Software Sustainability: Beyond the Tower of Babel

Colin C. Venters
Uni. of Huddersfield
United Kingdom
c.venters@hud.ac.uk

Sedef Akinli Kocak
Vector Institute for AI
Canada
sedef@vectorinstitute.ai

Stefanie Betz
Furtwangen University
Germany
besi@hs-furtwangen.de

Ian Brooks
UWE Bristol
United Kingdom
ian.brooks@uwe.ac.uk

Rafael Capilla
URJC
Spain
rafael.capilla@urjc.es

Ruzanna Chitchyan
Uni. of Bristol
United Kingdom
r.chitchyan@bristol.ac.uk

Letícia Duboc
La Salle
Spain
duboc@lasalle.es

Rogardt Heldal
Western Norway Uni. of Applied Sci.
Norway
rogardt.heldal@hvl.no

Ana Moreira
NOVA Uni. Lisbon
Portugal
amm@fct.unl.pt

Shola Oyedeji
Lappeenranta-Lahti University
Finland
oyedeji@lut.fi

Birgit Penzenstadler
Chalmers Uni. of Technology
Sweden
birgitp@chalmers.se

Jari Porras
Lappeenranta-Lahti University
Finland
jari.porras@lut.fi

Norbert Seyff
FHNW
Switzerland
seyff@fhnw.ch

I. Context

Principally associated with the field of ecology, the topic of sustainability has emerged as an increasingly important area of research in a number of sub-fields within the domain of computing including artificial intelligence, high-performance computing, human-computer interaction, requirements engineering, and scientific computing [9]. Within the field of software engineering, sustainability has been identified as an important topic given modern society's high dependency on increasingly complex and 'dangerously fragile' software systems, which operate in evolving, distributed eco-systems [17].

In relation to software, there exist at least two distinct viewpoints for the topic area of software and sustainability: sustainable software and software engineering for sustainability (SE4S) [13], [5]. The former is concerned with the principles, practices, and processes that contribute to software endurance, i.e. technical sustainability, and the latter focuses on software systems to support one or more dimensions of sustainability, concerning issues outside the software systems itself. While a number of communities have attempted to address the challenges of achieving sustainability from their different perspectives, there is a severe lack of common understanding of the fundamental concepts of sustainability and how it relates to software systems. As a result, there is no agreed definition of software sustainability or how it might be achieved. While there have been a number of contributions to formalise a definition of software sustainability, the concept remains an elusive and ambiguous term with individuals, groups and organisations holding diametrically opposed views [18]. This lack of clarity ultimately leads to confusion, and potentially to ineffective and inefficient efforts to develop sustainable software systems. Fundamental to the establishment of a Body of Knowledge of Software Sustainability is a clear definition of software sustainability. Our aim is to explore the range of definitions of software sustainability in order for the software engineering community to move towards a consensus on how software sustainability is defined and understood. As such, this paper argues that the advancement of software sustainability as a field of research requires a shared and common understanding of the concept.

II. Problem

What does software sustainability mean in the field of software engineering? Cohen et. al [4] exemplify the issue, where they argue that maintainability, [sustainability], and robustness are core aspects of building [sustainable software], which is underpinned by ensuring a [sustainable] approach to software development. While few would argue against maintainability being a core internal quality of a software system, the terms sustainability and sustainable are not defined in context, leaving them open to [mis]interpretation. In addition, little evidence or practical guidance is offered in the way of what a sustainable approach to software development is, how this can be achieved in practice or how sustainability can be measured or demonstrated. This raises the question, are the terms and process by which software sustainability can be achieved widely understood or have they been reduced to vacuous platitudes?

It is suggested that sustainability is generally understood as the capacity of a socio-technical system to endure [6], which aligns with how the term is defined in modern English [15]. An additional perspective was presented by Lago et. al [11] who suggested that sustainability was not only the

capacity to endure but to "preserve the function of a system over an extended period of time [10]. This aligns with the term's Latin origin, "sustinere", i.e. to maintain. If we accept the argument that software sustainability is the "capacity to endure", this leads to the natural question of over what time frame? It has been suggested that a sustainable software is a software-intensive system that operates for an average software lifetime of ten and a maximum of thirty years [16]. How this time frame was derived is unclear. Nevertheless, both these perspectives suggest that endurance, i.e. longevity, as an expression of time, and the ability to maintain are key factors at the heart of understanding sustainability. However, viewing software sustainability simply in terms of the software systems capacity to endure over a specified time period requires a clear mechanism by which it can be demonstrated at design time. In addition, further research is required to confirm or refute this position as evidence suggests that software engineering practitioners' tended to have a narrow understanding of the concept of sustainability with a focus on environmental and economic sustainability [3].

