Personality: Values and Behaviors in Individual Sustainability. *Int. J. of Higher Education* 4, 1 (2015), 12–21.
[39] B. Penzenstadler. 2014. Infusing Green: Requirements Engineering for Green In and Through Software Systems.. In *Third Intl. Workshop RE4SuSy*.
[40] B. Penzenstadler et al. 2013. Who is the advocate? Stakeholders for sustainability. In *Proc. of the 2nd Intl. Workshop on Green and Sustainable Software*. IEEE, 70–77.
[41] Birgit Penzenstadler, Stefanie Betz, Colin C. Venters, Ruzanna Chitchyan, Jari Porras, Norbert Seyff, Leticia Duboc, and Christoph Becker. 2018. Supplementary Material of "Everything is Interrelated: Teaching Software Engineering for Sustainability". http://arxiv.org/abs/1802.02517. (2018).
[42] B. Penzenstadler and H. Femmer. 2013. A generic model for sustainability with process-and product-specific instances. In *Workshop on Green in/by software engineering*. ACM, 3–8.
[43] B. Penzenstadler and A. Fleischmann. 2011. Teach sustainability in software engineering?. In *24th CSEET*. 454–458.
[44] J. Porras et al. 2016. PERCCOM: A Master Program in Pervasive Computing and COMmunications for Sustainable Development. In *Proc. of the IEEE 29th Int. CSEET*. 204–212.
[45] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164.
[46] Kaisu Sammalisto et al. 2016. Learning about Sustainability: What Influences Students' Self-Perceived Sustainability Actions after Undergraduate Education? *Sustainability* 8, 6 (2016).
[47] Ann E. Savageau. 2013. Let's get personal: making sustainability tangible to students. *Intl. J. of Sust. in Higher Education* 14, 1 (2013), 15–24.
[48] Daniel J. Sherman. 2008. Sustainability: What's the Big Idea? *Sustainability: The J. of Record* 1, 3 (2008), 188–195.
[49] Stanford d.school. 2017. A Virtual Crash Course in Design Thinking. (2017). https://dschool.stanford.edu/resources-collections/a-virtual-crash-course-in-design-thinking last accessed Oct 21 2017.
[50] H. Sverdrup and M. Svensson. 2005. *Defining the Concept of Sustainability - a Matter of Systems Thinking and Applied Systems Analysis*. Springer, 143–164.
[51] Siwaporn Tangwanichagapong et al. 2017. Greening of a campus through waste management initiatives: Experience from a higher education institution in Thailand. *Int. J. of Sustainability in Higher Education* 18, 2 (2017), 203–217.
[52] F.E. Trainer. 1997. The global sustainability crisis: The implications for community. *International J. of Social Economics* 24, 11 (1997), 1219–1240.
[53] Hans van Weenen. 2000. Towards a vision of a sustainable university. *Int. J. of Sust. in Higher Education* 1, 1 (2000), 20–34.
[54] Colin C Venters, Norbert Seyff, Christoph Becker, Stefanie Betz, Ruzanna Chitchyan, Leticia Duboc, Dan McIntyre, and Birgit Penzenstadler. 2017. Characterising sustainability requirements: A new species red herring or just an odd fish?. In *ICSE-SEIS*. IEEE, 3–12.
[55] Eric Williams. 2011. Environmental effects of information and communications technologies. *Nature* 479, 7373 (2011), 354–358.
[56] T. Worthington. 2012. A Green computing professional education course online: Designing and delivering a course in ICT sustainability using Internet and eBooks. In *Proc. of the 7th Intl. Conf. on Computer Science Education*. 263–266.
[57] C. Zhou. 2016. Developing creativity as a scientific literacy in software engineering education towards sustainability. In *12th Intl. Conf. on Natural Computation, Fuzzy Systems and Knowledge Discovery*. 2257–2261.