Mind the Chasm: A FishEye Lens View of Sustainable Software Engineering in UK Higher Education

Madeleine L. Gibson
University of Huddersfield
Huddersfield, UK
U1364186@unimail.hud.ac.uk

Colin C. Venters
University of Huddersfield
Huddersfield, UK
c.venters@hud.ac.uk

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

Stefanie Betz
Karlsruhe Insitute of Technology
Karlsruhe, Germany
stefanie.betz@kit.edu

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

Maria Palacin-Silva
LUT
Lappeenranta, Finland
maria.palacin.silva@lut.fi

Birgit Penzenstadler
CSU Long Beach
Long Beach, CA, USA
birgit.penzenstadler@csulb.edu

Norbert Seyff
FHNW and University of Zurich
Windisch, Switzerland
norbert.seyff@fhnw.ch

***Abstract*—Requirements that express the needs of all stakeholders and cover the key aspects of a software system (such as those addressing sustainability) are critical to the system's successful development and adoption. For practitioners who want to develop sustainable software-intensive systems, it is also argued that software requirements are the key leverage point.**

But what do software developers know about broader sustainability and sustainability requirements in particular? As part of the University of Huddersfield's *'Student as a Researcher'* initiative[1], this paper provides a FishEye[2] Lens View on how novice software developers who design and develop software systems at software companies relate to the notions of sustainability and sustainability requirements. The study has found that although sustainability is valued highly by the novice software developers, the concept of sustainability and sustainability requirements are not fully understood. This lack of knowledge in the software industry and organizations must be addressed to enable delivery of truly sustainability enabling software systems.

***Index Terms*—Karlskrona manifesto, requirements engineering, software engineering, software requirements, sustainability, sustainability design, sustainability requirements, sustainable software**

I. Introduction

Post-industrial societies are highly dependent on complex software systems that underlie almost every aspect of daily living from communication, entertainment, energy, commerce, transportation, finance, governance, health care, as well as defense and security [2], [3], [4]. In addition, software also now plays a critical role in the advancement of knowledge with a paradigm shift in research towards large-scale computational science and engineering, and data intensive science [5], [6]. Software's increasing importance in the field of research, has led for calls for it to be classified as a first-class, experimental scientific instrument [7] based on the maxim, *if your software is incorrect, so will your science* [8]. However, it is suggested that in terms of structure, content, and functionality, software systems are the most complex artifacts ever created [9]. As such, software designers are responsible for the long-term consequences of their designs [10].

The vision of sustainable software engineering is based on the idea of delivering software systems, which are designed and developed in a way that the consideration of the different sustainability dimensions and orders of effects are reflected in the system. Although there is no clear definition of the term 'sustainability requirement' [11], the bespoke reflection and discussion is often based on requirements that align with specific dimensions of sustainability. This highlights the importance of requirements engineering as a central aspect of sustainable software engineering. Moreover, it is argued that requirements are the key to sustainability, as it is during the requirement engineering phase when the foundations of a systems are laid [12].

The goal of the work presented in this paper is to better understand the perception of sustainable software engineering among UK students enrolled on computing degree programmes and young software developers in industry, which we will collectively refer to as novice software engineers in the reminder of this paper. More specifically, the paper aims to find out what these novice developers think about sustainability requirements and what they may know about it. The remainder of this paper is structured as follows: In section II, we describe relevant background research, while section III gives an overview of the methodology used for

[1]Student as a Researcher is 'a pedagogic approach to supporting students in their engagement with undergraduate research within and/or beyond the formal curriculum with the aim of furthering their own knowledge and understanding, and in some cases contributing to, the broader knowledge base of their discipline. The term is used to describe a pedagogic approach, rather than the students themselves' [1].

[2]A Fisheye lens is a special type of lens that shows a distorted view of the world.

the study. In section IV, we present the results of this work, with several relevant issues discussed in section IV. The paper concludes in Section V, with recommendations for next steps toward a more comprehensive and consistent perspective on sustainability requirements and their relationship to the field of software engineering, and the role of education in facilitating a programme of sustainable software engineering.

II. Background

A. Sustainability and Software

Derived from the Latin word *sustinere*, the Oxford English Dictionary [13] defines sustainability as *'the quality of being sustained'*, where sustained can be defined as *'capable of being endured'* and *'capable of being maintained'*. This suggests that *longevity* and the ability to *maintain* are key factors at the heart of understanding sustainability. As a part of the concept of *sustainable development*, the Brundtland commission [14], defined the term as *'meeting the needs of the present without compromising the ability of future generations to meet their own needs'*. The word *'need'* is central to this definition and includes a dimension of *'time'*, present and future.

In the last decade the concept of sustainability has emerged as a topic of interest in a range of different areas in the field of computing including software engineering (SE) [15] and requirements engineering (RE) [11]. In relation to software and software systems, there are primarily two distinct viewpoints that make up the topic area around software and sustainability: sustainable software and software engineering for sustainability (SE4S), where the former is concerned with the principles, practices, and process that contribute to software endurance i.e. technical sustainability [16], and the latter focuses on software systems to support one or more dimensions of sustainability, concerning issues outside the software systems itself [17]. The Karlskrona Manifesto [10] recognizes both viewpoints on sustainability as an emerging concern of central relevance and suggests to view sustainability as a construct across the five dimensions of sustainability: environmental, economic, individual, social and technical [12], [18]. These dimensions are defined as follows:

- The economic dimension refers to assets, capital, and added value including for example profitability and capital investment, etc;
- The environmental dimension refers to the long term effects of human activities on natural ecosystems and resources;
- The individual dimension refers to the individual well-being including for example freedom, self-respect, education etc;
- The social dimension refers to the interaction between individuals and the societal constructs based on this interaction and constituting a society including for example trust, communication, employment, democracy, etc;
- The technical dimension refers to the concepts of longevity of man-made systems, and infrastructure and their adequate evolution.

