The First Rule of Software Sustainability: Do not talk about Software Sustainability?

Colin C. Venters*, Christoph Becker‡, Stefanie Betz§, Ruzanna Chitchyan¶, Leticia Duboc‖, Steve M. Easterbrook‡, Birgit Penzenstadler**, Guillermo Rodriguez-Navas††, and Norbert Seyff‡‡

*University of Huddersfield, UK, c.venters@hud.ac.uk
‡University of Toronto, Canada, {christoph.becker@utoronto.ca; sme@cs.toronto.edu}
§Karlsruhe Institute of Technology, Germany, stefanie.betz@kit.edu
¶University of Leicester, UK, rc256@le.ac.uk
‖State University of Rio de Janeiro, Brazil, leticia@ime.uerj.br
**California State University Long Beach, USA, birgit.penzenstadler@csulb.edu
††Mälardalen University Västeras, Sweden, guillermo.rodriguez-navas@mdh.se
‡‡University of Zurich, Switzerland, seyff@ifi.uzh.ch

***Abstract*—Principally associated with the field of ecology in order to address humanities increasing ecological footprint on the planet Earth, sustainability as a concept has emerged as an area of interest in the field of computing. While a number of related communities have attempted to understand and address the challenges of sustainability from their different perspectives, there is a fundamental lack of understanding of: the concept of sustainability; how it relates to software artifacts and software systems; and the wider implications of the software development process and the impact of software products on the different dimensions of sustainability. This position paper argues that current efforts to address the sustainment i.e. longevity, of software and software systems are futile because of a lack of a common definition of sustainable software, which results in a misalignment with established software engineering theory and best practice that enables software products to endure. While there have been a number of previous attempts to define sustainability no consensus has been reached. As a result, the term remains illusive and ambiguous. This paper argues that the primary barrier to making progress in the field of sustainable software systems engineering is the lack of a concrete definition, which lays the foundation to understand how to design software that is [technically] sustainable. As a basis for discussion we propose a definition of sustainable software as a first-class, composite, non-functional requirement that includes as its basic building blocks the base sub-characteristics of maintainability and extensibility.**

***Index Terms*—Software sustainability, software engineering, sustainability, technical sustainability, non-functional requirements, maintainability, extensibility**

I. Introduction

Sustainable software. Software sustainability. Sustainable software development. Sustainability requirements. Sustainable HPC. Sustainable human-computer interaction. Software for sustainability. The concept of sustainment has become a topic of interest in the field of computing, which is evidenced by the rise in publication output [1], and the increase in the number of events that focus on the topic of sustainability [1] [2] [3] [4] [5] [6] [7] [8] [9]. However, despite this increasing interest in the topic, the concept of sustainability is not well understood in the field of computing. Definitions of the concept range from a measure of time [2] to simply an emergent property [3]. A more thorough treatment of the range of definitions of software sustainability is discussed by Venters et. al., [4]. As a result there is no consensus on how sustainability might be achieved with regards to the design of software artifacts and software systems, or how software sustainability might be quantified. While this position is not unique to the field of computing, this paper argues that the lack of clarity about the concept of software sustainability will lead to ineffective and inefficient efforts to address the concept or result in its complete omission from the software system [5]. Sustainability can be viewed from a range of different dimensions including environmental, economic, individual, social and technical [6], [5]. In this paper we focus primarily on the concept of sustainability from a technical perspective. We propose that the main barrier and one important challenge for the field of computing is to define the concept of sustainability. From this we can derive how to design sustainable software from the existing body of software engineering knowledge, identify gaps in software engineering theory and best practice, all of which leads to forming an understanding of the basis of a educational curriculum, which lays the foundation for training and educating the broad spectrum of domain scientists or advance the skills of software engineers to develop software that is sustainable. The basis for how this might be achieved is discussed in the following section, where a definition of software sustainability from a technical perspective is proposed along with a number of related challenges.

[1] AAAI Computational Sustainability, http://www.aaai.org/Conferences/AAAI/
[2] GREENS, http://greens.cs.vu.nl
[3] GInSEng, http://alarcos.esi.uclm
[4] ICT4S, http://ict4s.org
[5] RE4SuSy, http://web.csulb.edu/bpenzens/re4susy/
[6] S-HCI, http://www.sigchi.org/communities/hci-sustainability
[7] SHPC, https://sites.google.com/a/temple.edu/shpc2015/
[8] SSSE, http://http://sustainabilitydesign.org/ssse15
[9] WSSSPE, http://wssspe.researchcomputing.org.uk/

II. Sustainable Software

Before sustainable software can be designed it first must be understood [7]. What does sustainability mean in the context of software engineering? Despite a number of potential candidates, little progress has been made towards defining the concept and any consensus has still to be reached [8]. A generally accepted definition of sustainability is the capacity to endure that is a measurement of time. However, the generality of this definition renders it meaningless for software engineers and developers to enact upon, and fails to capture the complexity of a concept that can be viewed from a range of different dimensions [9]. In addition, it does not get at the heart of what makes a software product endure over time. What are the underlying factors that result in software sustainment? There is growing consensus that sustainability should be considered as a first-class, non-functional requirement [10] [11]. While this has been supported by a number of commentators [12] [13], it has been made without explicit reference to the characteristics or qualities that sustainability would be composed of. In contrast, Venters et. al., [14] and Chitchyan, Cazzola and Rashid [15] consider software sustainability as a composite, non-functional requirement, which is a measure of a number of quality attributes of a systems. We propose that at the very minimum, the genetic building blocks of [technically] sustainable software should address two core quality attributes:

- **Extensibility**: the software's ability to be extended and the level of effort required to implement the extension;
- **Maintainability**: the effort required to locate and fix an error in operational software.

