A Utility Model for Designing Environmentally Sustainable Software

Sedef Akinli Kocak
Vector Institute for Artificial Intelligence
Toronto, Canada
sedef.kocak@vectorinstitute.ai

Gulfem Isiklar Alptekin
Computer Engineering Department
Galatasaray University
Istanbul, Turkey
gisiklar@gsu.edu.tr

***Abstract*—Software systems play an increasingly central role in sustainability, since many aspects of our lives and society as a whole are mediated through software systems. Hence, the design of these systems requires attention from sustainability perspective. Recent interest in sustainability as a requirement has given rise to broader definitions in terms of technical, economic, environmental and individual perspectives. In this sense, there is a need for additional quality requirements to include sustainability concerns, as well as to examine the market equilibrium where stakeholders and customers meet. In Most cases, software development companies do not consider the explicit representation of sustainability requirements such as resource utilization, or customers concerns on sustainability. In this work, we focus on the relationship between customers, who require energy efficient software products, and software development companies, who offer these software products. The proposed introductory model aims to determine the demand and payoff functions by satisfying both sides with the quality level of the product and its price.**

***Index Terms*—Environmental Sustainability, Utility Model, Software Quality, Market Equilibrium, Green Software.**

I. Introduction

Sustainability, the "capacity to endure" [1] is a key issue facing society on multiple levels, from individuals and social groups to large socio-technical systems and the planet earth. As a concept, it concerns a specific system and has to be considered along multiple dimensions such as environmental, economic, social, technological and individual [29]. Humanity faces many complex challenges that present risks to societies, including the rise of CO_2 emission, biodiversity loss, cybersecurity, inequality and unfairness. Environmental risks have grown in prominence in recent years. It is fundamentally a systemic property, and any effort to address sustainability involves integrating concepts, principles, and methods from a range of disciplines.

In the context of software engineering, sustainability is often defined as to "preserve the function of a system over an extended period of time" [25]. As many aspects of our lives and society as a whole are mediated through software systems, sustainability is becoming a challenge for requirements and software professionals [5]. Although requirements engineering (e.g., [23] [34]) and software architecture researchers (e.g., [4] [36]) have been devoting effort to define the basis of the notion of sustainability-aware software, there are still a lack of suitable approaches to design software-intensive systems that enable sustainability goals.

In software engineering, sustainability is mostly related to technical impact (e.g., [3]), economic interests (e.g., [30], and recently environmental concerns (e.g. [27], [23]). Recent interest in sustainability as a requirement has given rise to broader definitions in terms of technical, economic and environmental perspectives [35]. In this sense, sustainability may lead to have additional quality requirements. New instruments (e.g., [19], [24], [22], [2]) have designed framing the design concerns around the four sustainability dimensions- technical, economic, social and environmental sustainability. We believe that decision-oriented models are one of the significant ways of examining the sustainability concerns of stakeholders.

In this paper, we present an introductory model which is part of a continuous work on designing decision-oriented model focusing on relevant quality requirements and contributing to the sustainability dimensions of software-intensive systems. In the proposed model, we assume that the software product customers have two main decision variables, when buying a software product: Price and quality. It is obvious that the buyers may have various other concerns when making the decision of buying, but for the sake of constructing a manageable demand function, we only select the most two important ones. The quality variable comprises all non-functional requirements of the offered software. As the price of the product increases, the customer‘s willingness to buy decreases, while the quality of the product increases, the customer‘s willingness to buy increases. Accordingly, we introduce a linear demand function for software development companies. Moreover, we introduce an acceptance probability variable for customers, that is maximized when the price of the product decreases, while it‘s quality increases. We propose that this approach may be suitable in the context of software quality requirements, since they are always subject to negotiation between customers/users and development company. In our model, the quality requirements involve sustainability concerns. The proposed introductory model is based on interactive multi-criteria decision making theory [11], [37] which focuses on the structure of multi-criteria or multi-attribute alternatives, usually in the presence of conflicting criteria. Therefore, the strategic interaction of customer and development company as well as the interdependencies of quality attributes and sustainability requirements are essential. The rest of this paper is structured as follows: Section 2 gives a more detailed description of what motivates this study and introduces the corresponding background. Section 3 describes the basic elements of the introductory model. We present a green software product scheme in Section 4, and we conclude the paper in Section 5 with a summary of the study and continuing directions of the research.