There exist interdependencies between these dimensions including trade-offs and synergies. Moreover, the impact on sustainability in these five dimensions manifests in three different orders of effect [12], [19], defined as follows:

- First order effects appear when software systems are built and used for their primary purpose, e.g. the work time used for implementation of Youtube;
- Second order effects appear when the use of the system over some time induces new types of behavior or expectations for the previous system, e.g. the energy waste due to letting Youtube videos play in the background, while only listening to the music;
- Finally, third order effects appear due to a large-scale, longer term use of the system, e.g. as most people use YouTube for music, professional music record producers and sellers gradually disappear as domain actors (which is a clear detrimental effect on the economic sustainability of these businesses), making self-production and free availability of music on Youtube a new social norm (which democratises the music market for the musicians, and increases the variety of available music for consumers, positively affecting individual and social sustainability).

While there have been a number of contributions to formalize a definition of software sustainability [20], [15], [16], [18], [21], consensus on what sustainability means in the field of software and requirements engineering is still emerging [22], [23]. The term is increasingly described in the literature as a first-class, non-functional requirement or software quality [24], [25], [11]. However, research is required to confirm or refute this position.

Furthermore, Ramsey [26] argues that the fundamental problem with definitions of sustainability is the definitional approach itself and not just with any particular definition. It is suggested that definitions do not lead to clarity about the meaning of sustainability because the meaning of the component terms, as well as the meaning of the general concept, presupposes the existence of a set of social structures within which the activity of defining makes sense. As such, any attempt to impose interpretations without recognizing the necessity and importance of those structures are doomed to failure.

As a result, the concept of sustainability requires context (such as that proposed by Tainter [27]), [social] structure [26], and the simultaneous consideration of several interrelated dimensions of sustainability [12]. In addition, rather than seeking broad conformity of definitions, the aim should be to clarify how the terms are used by different communities in order to have a shared understanding [28].

B. Sustainability Studies in Software Engineering

A number of recent studies have been conducted to understand how sustainability is perceived in the practice of software engineering and how sustainability can become an inherent part of software engineering practice.

Chitchyan et. al., [29] explored requirements engineering practitioners' perceptions and attitudes towards sustainability through interviews. The results revealed that software practitioners tend to have a narrow understanding of the concept of sustainability with a focus on environmental and economic sustainability. Their organizations also showed limited awareness of the potential opportunities and benefits for engineering sustainability in or through software. Notably, several interviewees explicitly considered sustainability as a separate field from that of software engineering.

Manotas et. al., [30] investigated what software practitioners from ABB, Google, IBM, and Microsoft thought about energy usage in relation to requirements, design, construction, testing, and maintenance of their software. Their results suggest that software engineering practitioners are concerned about energy when they build applications; however, it was suggested that they are not as successful as they could be because they lack the necessary information and support infrastructure. In addition, the results also highlighted that energy-usage is often stated in terms other than energy usage being more often desires rather than specific targets, which focus on *idle time* and are difficult to specify directly.

Kasurinen, Palacin-Silva and Vanhala [31] investigated the views of professional game developers on sustainability and Green IT. The results of their study highlighted that Game development organizations are not likely to consider eco-impact factors such as code reusability, energy-efficient programing, social awareness impact, reuse or repurposing old hardware components or marketing materials or support to legacy systems in their daily work. If anything at all, code reusability was the most regarded factor due to efficiency reasons. This suggests that the game developers do not recognize what sustainability means in their industry. Their organizations, as a whole, do not practice sustainability-conducive activities either. The authors attributed this to the lack of a common definition of sustainability in the field as a way of understanding what constituted eco-impact factors; a view supported by Penzenstadler [18]. In addition, it was also suggested that a *sustainability body of knowledge for software engineering* could provide specific guidance with regards to addressing sustainability aspects from a software development perspective.

Groher and Weinreich [32] investigated how sustainability is perceived by software engineering professionals in order to understand how sustainability is currently dealt with in software development projects in practice. The results of the study revealed that practitioners primarily relate sustainability to technical sustainability with a strong focus on maintainability and extensibility of the software systems. Economic sustainability was an indirect consideration related to influencing factors such as the market and customers. In contrast to Chitchyan et. al., [29], environmental sustainability was not mentioned at all by the study participants.

Chitchyan, Groher and Noppen [33] investigated what sustainability means to the field of software engineering within the context of Software Product Line Engineering (SPLE) through analysis of 11 case studies. The overall results suggest that technical and economic sustainability are the primary focus in current SPLE practice, with social sustainability issues, where they relate to organisations, also addressed. However, environmental and individual sustainability concerns were less prominent.

Overall, these studies suggest that practitioners have a narrow perception of sustainability and that sustainability is not a major concern in their current projects. However, more elaborate concepts and a more stable theoretical background regarding the body of knowledge about sustainable software engineering is emerging, which might also influence the perception of suitability in software engineering education. Therefore, we could expect to see a higher degree of awareness regarding sustainability issue when it comes to undergraduate and graduate students, but there is no related work exploring this issue. The research presented in this paper contributes to filling this void.