These fundamental building blocks could then be extended to include other quality attributes such as portability, resuability, scalability, usability, and energy efficiency etc. Nevertheless, this raises the question of what metrics and measures are suitable to demonstrate the sustainability of the software. A related challenge is therefore how to demonstrate that the quality factors have been addressed in a quantifiable way. This is an open question and provides an avenue for further research.

III. Beyond the Tower of Babel

This paper argued that the lack of consensus regarding the concept of sustainability is the principal barrier to progress in the design of software that is [technically] sustainable. To address this, we proposed that sustainability, in the context of software engineering, is a first-class, composite, non-functional requirement, that includes as its basic building blocks the base sub-characteristics of maintainability and extensibility. However, we recognize that this is a rather narrow view of sustainability and does not take into account other dimensions [16].

The call for participation to CSESSP identified the need for new approaches to scientific software that significantly improves sustainability and productivity, including leveraging software engineering research.This suggests that current software engineering techniques and practices are inadequate. This argument is simply without merit. There is a substantive body of knowledge available to draw upon in the design of sustainable software to whoever avails of it. This raises the question of why the existing software engineering body of knowledge, such as that contained in SWEBOK [17], is largely ignored by the computational scientific and engineering communities.

Beyond the definition of sustainable software, the challenge is how to distill this large body of knowledge into a set of core software engineering principles and accompanying practices that underpin the concept of sustainability that developers of software products can incorporate into their development streams as well as how different areas relate to sustainability as a whole.

References

[1] B. Penzenstadler *et al.*, "Systematic mapping study on software engineering for sustainability (se4s)," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2014.

[2] H. Koziolek, "Sustainability evaluation of software architectures: a systematic review," in *Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS*. ACM, 2011, pp. 3–12.

[3] WSSSPE'1, "Wssspe'1 collaborative notes." [Online]. Available: bit.ly/wssspe13

[4] C. C. Venters *et al.*, "Software sustainability: The modern tower of babel," in *RE4SuSy: Proceedings of the Third Int. Workshop on RE for Sustainable Systems*. Karlskrona, Sweden: CEUR-WS 1216, 2014.

[5] B. Penzenstadler and H. Femmer, "A generic model for sustainability with process-and product-specific instances," in *Proc. of the 2013 workshop on Green in/by software engineering*. ACM, 2013, pp. 3–8.

[6] R. Goodland, "Sustainability: Human, social, economic and environmental," in *Encyclopedia of Global Environmental Change*, T. Munn, Ed. John Wiley & Son, 2002.

[7] C. C. Venters *et al.*, "The nebuchadnezzar effect: Dreaming of sustainable software through sustainable software architectures," figshare, Tech. Rep. 1112484, 2014, http://dx.doi.org/10.6084/m9.figshare.1112484.

[8] D. S. Katz *et al.*, "Summary of the first workshop on sustainable software for science: Practice and experiences (WSSSPE1)," *Journal of Open Research Software*, vol. 2, 2014.

[9] C. Becker *et al.*, "Requirements: The key to sustainability," 2015, unpublished.

[10] N. Amsel *et al.*, "Toward sustainable software engineering," in *ICSE: Proceedings of the 33rd International Conference on Software Engineering*, Waikiki, Honolulu, HI, USA, 2011, pp. 976–979.

[11] B. Penzenstadler *et al.*, "Safety, security, now sustainability: The non-functional requirement for the 21st century," *Software, IEEE*, vol. 31, no. 3, pp. 40–47, May 2014.

[12] S. Naumann *et al.*, "The greensoft model: A reference model for green and sustainable software and its engineering," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 4, p. 294–304, December 2011.

[13] C. Calero, M. F. Bertoa, and M. Ángeles Moraga, "Sustainability and quality: Icing on the cake," in *RE4SuSy: Proceedings of the Second International Workshop on RE for Sustainable Systems*, Rio de Janeiro, Brazil, 2013.

[14] C. C. Venters *et al.*, "The blind men and the elephant: Towards an empirical evaluation framework for software sustainability," *Journal of Open Research Software*, vol. 2, pp. 1–6, 2014.

[15] R. Chitchyan, W. Cazzola, and A. Rashid, "Engineering sustainability through language," in *ICSE'15: Proceedings of the International Conference on Software Engineering*, Florence, Italy, 2015.

[16] C. Becker *et al.*, "Sustainability design and software: The karlskrona manifesto," in *ICSE'15: Proceedings of the International Conference on Software Engineering*, Florence, Italy, 2015.

[17] P. Bourque and R. E. Fairley, Eds., *SWEBOK version 3.0: Guide to the Software Engineering Body of Knowledge*. IEEE Press, 2014.