II. Related Work and Motivation

In this section, we have categorized the works related to our research topic under two categories: quality models in software engineering and software engineering and sustainability. Researchers have been studying sustainability in software engineering from different perspectives such as hardware energy efficiency in terms of power consumption; optimization of algorithms and / or software architecture; effective and efficient design and usage of software intensive system, and models to support decision making including sustainability.

There are rising attention and increasing research works in the field of software engineering and sustainability focusing on specific software sustainability aspects. For example, Hindle [15] related the direct impact of software change on energy consumption; Procaccianti et al. [31] related software code metrics and software architecture for assessing the impact of best practices for achieving software energy efficiency; Cai et al. [7] investigated the notion of design rules to detect flows at the software architecture level over extended periods; Lago et al. [24], [23] introduced a decision map can be used to frame the concerns of each of the sustainability dimensions to support decision making.

The recent quality model/standards are introduced by ISO (ISO/9126 and ISO/IEC 25010) [16]. In these standards, the sustainability issues are not considered as a separate criteria. Several sustainability related issues may be the consumption of energy, memory requirements, efforts (in terms of energy) required during its development, etc. In the software engineering literature, the first quality model for green and sustainable software was developed by Kern et al. [17]. The model considers the product quality factors, however, the quality aspects standardized in ISO /IEC 25000 are also related to the quality of software in use. Calero and Bertoa [8] considered sustainability as a new factor that affects software product and process quality. They presented a new quality model (ISO 2510+S) based on ISO/25010. In their model, they differentiated the quality factors concerning the sustainability impact and they described related and unrelated sub-characteristics. A cross-disciplinary initiative to create a common ground and develop a focal point of reference in software and sustainability is proposed by Becker et al. [5]. They discuss different interpretations of sustainability. Akinli Kocak et al. [19] analyzed the correlation between the standardized quality attributes and environmental sustainability attributes to identify their effects on environmental sustainability. Recently, Condori-Fernandez and Lago [10] developed a Software Sustainability-Quality Model to characterize the contribution of quality requirements to software sustainability. All these studies point out that the product, as well as the quality in use needs to be considered when assessing the sustainability of the software. However, knowledge about designing and configuring software in an environmental manner is not sufficient today [14]. Moreover, assessment based on the notion of sustainability as a software quality property is still emerging and poorly understood [9].

Companies are increasingly aware of many potential benefits provided by customer-oriented business strategies. Besides, environmental requirements are becoming indispensable regarding environmentally friendly product development. In our literature review, we examined that there is still a gap on explicit representation of sustainability requirements (e.g energy efficiency and resource efficiency) and stakeholders‘ concerns on the environmental sustainability of software products. In most cases, software development companies do not consider the explicit representation of sustainability requirements such as resource utilization, or customer‘s concerns on sustainability.

In this work, we focus on the relationship between customers, who require energy efficient software products, and software development companies, who offer these software products. The proposed introductory model aims to determine the demand and payoff functions by satisfying both sides with the quality level of the product and its price.

III. The Proposed Introductory Model

The goal of this introductory model is to find the equilibrium to meet the stakeholders needs and required quality attributes, while taking into account the environmental sustainability of the software product. This model is based on the theory of interactive decision-making model. It provides general mathematical techniques for analyzing situations in which two or more individuals make decisions that will influence one another‘s welfare [6]. These situations are referred to interactive decision processes. It consists of a collection of models. A model is an abstraction that we use to understand our observations and experiences [28]. In this section, the concepts and the elements of the proposed model are briefly presented.

The decision-making entities are considered as the individuals of the interactive decision process. In this model, we assume that the individuals are software development company and a customer and they are rational and intelligent [6]. The strategy of the one individual is a complete contingent plan of action for whatever situation might arise. The payoffs for an individual should capture everything in the outcomes that the individual cares. In many cases, the payoff function is represented by a utility function, which assigns a quantifiable value to each possible outcome, with higher utilities representing the more desirable outcomes. In an interactive decision-making model, it is assumed that each individual knows which strategy s/he prefers or which strategies are equally desirable for her/him.