C. Emergence of Sustainability Requirements

As the concept of sustainability has started to be discussed in software engineering, so has the concept of sustainability requirements. Becker et. al., [12] argue that the critical role that software plays in society demands a paradigm shift in the mindset of software engineering and that the focus of this shift begins in requirements engineering. A number of approaches have been proposed for sustainability requirements elicitation [34], [35], codification [36], modeling [37], [38], and management [18]. For example Penzenstadler and Femmer [18] proposed a goal modelling based approach to capture sustainability requirements using a generic reference model. Betz [39] presents a business process based view to modelling sustainability requirements. While Makropoulos [40] analyses the specific case of sustainable urban water management, proposing a general model of sustainability that needs to be contextualized to assess sustainability. As far as we can ascertain, there exists no formal definition of the term sustainability requirement. In most instances, a sustainability requirement is defined in terms of a non-functional requirement or software quality and aligned with one or more of the dimensions of sustainability [37], [34], [36], [41] based on characteristics defined in ISO/IEC 25010 [42]: functional suitability, reliability, performance, efficiency, usability, security, compatibility, maintainability, and portability. Roher and Richardson [35] defined a sustainability requirements as *'requirements that may be used to specify system behavior (e.g. requirements that will reduce a system's energy consumption) as well as to influence the users' behavior (e.g. the system incentivizes sustainable actions)'*. In contrast, Huber, Hilty, and Glinz [43] in their investigation into the requirements of a decision support system (DSS) suggested that a sustainability requirement is a *'requirement for a sustainable software system which concerns sustainability'*. They argue that sustainability requirements cannot simply be reduced to quality aspects and should be related to positive effects suggesting that when positive enabling effects are taken into consideration the requirements

for a software system change under the lens of sustainable development. While the latter definition is overly simplistic it implicitly suggests that requirements can be viewed through a multi-perspective *lens of sustainability*. The former definition essentially points towards a subset of functional requirements and towards user influence. As a result, neither definition provides any insight into how sustainability requirements differ from existing taxonomic classifications of requirements [44], [45]. However, Venters et. al. [11] revealed that the term *sustainability requirement* is used ambiguously and contains significant variations across the different application and scientific discipline domains. Their study suggests, that the dimensions of sustainability and its time effects could help to see which requirements pertain to sustainability of the software system and its wider impact.

III. Methodology

The principal aim of this study is to explore the knowledge of novice software developers with respect to sustainability, sustainability requirements, and their relationship to software engineering principles and practices. Its objective is two-folded: to learn about the perspectives and ideas of novice software developers in relation to sustainability and sustainability requirements, and to highlight whether sustainability is taught within the software engineering education. Therefore, the study considered the following research questions:

- RQ1 What are the opinions and expectations of novice software developers on sustainability and sustainability requirements?
- RQ2 What do novice software developers know about sustainability (within software engineering) when developing software systems?

A. Study Design

A qualitative research approach was chosen to answer the above research questions using semi-structured interviews, which included open and closed questions to elicit the data. This approach allows the classification of the data into concepts and themes in order to explain a particular phenomena and to gain a deeper understanding of the underlying reasons, opinions, and motivations [46], [47].

1) Planning: At the planning stage, the interview questions were designed collaboratively by all authors. The study was piloted with one interviewee to validate the clarity of questions and the interview structure resulting in no major changes.

2) Interviewees: The subjects were selected from current and former students from the School of Computing and Engineering at the University of Huddersfield, using the contact network of the first two authors of this paper. The criteria for selection was as follows:

- Current students were required to be enrolled on a Bachelors of Science (BSc.) or Masters (MEng, MSc.) degree programme in Computing, Computer Science, Games Design, Game Programming, Information Systems or Software Engineering;
- Former students were required to hold a Bachelors of Science (BSc.) or Masters (MEng, MSc.) degree in one of the above listed areas;
- Employed in the software industry for a minimum of six months;
- Worked or are currently working in the software industry as part of their industrial placement year[3];
- Elicited software requirements as part of their respective projects.

The study included eight participants of which, seven participants were male and one was female. Four are currently working in the software industry as part of their industrial placement year. Three were former students currently employed in the software industry. One was enrolled on the MEng. degree programme in Software Engineering. Their age range was between 20-29 years and they were associated with one of the following institutions:

- LNT Software[4]: a small to medium sized Enterprise (SME) software production company dedicated to the design and development of a range of management software systems used in care homes across the UK;
- School of Computing and Engineering, University of Huddersfield[5]: a public university located in Huddersfield, West Yorkshire, UK;
- Canalside Studios Game Design[6]: an independent game studio, within the School of Computing and Engineering at the University of Huddersfield.

3) Data Collection: The data collection was undertaken through in-person, semi-structured interview, which lasted approximately thirty minutes and covered four key areas:

1) Personal background, including demographics, software industry experience, and education level;
2) Sustainability, including their understanding of the term and concrete actions towards sustainability in both private and professional lives;
3) Software engineering principles and practices, normally applied in their work and their potential to contemplate sustainability;
4) Sustainability requirements, covering their understanding of functional and non-functional requirements, as well as sustainability requirements.

4) Data Analysis: The qualitative content analysis method [48] was used to extract views and perceptions on sustainability, sustainability requirements, and software engineering principles and practice from these interview transcripts. The interviews were fully transcribed and analyzed using Saturate [49]; a qualitative data analysis tool. A codebook was created by the first two authors of the paper in order for the data to be classified and trends observed. The initial set of codes were

[3]All undergraduate programs in the School of Computing and Engineering at the University of Huddersfield include an optional year in industry: https://www.hud.ac.uk/ce/placements/

[4]LNT Software: www.lntsoftware.com

[5]School of Computing and Engineering, University of Huddersfield: https://www.hud.ac.uk/ce/

[6]Canalside Studios Game Design: http://www.canalsidestudios.com

created by the first coder and was updated with each following coding activity. The initial codebook, as well as the updates, were discussed and agreed upon by all co-authors of this paper. The first two authors read each of the interviews and coded the text with conceptual categories relevant to sustainability perceptions, as well as peer-reviewed each other's work. The interview guide and the codebook are available at [50].

B. Limitations of the Study

As is normally the case with qualitative research, the results cannot be generalized to the wider population of software developers. This is both due to the limitations of the approach and the limited number of subjects [51]. In addition, this study focused solely on novice software developers, as a means to also evaluate the coverage of sustainability within software engineering education. As a result, participants may not have had enough industry experience to provide valuable insights into sustainability with software engineering and sustainability requirements. Furthermore, the fact that all participants attended the same university, also limits the conclusions to that particular institution. However, the results corroborate with the findings from other similar studies [29], [31], [32].

