Sustainability and Requirements: A Manifesto

Birgit Penzenstadler

WRITE A MANIFESTO? If you have something you want to shout out to the world and have heard. If you want to change something in the world but need many people to join you because otherwise there won’t be much impact. If you feel like starting a conversation and getting ideas into people’s minds for them to improve their output in some form or another.

The idea I want to get into people’s minds is the objective of sustainability, and as software engineers, our output is software systems. If we’re to get this new objective into our systems, the requirements must reflect it. So, the manifesto I talk about might significantly affect how developers perform requirements engineering.

The Genesis of a Manifesto
The history of manifestos goes back to as early as the 17th century. Galileo’s 1623 treatise *The Assayer*, on how science should be practiced, is called his scientific manifesto. The *Communist Manifesto* explored a radical political and societal alternative that emerged from a program for the Communist League. The Holstee Manifesto is a reminder of the values we live and work for.1 The Agile Manifesto summarizes guiding principles for software development derived from experience.2

“The Karlskrona Manifesto for Sustainability Design” is a call for discussion and action on the challenge of sustainability and its relation to software engineering.3 The manifesto aims to create common ground and develop a reference point for the global community of research and practice in software and sustainability.

The manifesto’s creation began at a working session at the Third International Workshop on Requirements Engineering for Sustainable Systems, held at the 22nd IEEE International Requirements Engineering Conference, in Karlskrona, Sweden. Over the next nine months, the emerging working group met online weekly to discuss and develop the manifesto. The group published a subset of the versions on arxiv.org3 and presented version 1.0 in a research paper at the 37th International Conference on Software Engineering.4 The vividly written paper describes the motivation for developing the manifesto and illustrates the collaborative reflective writing process. It assesses immediate implications and applications of the articulated principles, compares these to current practice, and suggests future steps.

A Brief Look at the Manifesto
Sustainability has been defined in many ways; the plethora of definitions and
the concept’s abstractness make for controversial discussions. The main reason is often unclear scoping. As anthropologist Joseph Tainter pointed out, when we talk about sustainability, we need to specify four things: What should be sustained? For whom? For how long? At what cost?5

The manifesto’s authors argue that we often define sustainability too narrowly. Instead, sustainability is at heart a systemic concept and must be understood on a set of dimensions, as I discuss in more detail later. The authors don’t proclaim an easy solution around the corner but rather point out ways in which sustainability is a wicked problem.

In particular, they explain 11 common misperceptions of sustainability. Three of these misperceptions are

- framing sustainability as only protecting the environment;
- dismissing sustainability as a distinct research discipline; and
- thinking that taking small steps toward sustainability is sufficient, appropriate, and acceptable.

Each of these misperceptions leads to problems. Framing sustainability as only protecting the environment ignores social and economic concerns. Dismissing it as a distinct research discipline leads to ignoring the topic within software-engineering research and practice. Relying on small steps can easily lead to great optimizations of one solution when a different solution might bring more benefits. For example, the car industry has made great progress at developing energy-efficient engines. However, to reduce emission rates globally, it would be more beneficial to improve the infrastructure for shared transport such that fewer vehicles are on the road.

Because of these misperceptions, even though sustainability’s importance is increasingly recognized, many software systems are unsustainable. Even more alarming is that most software systems’ broader impacts on sustainability are unknown. To change this, the Karlskrona Manifesto proposes these principles and commitments:

- **Sustainability is systemic.** Sustainability is never an isolated property. Systems thinking must be the starting point for the transdisciplinary common ground of sustainability.
- **Sustainability is multidimensional.** Our analysis must include the economic, social, environmental, technical, and individual (personal) dimensions if we’re to understand the nature of sustainability in any given situation.
- **Sustainability is interdisciplinary.** Working on sustainability means working with people across many disciplines, addressing the challenges from multiple perspectives.
- **Sustainability transcends the system’s purpose.** Sustainability must be considered even if the primary focus of the system...
under design isn’t sustainability, because the use of any software can impact its environment.

• **Sustainability applies to both a system and its wider contexts.** System design involves at least two spheres: the sustainability of the system itself and how it affects the sustainability of the wider system of which it will be part.

• **Sustainability requires action on multiple levels.** Some interventions have more leverage on a system than others. Whenever we take action toward sustainability, we should consider opportunity costs: action at other levels might offer more effective forms of intervention.

• **Sustainability requires multiple timescales.** We should assess the benefits and impacts on multiple timescales, and include longer-term indicators in assessment and decisions.

• **Changing design to take into account long-term effects doesn’t automatically imply sacrifices.** However, innovation in sustainability can play out as decoupling present and future needs. Instead, we can often identify opportunities and changes that benefit both the present and future.

• **System visibility is a precondition for and enabler of sustainability design.** The system’s status and context should be visible at different abstraction levels and from different perspectives to enable participation and informed responsible choice.

Finally, the manifesto proposes concrete ideas of how to start implementing these principles in different roles in the software-engineering community: software practitioners, researchers, professional associations, educators, customers, and users. Regarding your software practitioners, try to identify your projects’ effects on technical, economic, and environmental sustainability. Ask about how to incorporate the principles into daily practice. Think about the social and individual dimensions. Talk about this with your colleagues.

Will this manifesto change how we elicit and analyze requirements? Will it change how we design software? Will it have a big impact on software engineering’s future? We’ll know that in a few years. However, the main objective of initiating a discussion about sustainability depends strongly on you, valued reader. Start with reading the manifesto at www.sustainabilitydesign.org, become a signatory if you agree, and join the discussion on Twitter at #KarlskronaManifesto and #SustainabilityDesign.

**References**


**BIRGIT PENZENSTADLER** is a professor of software engineering at California State University, Long Beach. Contact her at birgit.penzenstadler@csulb.edu.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.