TABLE I
NOTATIONS FOR THE MODEL

Symbol	Description
Ψ	Demand
b_1, b_2	Price and quality sensitivities of the demand, respectively
a	Fixed demand
d	Software product price, when *Q=0*, i.e. min product price when *Q=0*
P	Software product price
Q	Software product quality
U	Customers utility function
Π	Software development companys utility function from product sells
DC	Software product development cost
FC	Software product development, fixed cost
e	Software product quality sensitivity of price
w_i	Level of importance weight of each software quality attribute
A_i	Quality attributes $(i = 1, \ldots, n)$
μ	Utility sensitivity of the user
C	Constant

A. The Basic Model and its Components

We consider a differentiated market in which customers differ in their willingness to pay for the environmental quality of the software product. Each customer has a reservation price and preferences over the product‘s quality attributes. In such a market, the software development company‘s objective is both to determine the optimal price for its product and the optimal environmental quality attributes‘ level with respect to customers‘ requirements. The software development industry is generally concentrated. The up-front cost of developing software products is prohibitively high. Presumably, product design is mainly achieved via R&D-related expenditures (fixed costs) with little to no increase in marginal cost. In our model, the interacting individuals are software development companies. They build a strategic interaction with their customers. In the model, it is assumed that there are interdependencies among quality attributes and environmental attributes. All the necessary notations are given in I.

The Software Development Company

Similar to the model presented at [12], the software development company wants to maximize its own profit. The decision variables of the companys utility function are product price (P) and quality $Q(A_i)$. $Q(A_i, i = 1, \ldots, n)$ is assumed to be uniformly distributed on $[0, 1]$. We assume the existence of a normalized upper bound for quality, 1. The demand that the company faces is represented with $\Psi(P, Q)$. We follow a common assumption and define a linear demand function:

$$\Pi = \Psi(P, Q)P - DC, \tag{1}$$

where the development cost is defined as $DC = FC + e.Q(A_i)$. The parameter *FC* is the fixed cost tied to research and development and the design of the product. This fixed cost is assumed to be independent from the quality attributes‘ level. Total product quality is represented by the quality attributes *(A_i)* and related importance weights *(w_i)*. The product quality is calculated by multiplying the quality attributes level with corresponding importance weight:

$$Q = \sum wiA_i, \tag{2}$$

Then, the companys demand function is:

$$\Psi(P, Q) = a - b_1 P + b_2 Q(A_i), \tag{3}$$

with

$$P = d + eQ(A_i), \tag{4}$$

where *d>0 and e>0.*

When we substitute the price with quality attributes, then the demand is:

$$\Psi(P, Q) = (a - b_1 d) + (b_2 - b_1 e)Q(A_i), \tag{5}$$

and accordingly

$$\Pi(P, Q) = [(a - b_1 d) + (b_2 - b_1 e)Q(A_i)]P - FC + eQ(A_i)P, \tag{6}$$

The Customer

Let us now describe the customers utility. A customers utility depends on the price and quality attributes levels of the software product s/he is using. In this model, the customer is assumed to require maximizing the environmental quality of the software product at minimum expense. A customer accepts an offer, if the price asked is reasonable and the quality level is satisfying. Hence, we can introduce an acceptance probability *A (U, P)*, where *U* is the utility of the customer and *P* is the associated price of the software. The acceptance probability should be an increasing function of *U* for a fixed *P*, while decreasing function in *P* for fixed *U*:

$$A(U, P) = 1 - e^{-1CU^{\mu}Q^{(}b_2 P - b_1)} \tag{7}$$

The acceptance probability function can be differentiated among customers through the above parameters. The objective is to maximize the acceptance probability of each customer. In our software product, the energy efficiency and resource efficiency are the quality attributes in which the companys sustainability effort can be observed. Constant, *C*, which can be used to control the rate of feature perturbation, is set to 1 by default.

IV. A GREEN SOFTWARE PRODUCT SCHEME

In this section, we introduce a green software scheme based on the characteristics of the environmental sustainability requirements of the customers. A desired level of quality for software may be achieved by defining appropriate quality characteristics, taking into account the purpose of usage of the software product. Developing green software necessitates identification of the traditional quality attributes as well as and environmental attributes. The software development companies can show their effort to support sustainability by using these attributes and metrics.