IV. Study Findings

The main findings of this qualitative study fall into three categories, which are discussed in the following sections. Each of the individual interviewees is referenced by a fictitious name to ensure anonymity.

A. Sustainability Findings

The first set of questions were intended to uncover the participants' understanding of and commitment to sustainability.

What is your understanding of the concept of Sustainability? All interviewees related sustainability to the concept of *endurance over time*. For example, Scott stated that sustainability is *"something that can be sustained effectively over a long or short period of time"*. In addition, two interviewees suggested that sustainability could also be related to technical aspect of sustainability including software metrics and hardware. Thomas stated that sustainability was also related to the *"metrics used to quantify how sustainable a piece of software is"* both through the *"development phase and after its hand-off to a client"*. Jamie suggested that sustainability was related to the longevity of resources stating that there are issues with *"hardware sustainability"* particularly with advances in hardware driving the game industry. Ryan said that the concept should be considered in relation to the environment in a more direct manner, stating that sustainability is also related to *"using less or an equal amount of resources that are being produced at the same time"*.

Have you ever considered the following sustainability dimensions: Environmental, Economic, Individual, Social or Technical? When asked whether they had ever considered sustainability in relation to the five sustainability dimensions, those participants who understood the differentiation of the terms in relation to sustainability generally had a good understanding of the *environmental*, *economic* and *technical* dimensions of sustainability but lacked any real understanding with regard to the individual and social dimensions; these were unheard of by seven of the participants. The other participant stated that they had not considered sustainability in relation to the five dimensions. This is in itself surprising especially with regards to the prominence of environmental sustainability in post-industrial societies particularly around recycling and energy conservation.

Four interviewees (Scott, Claire, Steven and Thomas) stated that they had considered environmental sustainability, citing issues related to the impact of the environment including *global warming, deforestation, resource depletion and farming*, as well as ways to mitigate that impact in terms of moving towards *renewable energy* sources such as wind and tidal, and *recycling*. However, Steven also highlighted that environmental sustainability *"doesn't need to be natural, it could be artificial such as this office space"* emphasizing that the environments we work in also need to consider sustainability. In addition, Thomas related environmental sustainability to *hardware and server capacity* stating that "if the servers went down, the software's environment must also be sustainable", indirectly acknowledging the relevance of the wider notion of the systems dependability for sustainability.

Four interviewees (Steven, Thomas, Ryan, James) also stated that they had considered economical sustainability, three of whom referred to it as ensuring *business value, capital growth and investment, and financial operations*. For example, James suggested that in terms of economic sustainability *"I don't think they've made as much money as they have put into this [Canalside Studios], so really, they are paying for the students to get experience really"*.

Finally, four interviewees (Steven, Claire, Mark and Ryan) stated that they had considered technical sustainability, which covered both the software and hardware aspects of software systems. Steven stated that technical sustainability was *"related to the ability to build a software product that is built in a way so that it does not die"*. Similarly, Claire and Mark highlighted the importance of maintainability and extendability of the software systems. Claire emphasized the importance of *"the ability to maintain and evolve software systems over a prolonged period of time"*. Mark explained that *"I program in a way that's easily maintainable for people other than myself, I make short methods that are recognizable so we minimize the amount of lines we use"*. Ryan suggested that technical sustainability was related to effectively utilizing resources, explaining that *"technical sustainability is not using more resources than you have, such as computer power or man power"*.

What do you do for sustainability in your daily private life? The most common answers amongst participants were related to [environmental] sustainability. Ben, Jamie, Scott, Claire, Steven and Thomas articulated tasks such as *switching lights off, saving energy, and recycling*. However, Ben, Jamie and Scott also emphasized the importance of saving and

investment in relation to ensure economic stability thereby linking activities related to economic resource usage to [economic] sustainability. However, Mark stated, *"I don't think I do sustainable, I rarely recycle or take consideration of the environment"*.

What do you do for sustainability in daily work life? In contrast to how participants viewed sustainability in their private life, there was a strong emphasis by seven participants on [technical] sustainability specifically related to code maintainability and evolution associated with actions to ensure *code readability*, *code efficiency*, *coding standards*, as well as the importance of *refactoring legacy code* and *code reusability*. Mark highlighted the issues related to unsustainable practice stating that *"the current project I am recoding is full of memory leaks because it wasn't coded well"*. Ryan stated that *"the team make sure the code is sustainable so others can read it later on, change it and understand it"*, which suggested that this was an important task for the entire team. This was a view shared by the majority of participants. Claire also highlighted the importance of the role of project management in underpinning the process. Notably, the activities related to environmental sustainability previously mentioned for the context of private life (e.g., *switching lights off*, *saving energy*, and *recycling*) did not explicitly cross over into the interviewees' daily working practice.

B. Software Engineering Principles and Practice

The second set of questions investigated the subjects' knowledge with respect to software engineering principles and practices.

What are the main software engineering principles you consider when working on a project?

With respect to software engineering principles, the most common answer was related to improving *"code readbility"* including *coding style*, *formatting*, *naming*, *comments*, *coding conventions*, and *documentation*. James emphasized that it was critical that *"others within the team must be able to understand the code with minimal explanation"*. In addition, Thomas also mentioned the need to have *comprehensive testing* and utilise *requirements traceability*. The most in-depth answer was given by Steven, who referred to the ability to *"adapt to change"* and the *"agile principles"*. He also claimed that *"everything I apply within the project is to ensure sustainability, everything is considered for the long-term effects of the software"*. Two participants (Ben, Claire) were unsure of what software engineering principles were. Ben stated that *"as a Game Designer, I have never really considered software engineering principles"*. However, the responses highlight a general misperception between software engineering principles[7] such as GRASP[8], SOLID[9], YAGNI[10], DRY[11], KISS[12], and best software engineering practice.

