Sustainability Design and Software: The Karlskrona Manifesto

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

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

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

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

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

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

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

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

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

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

I. INTRODUCTION

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

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

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

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

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

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

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

II. SOFTWARE SYSTEMS AND SUSTAINABILITY

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

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

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

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

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

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

This approach is rejected as weak sustainability by those who argue that it’s an error to seek to balance the three concerns. According to this view, Figure 1(a) is misleading, as the economy is really only a subsystem of society, which
in turn is a subsystem of the environment (Figure 1(b)). To achieve strong sustainability, we have to acknowledge that there are fundamental biophysical limits that constrain the flows of natural resources on planet earth, and no arrangement of society can be considered sustainable unless it lives within these limits [31]. In this view, it is wrong to talk about sustainability in terms of a set of ‘dimensions’, as the concerns are strictly hierarchical.

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

While we believe these questions are important, we do not believe they offer a useful starting point for software practitioners and researchers struggling with the question of what sustainability means for them. A more pragmatic view is shown in Figure 1(c), where sustainability is depicted as a learning process by which we move towards integrated thinking. Software practitioners tend to treat techno-centric concerns (e.g., software qualities and the economic value they create) separately from socio-centric concerns (how software can make people’s lives better) and eco-centric concerns (protecting the environment). Rather than asking whether it is appropriate to balance these concerns, we should instead be asking What methods and tools are needed to explore inter-dependencies between these concerns, and to foster more
integrated and long-term thinking?

In the past few decades, production and use of information technologies (IT) have had a dramatic effect on society, giving us new tools and new capabilities, but also generating a massive growth in demand for energy and other resources. Software systems, in particular, play a transformative role, as they enable dematerialization [33], drive consumption patterns for products, services, materials, and energy, and facilitate structural changes from consuming material goods towards consuming immaterial services, such as the shift from listening to music online instead to purchasing and discarding hard copies. They also collect, manage and distribute information needed to understand long-running complex phenomena ranging from climate data to personal health records and statistics on global equity and capital. As such, the software industry increasingly represents a central driver for innovation and economic prosperity, but simultaneously increases social inequity, as people without access and technical skills are left behind [34], and causes environmental damage, as consumption of technology grows [35].

The approach we have adopted is to focus on how we understand and take responsibility for the multiple interacting opportunities and impacts of software technology, including first, second and third order effects [36]. First order effects are impacts and opportunities created by the immediate existence of a software system, arising from its design features and flaws. Second order effects are those created by the ongoing use and application of the software, such as how it changes what we do and what we’re capable of. Third order effects are the changes that occur through the aggregated behaviours of very large numbers of people using the technology over the medium to long term (e.g., energy demand, mass surveillance, etc). These effects play out across many domains. Following Goodland [37] and Penzenstadler and Femmer [38], we identify five sustainability dimensions:

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

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

A growing concern among software researchers about these impacts, and a desire to find good leverage points, has inspired a number of workshops at software-related conferences dedicated to dis-aggregating and analyzing relevant issues.

---

1 We observe that these dimensions are interdependent. Cumulative effects from the individual dimension will bleed into the social one; effects from the environmental dimension into the individual, social, and economic, and so on. Yet, these dimensions provide a useful tool for dis-aggregating and analyzing relevant issues.
cated to software and sustainability. For example, at ICSE’09, a special conference session explored the relationship between Software Engineering and climate change, which led to a series of workshops on software research and climate change (WSRCC), at Oopsla/Onward! in 2009, at ICSE in 2010, and at ECOOP in 2011. These led to a special issue of IEEE Software [40]. As interest grew and broadened, the community was brought together under the umbrella of the GREENS (Green & Sustainable Software) workshop series at ICSE through a workshop merger, and held in 2012, 2013, and 2014, and another series of workshops on RE for Sustainable Systems (RE4SuSy) at Refsq in 2012 and at the RE Conference in 2013 and 2014.

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

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

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

What makes a manifesto a successful ‘point of reference’? Historically, a common vehicle for catalyzing communities and providing such a focus in comparable situations has been the genre of the manifesto. Communities have relied upon it to articulate their viewpoint, often as an alternative one opposed to a prevailing paradigm. Some manifestos very effectively captured key messages with an appeal beyond the originating community and hence provided a platform for subsequent thoughts and initiatives to develop. Examples in the software world include the GNU2 and the Agile Manifesto3. However, these manifestos have little in common in terms of their structure and content, and the manifesto is a delicate genre. What makes a manifesto a successful ‘point of reference’?