A. *Selection of the Quality Attributes*

Software product quality attributes were selected from the ISO/IEC 25000 (SQuaRE) [16] series. ISO/IEC 25010 is a part of Square series that is composed of a quality in use and a product quality model. The criteria defined by model are relevant to all software product and computer systems. Considering the scope of our study, we adopted ISO/IEC 25010 product quality model that categorizes product quality properties into eight characteristics: functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability. While functional suitability, reliability, performance efficiency, usability, security and compatibility are defined as internal quality characteristics; maintainability and portability are external characteristics. In their survey study Akinli Kocak et al. [19] showed that security and compatibility has negligible impact of the environmentally sustainable software product quality. Therefore, in this work, we only adopted functional suitability, performance efficiency, reliability and usability as quality attributes. As for the environmental attributes, we adopted energy efficiency and resource efficiency from Akinli Kocak et al. [19], [18]. Environmental sustainability aims at improving human welfare while protecting natural resources. If the software product is considered, this dimension aims at addressing environmental requirements. When designing software intensive systems, immediate features and effects, longer-running, aggregate and cumulative impact of these systems have to be considered [4]. However, for simplicity we considered only energy and resource efficiency as environmental attributes which have direct effects (first order effects [14]) on the environmental sustainability. Computing resources (memory, processing, network bandwidth, and storage) are the principal sources of consumption within the software system. Given a monitoring of energy consumption over certain period, energy efficient resource usage possibilities may be spotted and subsequently applied. For this reason, energy efficiency and resource efficiency are chosen as characteristic of environmental quality of the product. Table II summarizes all the selected attributes of software quality.

B. *Determining the Importance Weights (wi)*

The quality attributes of software products may impact costumer‘s perceived value with various levels of intensity. We classified software products by characterizing them as high-intensity (a high), medium-intensity (a medium) and low-intensity (a low). High-intensity software products contain attributes that have biggest contribution on sustainability and are critical to achieve the purpose of the software. For the level of intensity, we use 1 to 5 scale [26]. In that scale, the high product intensity (scale: 1-2) means that this product will be accepted (as accept). Similarly, the medium product intensity (scale: 3-4) means that it will be reviewed (review) and the low product (scale: 5) means that it will be rejected. These levels of quality attributes *(A_i)* are used when calculating the total quality of the product that is given in the Equation 2.

General constraints of the attributes is $0 \leq A_i \leq 1.0$ *($i = 1, \ldots, n$)*. Table III shows the adopted attributes, their related metrics and level of intensity. In order to obtain the levels of quality, we widely use the Pareto principle often referred to as the 20-80 rule [33]. In the literature, this rule is widely used (e.g., [13]). For the energy efficiency, we adopted Intels holistic approach for energy star ratings across the broader range of products [21]. The energy efficiency of a computer with 100% workload has given as 82-85%. In order to use the same scale, we normalized them into [0-1] interval. We also used the power consumption measurements of a database software obtained by Akinli Kocak et al. [18], [20].

The importance weights of the quality attributes are also fundamental to set the priority, both for the customer and the company. Prioritizing is a process of managing the relative importance and determination of different requirements within the limited resources. Quality attributes weights and environmental attributes weights of software product can be obtained from evaluation of the criteria using multi-criteria decision making methodologies. In one of our works [18], we have used a well-known multi- criteria decision-making approach: Analytical Network Process (ANP) [32] to calculated the relative weights *(wi)* of the quality attributes.

V. Conclusion and Future Work

Many disciplines have been facing challenges in how to sustain economic, social and ecological systems. The design of environmentally sustainable software product is a challenging task, when it is compared to develop traditional software product, since an environmentally sustainable software has different quality attributes (especially in terms of quality requirements). In this work, we introduce a software classification scheme using standard software quality requirements and introducing environmental attributes. Accordingly, we propose a way of determining demand in terms of price and quality, and utility of a software development company. We derive customer‘s and development company‘s payoffs. As a continuation of this work, the proposed simple demand functions need to be elaborated by quantifying the sensitivity coefficients. These demand functions may be integrated into the given green software scheme to build a green decision support tool. Our contribution will serve as a normative guide to software development companies for the design of green product with handling environmental sustainability as a quality objective. As next step, we plan to examine the theoretical introductory model with simulation and find the equilibrium to meet the both development company and customer needs within the necessary quality attributes.

Acknowledgment

This research has been financially supported by the Galatasaray University Research Fund, project number: 19.401.005.