What are the software engineering principles you apply that are related to sustainability in software systems? Similar to the responses to the previous question on software engineering principles, the most common answer was related to improving the overall readability of the code through a variety of best practice activities i.e. code structure, use of comments etc. However, Mark and Ryan emphasized that this was primarily to achieve the *"maintainability"* of the code base. Mark explained that *"I code in a maintainable way. I always code so if I wasn't here or available, other people would continue from me easy, without causing [memory] leaks"*. This suggests that participants consider that *maintainability* is a core activity in ensuring that the code is [technically] sustainable. However, three participants (Claire, Ben and Steven) could not suggest any software engineering principles specifically related to sustainability.

What software engineering practices do you consider when building software systems? With regard to software engineering practices, the most common answer was related to software development process with four participants (Steven, Claire, Thomas and Scott) stating that they employed *SCRUM* and *Agile* development processes' and practice. Claire stated that everything starts off with *"an 'amigo' meeting in which a small team discusses the software systems features, with tasks be delegated into sprints"*. Steven added that in addition to the core SCRUM approach, they regularly employ techniques such as *prototyping*, *code reviews*, *pair programming*, and *automated testing and deployment*. The majority of responses here are related to software engineering process.

What software engineering practices do you consider when building software systems?

When asked to relate sustainability with software engineering practices, half of the participants (James, Scott, Mark, Steven) fell back to the concepts of *efficiency* and *maintainability*. The remaining four participants (Ben, Ryan, Claire, Thomas) could not relate any of their current software engineering practice to sustainability. Thomas stated, *"I really don't know if I contemplate sustainability at all when considering practice"*.

As a general observation of the interview, even though all participants studied computing related subjects at a UK institution of Higher Education with a common core of software engineering courses, the majority of the them struggled to answer the questions related to the basic concepts of software engineering principles and practices, suggesting that these were not well understood by our volunteers. For instance, there is a general lack of clarity on the difference between software engineering principles and practice, suggesting that this differentiation may not have been covered in the Huddersfield's curriculum.

[7]principle, (n). a primary assumption forming the basis of a chain of reasoning [13]

[8]GRASP: General Responsibility Assignment Software Patterns

[9]SOLID: Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion

[10]YAGNI: You Ain't Gonna Need It! [52]

[11]DRY: Don't Repeat Yourself

[12]KISS: Keep It Simple, Stupid!

C. Sustainability Requirements

The last set of questions referred to the participants' understanding and ideas about requirements in general and sustainability requirements in particular.

What is your understanding of the term Functional Requirement (FR)? Surprisingly, participants' formal understanding of the term *functional requirement* deviated significantly from how it is generally understood within the field of software and requirements engineering, defined in standard software engineering textbooks [53] or SWEBOK [54]. Four participants (Ryan, Scott, Claire, Steven) gave reasonable definitions or examples of a *functional requirement*, such as *"it is what the users of the product need to be able to achieve [...] ultimately the users have some action that they need to achieve [...] I guess it could be related to the controls of the game. [...] a feature you expect to do something"*. However, four participants (Ben, James, Mark, Thomas) had no or an incorrect understanding of the term. For example, Mark stated, *"no idea, I'd just assume it's coding in a clean and clear way and following common practices"*. These results are unexpected, as all participants work in the software industry.

What is your understanding of the term Non- Functional Requirement (NFR)?

Similarly, participants' understanding of the term *non-functional requirement* also deviated significantly from the common view in the software/requirements engineering communities. Three participants (James, Scott, Steven) suggested that *non-functional requirements* were related to *"performance"* or *"reliability"*. However, four participants (Ryan, Ben, Mark, Thomas, Claire) had no or an incorrect interpretation of this term. Ben described it as *"something that is not compulsory, such as something that the project doesn't need to succeed, but something the user wants"*, ignoring that fact that software is unlikely to be successfully used if non-functional requirements are overlooked. Again, these results are somewhat surprising given respondents' previous work in the software industry.

What is your understanding of the term "sustainability requirements"? When asked about "sustainability requirements", interviewees gave answers which were consistent with their understanding of sustainability. Four participants (Mark, Scott, James, Ben) considered the notions of *"time"*, *"longevity"* or *"maintenance"*. For example, Mark suggested that it is related to *"making a system that will keep going over a long period of time* [...], followed by examples of coding standards (e.g. classes with less than 1000 lines, methods limited to 20 lines and a maximum of five variables within a method). Ryan suggested a relationship with *reuse* when describing a sustainability requirement: *"someone could review my code and easily reword/restructure it as well as read it and understand it easily"*. In general, explanations lacked precision and showed confusion over the concept. For example, James suggested that a sustainability requirement is *"a requirement that is sustainable over a prolonged period"*. Thomas said that it was about *"having a piece of software that is sustainable, rather than having requirements that just meets functionality demands"*. One interviewee was unable to give an answer.

Elicitation, modelling, and evaluation of "sustainability requirements". The last three questions of the interview focused on elicitation, modelling and the evaluation of *sustainability requirements*[13].

However, while the questions generated a small amount of discussion, participants either lacked confidence or were unable to answer these questions with the majority opting for a fall back position of *"I don't think I could give you an answer to that"*. Five participants (Ryan, Jamie, Scott, Mark, Steven and Thomas) were unsure how to formally analyze requirements per se, which may in some instance be related to their working environment where *"the specifications and requirements are already drawn up for us"* suggesting that this task is the responsibility of another team member. Most telling was that the majority of the participants were unable to relate any existing software and requirements engineering practice to the elicitation, modelling or the evaluation of *sustainability requirements*. However, with regards to modelling a sustainability requirement, Thomas suggested , *"I would use basic UML modelling so I could visualize the problem or requirement...I don't really have any examples"*. Claire suggested employing *"user stories then breaking that down into scenarios"*; she believed that the method as a whole could be used to enable sustainability within software development.