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

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

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

However, the manifesto has also been called a ‘defunct format’ that ‘belongs to the early twentieth century’

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

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

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

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

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

Participation and transparency. The discussion was initiated within the workshop, and all participants of the workshop were invited to the subsequent process. No conditions are set for entering the discussion process. The initial document was released publicly for comments and presented at the RE closing session, and direct discussions with a number of experts in the fields of sustainability have been initiated. Broader engagement with the community includes a discussion panel held at the SPLC’14 conference [44], a workshop proposed for the iConference 2015, a discussion at WSSSPE2, and a special discussion session planned for RE4SuSy’2015.

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

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

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

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

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

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

IV. THE KARLSKRONA MANIFESTO

---

4See [43] for a cultural perspective on manifestos in art.
6http://ischools.org/the-iconference/
7http://wssspe.researchcomputing.org.uk/wssspe2
8This paper was developed in similar fashion on overleaf.com.
THE KARLSKRONA MANIFESTO
FOR SUSTAINABILITY DESIGN

Version 0.5, January 2015

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

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

There is a narrow perception of sustainability that frames it as protecting the environment or being able to maintain a business activity. Whereas as a systemic property, sustainability does not apply simply to the system we are designing, but most importantly to the environmental, economic, individual, technical and social contexts of that system, and the relationships between them. There is a perception that sustainability is a distinct discipline of research and practice with a few defined connections to software. Whereas sustainability is a pervasive concern that translates into discipline-specific questions in each area it applies.

There is a perception that sustainability is a problem that can be solved, and that our aim is to find the “one thing” that will save the world. Whereas it is a “wicked problem” - a dilemma to respond to intelligently and learn in the process of doing so; a challenge to be addressed, not a problem to be solved. There is a perception that there is a tradeoff to be made between present needs and future needs, reinforced by a common definition of sustainable development, and hence that sustainability requires sacrifices in the present for the sake of future generations. Whereas it is possible to prosper on this planet while simultaneously improving the prospects for prosperity of future generations.

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

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

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

There is a tendency to treat sustainability as a desirable quality of the system that should be considered once other priorities have been established. Whereas sustainability is not in competition with a specific set of quality attributes against which it has to be balanced - it is a fundamental precondition for the continued existence of the system and influences many of the goals to be considered in systems design. There is a desire to identify a distinct completion point to a given project, so that success can be measured at that point, with respect to a pre-ordained set of criteria. Whereas measuring success at one point in time fails to capture the effects that play out over multiple timescales, and so tells us nothing about long-term success. Criteria for success change over time as we experience those impacts.

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

There is a tendency to interpret the codes of ethics for software professionals narrowly to refer to avoiding immediate harm to individuals and property. Whereas it is our responsibility to address the potential harm from the 2nd and 3rd-order effects of the systems we design as part of our design process, even if these are not readily quantifiable. As a result, even though the importance of sustainability is increasingly understood, the majority of software systems are created unsustainably and often decrease sustainability instead of increasing it.

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

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

Sustainability has multiple dimensions. We have to include those dimensions into our analysis if we are to understand the nature of sustainability in any given situation.

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

Sustainability is a concern independent of the purpose of the system. Sustainability has to be considered even if the primary focus of the system under design is not sustainability.

Sustainability applies to both a system and its wider contexts. There are at least two spheres to consider in system design: the sustainability of the system itself and how it affects the sustainability of the wider system of which it will be part of.

System visibility is a necessary precondition and enabler for sustainability design. Strive to make the status of the system and its context visible at different levels of abstraction and perspectives to enable participation and informed responsible choice.

Sustainability requires action on multiple levels. Seek interventions that have the most leverage on a system and consider the opportunity costs: Whenever you are taking action towards sustainability, consider whether this is the most effective way of intervening in comparison to alternative actions (leverage points).

It is possible to meet the needs of future generations without sacrificing the prosperity of the current generation. Innovation in sustainability can play out as decoupling present and future needs. By moving away from the language of conflict and the trade-off mindset, we can identify and enact choices that benefit both present and future.

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

Signed,
V. Implications for Software Engineering

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

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

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

Consider an imaginary software company called CodeIT. Their next project is developing a community car sharing application that specifically intends to satisfy the needs of a suburban community in a western country such as the US. Kodi, the project manager, is sensitive to the impact of car booking becomes as efficient and effortless as using one’s own car, though, car traffic could ultimately even increase, marginalizing social, economic, technical, and natural environment. Sustainability requires action on multiple levels. Sustainability transcends multiple disciplines. Sustainability design in this scenario will require cross-disciplinary expertise covering transportation systems, carbon emissions, social network effects, effects on family structures, but most importantly, the interaction between these and additional aspects. The problem of transportation in suburban communities is often a dilemma to be addressed rather than a problem to be solved, and success may be a moving target.

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

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

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

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

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

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

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

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

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

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

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

VI. Conclusions and Outlook

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

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

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

VII. Acknowledgments

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

REFERENCES