TABLE II
SOFTWARE QUALITY AND ENVIRONMENTAL SUSTAINABILITY ATTRIBUTES

(Q)	**Software Quality**
Q1: Functional Suitability	The product fits the functional requirements of the user and customer.
Q2: Performance Efficiency	How well the product responds to user requests and how efficient it is at execution time.
Q3: Reliability	The product produces failures and hence may not be available.
Q4: Usability	How easily the system can be used.
(E)	**Environmental Sustainability**
E5: Energy Efficiency	The level of energy performance of the software and the amount of energy resources used, under stated conditions.
E6: Resource Efficiency	How efficiently the resources used by the product when performing its functions and/or serving useful workload.

TABLE III
QUALITY ATTRIBUTES, METRICS AND LEVEL OF INTENSITY

Attribute	**Related Metric**	**Levels of Intensity**
Functional suitability (A_1)	Functional completeness	Low, if $A_1 \leq 0.20$ Medium, if 0.21<A_1<0.80 High, if A_1≥0.80
Performance efficiency (A_2)	Response time	Low, if A_2≤0.30 Medium, if 0.30<A_2<0.70 High, if A_2≥0.70
Reliability (A_3)	Fault tolerance	Low, if $A_3 \geq 0.80$ Medium, if 0.20<A_3<0.80 High, if $A_3 \leq 0.20$
Usability (A_4)	Accessibility	Low, if $A_4 \leq 0.20$ Medium, if 0.20<A_4<0.70 High, if $A_4 \geq 0.70$
Energy efficiency (A_5)	Power consumption	Low, if $A_5 \geq 0.80$ Medium, if 0.20<A_5<0.80 High, if $A_5 \leq 0.20$
Resource efficiency (A_6)	Sum of CPU, I/O and memory consumption	Low, if $A_6 \geq 0.80$ Medium, if 0.20<A_6 <0.80 High, if $A_6 \leq 0.20$

REFERENCES

[1] Oxford English Dictionary Online, 2nd edition. http://www.oed.com/, July 2003.

[2] M. Al Hinai and R. Chitchyan. Engineering requirements for social sustainability. 2016.

[3] P. Avgeriou, M. Stal, and R. Hilliard. Architecture sustainability [guest editors' introduction]. *IEEE Software*, 30(6):40–44, 2013.

[4] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters. Requirements: The key to sustainability. *IEEE Software*, 33(1):56–65, 2015.

[5] C. Becker, R. Chitchyan, L. Duboc, S Easterbrook, M. Mahaux, B. Penzenstadler, G. R. Navas, C. Salinesi, N. Seyff, C. C. Venters, C. Calero, S. A. Koçak, and S. Betz. The Karlskrona manifesto for sustainability design. *CoRR*, abs/1410.6968, 2014.

[6] R. B.Myerson. *Game Theory: Analysis of Conflict*. Harward University Press, 1997.

[7] Y. Cai, Lu. Xiao, R. Kazman, R. Mo, and Q. Feng. Design rule spaces: A new model for representing and analyzing software architecture. *IEEE Transactions on Software Engineering*, 2018.

[8] C. Calero, M. F. Bertoa, and M. A.Moraga. Sustainability and quality: Icing on the cake. In *Second Int. Workshop on RE for Sustainable Systems (RE4SuSy)*, 2013.

[9] N. Condori-Fernandez and P. Lago. Characterizing the contribution of quality requirements to software sustainability. *Journal of Systems and Software*, 137:289–305, 2018.

[10] N. Condori Fernandez and P. Lago. *A Sustainability-quality Model*. VU Technical Report, 11 2018. Version 1.0.

[11] L. Fang, K. W. Hipel, and D. M. Kilgour. *Interactive decision making: the graph model for conflict resolution*, volume 11. John Wiley & Sons, 1993.

[12] C. Faugère and G. K. Tayi. Designing free software samples: a game theoretic approach. *Information Technology and Management*, 8(4):263–278, 2007.

[13] N. E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software engineering*, 26(8):797–814, 2000.

[14] L. Hilty and B. Aebischer. Ict for sustainability: An emerging research field. In *ICT Innovations for Sustainability*, pages 3–36. Springer, 2015.

[15] A. Hindle. Green mining: a methodology of relating software change and configuration to power consumption. *Empirical Software Engineering*, 20(2):374–409, 2015.