V. Discussion

A. Sustainability

All interviewees had a very broad understanding of sustainability, which was not limited to an environmental or technical perspective. Only when prompted on the dimensions, the interviewees related specific aspects, and, unsurprisingly, they could relate better to the environmental, economic, and technical dimensions than to the individual and social ones. Emphasis on action for sustainability in their private lives centered around the environmental dimension, including activities like saving energy and recycling. In their work lives, it was focused strongly on technical sustainability, especially in terms of maintainability, coding practices and team management. These findings confirm the results reported for experienced software [32] and requirements [29] engineering practitioners, and professional game developers [31]. Even though we cannot [yet] draw wider conclusions on the generalisability of the results, this may serve as indicator that we need to better infuse the different dimensions of sustainability into software engineering education, and specifically give examples for the individual and social dimensions [55].

[13]The questions are:

- *How would you elicit a sustainability requirement? Could you utilise any existing techniques?*
- *How would you model a sustainability requirement? Could you utilise any existing techniques?*
- *How would you test or evaluate a sustainability requirement? Could you utilise any existing metrics and measures?*

B. Sustainability Requirements

The interviewees in this study sample appear to have *no considered opinion on or grounded knowledge of sustainability in software engineering in general and sustainability requirements in particular*. Though, when pressed for an answer, they relate the notion of sustainability requirements to software maintainability and longevity, which is consistent with findings from [29]. Moreover, our interviewees were not clear about the notions of functional and non-functional requirements either. While the missing knowledge on sustainability could be attributed to lack of this topic's coverage in Software Engineering Curriculum, we are certain that the notion of functional and non-functional requirements is presented to the students within several modules throughout their SE studies. Yet, as per this study, this knowledge has not been retained, suggesting that it has not been applied in common practice.

Furthermore, the SE community is well aware that non-functional requirements (such as maintainability, extendability, usability, security, etc.) are often the key factors in the software systems' success or failure. Yet, our interviewees lack proper understanding of these. One explanation of this contradiction may lie in the *"early career"* stage of our sample. As novice developers, they are likely to be focused on the specific functionality implementation allocated to them, while senior colleagues, such as team leads, or software architects handle the concerns of non-functional (including sustainability-related) requirements. This view is supported by prior work (e.g., [56]) where senior engineers demonstrate key engagement with NFRs.

Consideration of the concept of a sustainability requirements did not lead our interviewees to relate it to their practice (e.g., one even interpreting environmental dimension of sustainability as having to do with the weather). While disappointing, this result is not entirely unexpected, as the notion of sustainability is so very broad that, at first encounter, it seems not connected to a specific task. Moreover, requirements that directly relate to sustainability are not perceived as such, as they are traditionally treated under segregated headings (e.g., accessibility and usability directly relate to the individual sustainability dimension; maintainability to the technical and economic dimension, etc.). To address this challenge, related research on collecting domain-specific sustainability requirements has recently been initiated [57], as well as an effort to illustrate sustainability relevance to software process and products with examples [12], [58]. A collection of such practical examples could serve as a good educational resource for the novice practitioners, such as our interviewees.

C. Software Engineering: Principles and Practice

From our interviews we observe, that none of the participants considered principles specifically dedicated to sustainability. Some participants have their own sets of practices that pertain to system longevity and maintainability, and also used agile methods (like code reviews and pair programming). However, four out of eight interviewees, were not able to relate any software engineering practice with sustainability. This shows that there is a lack of coverage of sustainability in software engineering education, although there exist several examples that propose principles and practices for including and explicitly addressing sustainability in SE; some of this work is described in the background section of this paper. For example, the principles for sustainabiltiy design are explicitly listed in the Karlskrona Manifesto [10], and many practices are proposed by Naumann et. al. [15], Mahaux [59], Penzenstadler [60], Betz [39], Chitchyan et. al., [58]. This is even more evident as the study has been conducted with recently graduated students.

Overall, this leads to the question as to what education about sustainable software engineering should look like and how to integrate sustainability principles and practice into it? Should there be a dedicated teaching module for this, or should the topic be spread across the regular SE modules? One way of integrating sustainability education into the SE curriculum is through infusing the sustainability considerations into SE courses as an overall aspect of long-term (strategic) development, which prevents accumulation of sustainability debt (including technical debt, social and personal dissatisfaction, economic losses, and environmental costs) [39]. Indeed, sustainability is an ever present concern in SE, even if not always explicitly acknowledged [10].

VI. Conclusions and Outlook

This paper reported results of an interview study on the knowledge of novice software developers with respect to sustainability, sustainability requirements, and the relationship of these concepts to software engineering principles and practices.

Our findings highlight that sustainability is generally understood in its broadest sense i.e. *'capable of being endured'* but it also often encompasses environmental, economic, and technical dimensions. However, we also observe a gap in understanding the notions of social and individual sustainability, which is corroborated by the findings from other similar studies [33], [32], [31].

The results also suggest that while sustainability is not a primary or overarching focus of our interview participants' cohort, the concept of sustainability is valued highly by these novice software developers, with a strong emphasis on attaining technical sustainability through maintainability related activities. In this context, the drive towards achieving maintainability aligns with 'the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment' [54]. As such, it is important to recognize that sustainability is an explicit consideration even if the primary focus of the system under design is not sustainability; a view supported by the Karlskrona Manifesto [10].

In addition, the results highlight a deficit in the knowledge of basic software engineering theory and practice as it is generally understood by the software and requirements engineering community as a whole, as well as a misalignment with existing software engineering principles and practice in

relation to sustainability. While these results by themselves provide a Fisheye view lens of software engineering and sustainability, it raises serious questions regarding whether software engineering education at this particular institution is fit for purpose or whether it is symptomatic of a larger problem in the software engineering curriculum. To address this question, we require further research and evidence.