In contrast to viewing software sustainability as a measure of endurance, a number of definitions have emerged from the field of software engineering, which align software sustainability to one or more software quality attributes that contribute to the sustainability of the software artefact including maintainability and extensibility [14], [8]. One of the principal challenges in defining sustainability as a non-functional requirement is how to demonstrate that the quality factors have been addressed in a quantifiable way. Nevertheless, it is argued that maintainability and evolution of the software artefact are key enablers to achieving long-living software [7]. As such, it has been argued that while this addresses the technical dimension of sustainability in developing long-living software, sustainability applies to both the system and its wider contexts. As a result, there is a need to take into account the direct and indirect negative impacts on the economy, society, individuals, and environment that result from the development, deployment, and continued use of the software system [12], [11], [1], [6]. This suggest that a sustainable software product can only be achieved if the negative and positive impacts on the dimensions of sustainability are taken into account and can be demonstrated. While a number of initiatives have attempted to address this, it remains an open and challenging research issue [5], [6].

III. Solution

Before the sustainability of a software system can be measured, it is argued that it must be understood [14]. Defining software sustainability as its capacity to endure is overly simplistic and requires greater precision and clarity. We reject the view proposed by [2] that sustainability is simply another software quality in its own right, which has equivalence to other software qualities such as maintainability or extensibility. From a purely technical perspective, we propose that software sustainability is a defined as a composite, first-class, non-functional requirement that is a measure of a range of core software quality attributes, which includes at a minimum, maintainability, extensibility, and usability [14], [18], [8]. This position aligns with the Karlskrona Manifesto for Sustainability Design [1], which argues that sustainability is an overarching concern even if the primary focus of the system under design is not sustainability, i.e. a concern independent of the purpose of the system, which requires action on multiple levels. Further research is required by the community to confirm or refute this position. However, while this addresses the technical dimension of sustainability is does not account for the alignment with the other dimension of sustainability and the need to take into account the direct and indirect negative impacts on different dimensions of sustainability that result from the development, deployment, and continued use of the software system. How the software engineering community want to define and measure the sustainability of software systems remains an open issue that must be addressed in the development of a Body of Knowledge on Software Sustainability.

References

[1] C. Becker et al. Sustainability design and software: The karlskrona manifesto. In *ICSE 2015: IEEE/ACM 37th Int. Conf. on Soft. Eng.*, volume 2, pages 467–476, 2015.

[2] C. Calero, M. F. Bertoa, and M. Á. Moraga. A systematic literature review for software sustainability measures. In *Greens 2013: 2nd Int. Workshop on Green and Sustainable Soft.*, pages 46–53, 2013.

[3] R. Chitchyan et al. Sustainability design in requirements engineering: State of practice. In *ICSE 2016: IEEE/ACM 38th Int. Conf. on Soft. Eng.*, pages 533–542, 2016.

[4] J. Cohen et al. The four pillars of research software engineering. *IEEE Soft.*, 38(01):97–105, jan 2021.

[5] N. Condori-Fernandez and P. Lago. Towards a software sustainability-quality model: Insights from a multi-case study. In *2019 13th Int. Conf. on Research Challenges in Info. Sci.*, pages 1–11, 2019.

[6] L. Duboc et al. Requirements engineering for sustainability: an awareness framework for designing software systems for a better tomorrow. *Requirements Engineering*, 25(4):469–492, 2020.

[7] Z. Durdik et al. Sustainability guidelines for long-living software systems. In *ICSM 2012: 28th IEEE Int. Conf. on Soft. Maintenance ()*, pages 517–526, 2012.

[8] I. Groher and R. Weinreich. An interview study on sustainability concerns in software development projects. In *SEAA 2017: 43rd Euromicro Conf. on Soft. Eng. and Adv. Applic.*, pages 350–358, 2017.

[9] L. M. Hilty and B. Aebischer. *ICT Innovations for Sustainability*. Springer International Publishing AG, 2014.

[10] L. M. Hilty et al. The relevance of information and communication technologies for environmental sustainability – a prospective simulation study. *Env. Modelling & Soft.*, 21(11):1618 – 1629, 2006.

[11] P. Lago et al. Framing sustainability as a property of software quality. *Comm. ACM*, 58(10):70–78, September 2015.

[12] S. Naumann et al. The greensoft model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Sys.*, 1(4):294 – 304, 2011.

[13] B. Penzenstadler. Towards a definition of sustainability in and for software engineering. In *ACM Symposium on Applied Computing*, pages 1183–1185, 2013.

[14] R. C. Seacord et al. Measuring software sustainability. In *ICSM 2003: Int. Conf. on Soft. Maintenance*, pages 450–459, 2003.

[15] Sustainability. *Oxford English Dictionary*. Oxford Univ. Press, 2021.

[16] T. Tamai and Y. Torimitsu. Software lifetime and its evolution process over generations. In *Proc. Conf. on Soft. Maint.*, pages 63–69, 1992.

[17] C. Venters et al. Software sustainability: Research and practice from a software architecture viewpoint. *J. of Sys. & Soft.*, 138:174–188, 2018.

[18] C. C. Venters, C. Jay, L. M S Lau, M.K. Griffiths, V. Holmes, R. R. Ward, J. Austin, C. E. Dibsdale, and J. Xu. Software sustainability: The modern tower of babel. *CEUR Workshop Proc.*, 1216:7–12, 2014.