[16] ISO/IEC25010:2011. Systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models, 2011.

[17] E. Kern, M. Dick, S. Naumann, A. Guldner, and T. Johann. Green software and green software engineering–definitions, measurements, and quality aspects. *Hilty et al.(2013)*, pages 87–94, 2013.

[18] S. A. Koçak, G. I. Alptekin, and A. Başar Bener Başar. Evaluation of software product quality attributes and environmental attributes using anp decision framework. In *Proceedings of the Third International Workshop on Requirement Engineering for Sustainable Systems (pp. pp. 37-44). Karlskrona: Central Europe Workshop Proceedings*, 2014.

[19] S. A. Koçak, G. I. Alptekin, and A. Başar Bener Başar. Integrating environmental sustainability in software product quality. In *RE4SuSy@ RE*, pages 17–24, 2015.

[20] S. A. Koçak, G. I. Alptekin, A. Miranskyy, A. Başar Bener Başar, and E. Cialini. An empirical evaluation of database software features on energy consumption. In *ICT4S*, pages 1–19, 2018.

[21] R. Kolappan, H. Wong, M. Duggiralam, K. Kaplan, E. Haines, and T. Bolioli. Energy star version 5.0 system implementation, technical report, intel, (2009). http://www.energystar.gov/ia/partners/product_specs/program_reqs/Computers_Intel_Whitepaper_Spec5.pdf.

[22] P. Lago. A software sustainability assessment method. Technical report, Technical Report, figshare, 2014. http: //goo.gl/HuY6tf, 2016.

[23] P. Lago. Architecture design decision maps for software sustainability. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Society*, pages 61–64. IEEE Press, 2019.

[24] P. Lago, S. A. Koçak, I. Crnkovic, and B. Penzenstadler. Framing sustainability as a property of software quality. *Commun. ACM*, 58(10):70–78, 2015.

[25] P. Lago and B. Penzenstadler. Reality check for software engineering

for sustainabilitypragmatism required. *Journal of Software: Evolution and process*, 29(2):e1856, 2017.

[26] R. Likert. A method of constructing an attitude scale. *Scaling: A sourcebook for behavioural scientists*, pages 233–243, 1974.

[27] F. A. Moghaddam, G. Procaccianti, G. A. Lewis, and P. Lago. Empirical validation of cyber-foraging architectural tactics for surrogate provisioning. *Journal of Systems and Software*, 138:37–51, 2018.

[28] M. J. Osborne et al. *An introduction to game theory*. Oxford university press New York, 2004.

[29] B. Penzenstadler et al. Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century. *IEEE Software*, 31(3):40–47, May 2014.

[30] E. R. Poort and H. van Vliet. Architecting as a risk-and cost management discipline. In *2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, pages 2–11. IEEE, 2011.

[31] G. Procaccianti, H. Fernández, and P. Lago. Empirical evaluation of two best practices for energy-efficient software development. *Journal of Systems and Software*, 117:185–198, 2016.

[32] T. L. Saaty. *Decision making with dependence and feedback: The analytic network process*, volume 4922. RWS publications Pittsburgh, 1996.

[33] G. G. Schulmeyer and J. I. McManus. *Handbook of software quality assurance*. Van Nostrand Reinhold Co., 1992.

[34] C. C. Venters, R. Capilla, S. Betz, B. Penzenstadler, T. Crick, S. Crouch, E. Y. Nakagawa, C. Becker, and C. Carrillo. Software sustainability: Research and practice from a software architecture viewpoint. *Journal of Systems and Software*, 138:174–188, 2018.

[35] C. C. Venters, L. Lau, M. Griffithsa, V. Holmes, R. Ward, C. Jay, C. Dibsdale, and J. Xu. The Blind Men and the Elephant: Towards an Empirical Evaluation Framework for Software Sustainability. *Journal of Open Research Software*, 2(1), 2014.

[36] C. C. Venters, N. Seyff, C. Becker, S. Betz, R. Chitchyan, L. Duboc, D. McIntyre, and B. Penzenstadler. Characterising sustainability requirements: A new species red herring or just an odd fish? In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track (ICSE-SEIS)*, pages 3–12. IEEE, 2017.

[37] J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb. Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management science*, 54(7):1336–1349, 2008.