We note that the education sector as a whole has an important role to play in ensuring that software designers of the future fully understand the concept of sustainability and its integral relationship to the field of software engineering by bridging the chasm between software engineering and sustainability. Education presents a key avenue for improvement in understanding sustainability. Educators need to consider how to integrate sustainability into software engineering curricula and articulate the competencies required for successful sustainability design. In this context, there is a need to expand the pool of examples of how to include sustainability in software engineering curricula, particularly from the international organizations that provide curricula guidelines. In ACM/IEEE SWE Curricula, sustainability is mentioned only once. Similarly, in the SWEBOK curricula, sustainability is mentioned twice in the software economics area.

This paper provides further evidence that we must consider the question of what would *sustainable software engineering* education would look like? To further investigate the current situation with regards to software engineering education and sustainability, future work will extend this study by conducting more studies in different organizations from different countries. Based on the results of theses studies, we are planning to work on curricula guidelines and recommendations.

ACKNOWLEDGEMENTS

We would like to express our sincere thanks and gratitude to our friend and colleague Professor Christoph Becker[14], University of Toronto (Canada), for his significant contributions and insights into the ongoing research agenda of the Karlskrona Consortium into sustainability design[15]. We also gratefully acknowledge the eight participants who willingly gave up their time to participate in this study.

REFERENCES

[1] H. Walkington, "Students as researchers: Supporting undergraduate research in the disciplines in higher education," 2015.

[2] H. Pham, *Software reliability*. Wiley Online Library, 1999.

[3] F. P. Deek, J. A. M. McHugh, and O. M. Eljabiri, *Strategic Software Engineering: An Interdisciplinary Approach*. Auerbach Publications, 2005.

[4] R. Kitchin and M. Dodge, *Code/space: Software and everyday life*. Mit Press, 2011.

[5] A. Geist and R. Lucas, "Major computer science challenges at exascale," *Journal of High Performance Computing Applications*, vol. 23, no. 4, pp. 427–436, 2009.

[6] T. Hey, S. Tansley, K. M. Tolle *et al.*, *The fourth paradigm: data-intensive scientific discovery*. Microsoft research Redmond, 2009, vol. 1.

[7] C. Goble, "Better software, better research," *IEEE Internet Computing*, vol. 18, no. 5, pp. 4–8, Sept 2014.

[14]Christoph Becker: https://ischool.utoronto.ca/christoph-becker/

[15]Sustainability Design: http://sustainabilitydesign.org/

[8] Z. Merali, "Computational science: Error, why scientific programming does not compute," *Nature*, vol. 467, no. 7317, pp. 775–777, 2010.

[9] M. M. Lehman, "Software's future: Managing evolution," *IEEE software*, vol. 15, no. 1, pp. 40–44, 1998.

[10] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, "Sustainability design and software: The karlskrona manifesto," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ser. ICSE-SEIS '15, May 2015, pp. 467–476.

[11] 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 *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Society Track*, ser. ICSE-SEIS '17, 2017, pp. 3–12.

[12] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S. M. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, "Requirements: The Key to Sustainability," *IEEE Software*, vol. 33, no. 1, pp. 56–65, Jan-Feb 2016.

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

[14] G. H. Brundtland and UN World Commission on Environment and Development, *Our common future*. Oxford University Press, 1987.

[15] S. Naumann *et al.*, "The greensoft model: A reference model for green and sustainable software and its engineering," *Sustainable Computing: Informatics and Systems*, pp. 294–304, 2011.

[16] H. Koziolek, "Sustainability evaluation of software architectures: a systematic review," in *ACM SIGSOFT Conf. QoSA and ISARCS*. ACM, 2011, pp. 3–12.

[17] A. Molla, V. A. Cooper, and S. Pittayachawan, "It and eco-sustainability: Developing and validating a green it readiness model," *ICIS 2009 Proceedings*, p. 141, 2009.

[18] B. Penzenstadler, "Towards a definition of sustainability in and for software engineering," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13, 2013, pp. 1183–1185.

[19] L. M. Hilty and B. Aebischer, "Ict for sustainability: An emerging research field," in *ICT Innovations for Sustainability*. Springer, 2015, pp. 3–36.

[20] R. C. Seacord *et al.*, "Measuring software sustainability," in *ICSM*, Sept 2003, pp. 450–459.

[21] C. Calero *et al.*, "Quality in use and software greenability," in *Workshop RE4SuSy*, 2014.

[22] C. C. Venters, C. Jay, L. Lau, M. K. Griffiths, V. Holmes, R. Ward, J. Austin, C. E. Dibsdale, and J. Xu, "Software sustainability: The modern tower of babel," in *Proceedings of the Third International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy 2014)*, 2014.

[23] C. Calero and M. Piattini, *Green in software engineering*. Springer, 2015.

[24] C. C. Venters, L. Lau, M. K. Griffiths, V. Holmes, R. R. Ward, C. Jay, C. E. Dibsdale, and J. Xu, "The blind men and the elephant: Towards an empirical evaluation framework for software sustainability," *Journal of Open Research Software*, vol. 2, no. 1, 2014.

[25] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, "Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century," *IEEE Software*, vol. 31, no. 3, pp. 40–47, May 2014.

[26] J. L. Ramsey, "On not defining sustainability," *Journal of Agricultural and Environmental Ethics*, vol. 28, no. 6, pp. 1075–1087, 2015.

[27] J. A. Tainter, "Social complexity and sustainability," *Journal of Ecological Complexity*, no. 3, pp. 91–103, 2006.

[28] B. Knowles *et al.*, "Exploring sustainability research in computing: Where we are and where we go next," in *UbiComp*. ACM, 2013, pp. 305–314.

[29] R. Chitchyan, C. Becker, S. Betz, L. Duboc, B. Penzenstadler, N. Seyff, and C. C. Venters, "Sustainability design in requirements engineering: State of practice," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ser. ICSE-SEIS '16, 2016, pp. 533–542.

[30] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, and J. Clause, "An empirical study of practitioners' perspectives on green software engineering," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16, 2016, pp. 237–248.

[31] J. Kasurinen, M. Palacin-Silva, and E. Vanhala, "What concerns game developers?: A study on game development processes, sustainability and metrics," in *Proceedings of the 8th Workshop on Emerging Trends in Software Metrics*, ser. WETSoM '17, 2017, pp. 15–21.

[32] I. Groher and R. Weinreich, "An interview study on sustainability concerns in software development projects," in *Proceedings of the 43rd Euromicro Conference on Software Engineering and Advanced Applications*, ser. SEAA '17, 2017.

[33] R. Chitchyan, I. Groher, and J. Noppen, "Uncovering sustainability concerns in software product lines," *Journal of Software: Evolution and Process*, vol. 29, no. 2, February 2017.

[34] M. Mahaux, P. Heymans, and G. Saval, "Discovering sustainability requirements: an experience report," in *Intl. Working Conf. REFSQ*, 2011, pp. 19–33.

[35] K. Roher and D. Richardson, "A proposed recommender system for eliciting software sustainability requirements," in *Workshop USER*, 2013, pp. 16–19.

[36] ——, "Sustainability requirement patterns," in *Requirements Patterns (RePa), 2013 IEEE Third Intl Workshop on*, 2013, pp. 8–11.

[37] J. Cabot *et al.*, "Integrating sustainability in decision-making processes: A modelling strategy," in *31st ICSE*. IEEE, 2009, pp. 207–210.

[38] S. A. Kocak, "Green software development and design for environmental sustainability," in *11th International Doctoral Symposium an Empirical Software Engineering (IDOESE 2013). Baltimore, Maryland*, vol. 9, 2013.

[39] S. Betz, "Sustainability aware process management using xmlnets," in *Proceeding of the 28th EnviroInfo Conference*, 2014.

[40] C. K. Makropoulos, K. Natsis, S. Liu, K. Mittas, and D. Butler, "Decision support for sustainable option selection in integrated urban water management," *Environ. Model. Softw.*, vol. 23, no. 12, pp. 1448–1460, Dec. 2008.

[41] S. A. Kocak, G. I. Alptekin, and A. B. Bener, "Integrating environmental sustainability in software product quality," in *RE4SuSy 2015: Requirements Engineering for Sustainable Systems*, vol. 1416. CEUR, 2015, pp. 17–24.

[42] "ISO/IEC 25010:2011: Systems and software engineering. Systems and software quality requirements and evaluation (SQuaRE)," 2011.

[43] M. Z. Huber, L. M. Hilty, and M. Glinz, "Uncovering sustainability requirements: An exploratory case study in canteens," in *RE4SuSy 2015: Requirements Engineering for Sustainable Systems*, vol. 1416. CEUR, 2015, pp. 35–44.

[44] M. Glinz, "Rethinking the notion of non-functional requirements," in *Third World Congress for Software Quality*, Sep. 2005, pp. 55–64.

[45] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are "non-functional" requirements really non-functional?: An investigation of non-functional requirements in practice," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16, 2016, pp. 832–842.

[46] E. Fossey, C. Harvey, F. McDermott, and L. Davidson, "Understanding and evaluating qualitative research*," *Australian and New Zealand Journal of Psychiatry*, vol. 36, no. 6, pp. 717–732, 2002.

[47] J. W. Creswell, *Qualitative inquiry & research design: choosing among five approaches*, 3rd ed. SAGE, 2013.

[48] P. Mayring, "Qualitative Content Analysis," in *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, vol. 1, no. 2, 2000.

[49] Simple Collaborative Qualitative Analysis. [Online]. Available: http://www.saturateapp.com

[50] "Fisheye lens view: Interview guide & codebook," http://tinyurl.com/h8dz4s3.

[51] P. A. Ochieng, "An analysis of the strengths and limitation of qualitative and quantitative research paradigms," *Problems of Education in the 21st Century*, vol. 13, June/2009 2009.

[52] R. E. Jeffries, A. Anderson, and C. Hendrickson, *Extreme Programming Installed*. Addison-Wesley Longman Publishing Co., Inc., 2000.

[53] I. Sommerville, *Software engineering*, tenth, global ed. Pearson Education Limited, 2016.

[54] P. Bourque, R. E. Fairley, and I. C. Society, *Guide to the software engineering body of knowledge: SWEBOK, version 3.0*. IEEE, 2014.

[55] R. Chitchyan, S. Betz, L. Duboc, B. Penzenstadler, S. Easterbrook, C. Ponsard, and C. Venters, "Evidencing sustainability design through examples," in *Fourth International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)*, August 2015.

[56] D. Ameller, C. P. Ayala, J. Cabot, and X. Franch, "Non-functional requirements in architectural decision making," *IEEE Software*, vol. 30, no. 2, pp. 61–67, 2013.

[57] M. Al Hinai and R. Chitchyan, "Engineering requirements for social sustainability," *ICT4S*, 2016.

[58] R. Chitchyan, W. Cazzola, and A. Rashid, "Engineering sustainability through language," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ser. ICSE '15, vol. 2. IEEE, 2015, pp. 501–504.

[59] M. Mahaux, P. Heymans, and G. Saval, "Discovering sustainability requirements: an experience report," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2011, pp. 19–33.

[60] B. Penzenstadler, "Infusing green: Requirements engineering for green in and through software systems." in *RE4SuSy@ RE*, 2014, pp. 44–53.