Exploring Assessment Criteria for Sustainable Software Engineering Processes

Michael Wahler
Zurich University of Applied Sciences (ZHAW)
Winterthur, Switzerland
wahl@zhaw.ch

Norbert Seyff
University of Applied Sciences and Arts Northwestern Switzerland (FHNW)
Windisch, Switzerland
norbert.seyff@fhnw.ch

Maria Susana Soriano Ramirez
Zurich University of Applied Sciences (ZHAW)
Winterthur, Switzerland
sori@zhaw.ch

ABSTRACT

It is our duty as software engineers to understand our contribution towards sustainability and ultimately assess and improve the sustainability of the software engineering (SE) processes we apply. However, commonly established criteria for such an assessment are currently lacking. In this experience report, we share insights from our investigation into the sustainability of software engineering processes, focusing on a collaborative project with an industry partner. Our research delves into lessons learned while exploring this critical issue. Our contribution lies in the introduction of an initial framework, which includes assessment criteria as the core element, and on the results of using this framework to assess the software engineering process of our industry partner. By sharing our experiences and findings, we aim to contribute to the understanding of sustainable software engineering practices and stimulate dialogue on how software engineering can address societal and environmental challenges. Our work underscores the significance of adopting sustainable practices and encourages the software engineering community—in both academia and industry—to embrace a proactive role in advancing sustainability for society.

CCS CONCEPTS

• **Software and its engineering → Software development process management**; *Collaboration in software development*.

KEYWORDS

software engineering, sustainability, criteria, assessment

ACM Reference Format:
Michael Wahler, Norbert Seyff, and Maria Susana Soriano Ramirez. 2024. Exploring Assessment Criteria for Sustainable Software Engineering Processes. In *Software Engineering in Society (ICSE-SEIS'24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3639475.3640109

Lay Abstract. It is important for software engineers to understand the effects of their work on sustainability. This also includes understanding the sustainability of the software engineering processes used and figuring out how to make them more sustainable. The problem is that currently, it is hard for software engineers and software companies to understand how sustainable their software engineering process is. There is a lack of criteria and established assessment frameworks that would allow that. In our experience paper, we talk about what we have learned from our efforts in establishing the first version of such a framework. Key contributions include identifying a set of criteria for the assessment and performing a first assessment in an industrial setting. We want to share what we have learned so that others can understand how to make software engineering processes more sustainable. The results are relevant for both researchers and practitioners.

ICSE-SEIS'24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0499-4/24/04.
https://doi.org/10.1145/3639475.3640109

1 INTRODUCTION

Digital products and services have the potential to significantly increase the sustainability of critical processes that impact society, for example by improving energy efficiency [11]. These solutions are shaped by the software engineers and the software engineering process applied. Poor requirements engineering, a lack of user involvement, inadequate architectural decisions, insufficient testing and quality assurance are issues that can lead to software solutions that have a negative effect on society.

Moreover, the immediate negative consequences of an insufficient software engineering process are not limited to the software product alone. Poor communication, ignored stakeholders, inadequate planning, overworked developers, ageism, the unnecessary consumption of resources, and cultural problems can have a profound and immediate impact on individuals involved in software engineering and the broader society.

In this context, we argue that sustainable software engineering processes (agile and plan-driven) should encompass a comprehensive perspective, considering the five dimensions of sustainability: environmental, social, individual, economic, and technical, as defined in [8]. With an estimated 26.9 million people working in software engineering around the world [13], understanding and assessing the sustainability—i.e., the ability to endure—of software engineering processes brings significant benefits to society. For example, by promoting eco-friendly practices and optimizing resource utilization, it contributes to environmental preservation and climate change mitigation. Emphasizing sustainability in software engineering aligns with societal values, paving the way for a more equitable, environmentally responsible, and socially conscious digital landscape.

Yet, we are not aware of generally accepted criteria for the sustainability of software engineering processes. Therefore, we have started to identify key criteria ourselves.

In this paper, we report on our experiences in a collaborative research project with an industry partner on the identification of specific criteria relevant to the sustainability of software engineering processes and how we utilized them to explore the creation of an initial assessment framework. We hereby focus on aspects that are directly related to software engineering activities. Other common human activities, such as commuting or heating an office building, which might occur in relation to dedicated development activities, lie beyond the immediate purview of this research. Whereas sustainability is often associated with ecological aspects only, principles and values play a pivotal role in the pursuit of sustainable software development [3]. These commitments, including multidisciplinarity, transparency, and flexibility, must be integrated throughout all phases of the software engineering process for any digital solution. We consider the experience gained in the discussions and in the identification and evaluation of criteria with our industry partner to be equally important for this paper, and in this context we place particular emphasis on presenting lessons learned. Given the importance of the topic, we consider these experiences to be highly relevant for both researchers and practitioners.

This paper is structured as follows. In Section 2, we present our research goal and questions and discuss key steps of our research. The identified criteria are presented in Section 3. In Section 4, we use these criteria to assess the sustainability of the software engineering process applied by our industry partner. In Section 5, we present threats to validity and related work and conclude the paper with Section 6.

2 RESEARCH GOAL, QUESTIONS AND METHOD

Our overall research goal is to identify criteria for assessing the sustainability of software engineering processes and to build an assessment framework for assessing its sustainability. Based on this main goal we have derived two research questions (RQs):

RQ1 What aspects have an influence on the sustainability of software engineering processes?

RQ2 How can software engineering processes be assessed regarding their sustainability?

The work was carried out as part of a funded research project with Rieter, a manufacturer of machines for the textile industry. Rieter develops a digital tool that analyzes data of their machinery and calculates key performance indicators. Sustainability is a clear business objective of the company as a whole. The goal of the research project was to find out how their software engineering activities relate to the sustainability targets of the company.

To conduct our research, we adopted the Design Science research methodology [23]. Design Science is a suitable research methodology for our work as it aligns with the goal of creating innovative artifacts to address the specific problem of sustainability in software engineering processes. By adopting Design Science, we aim to design, develop, and evaluate a novel assessment framework tailored to promote sustainable software development practices. The iterative nature of Design Science allows us to refine the artifact through continuous evaluation and improvement, ensuring its relevance and applicability in real-world contexts.

In our research, we adopted a systematic approach using the Relevance, Design, and Rigor Cycles. We initiated the Relevance Cycle by identifying the significance of sustainable software engineering processes and their practical implications. We analyzed the existing software engineering process and its documentation of our industry partner to understand the environment and the relevance of the problem in a real-world scenario as part of the first Relevance Cycle. Furthermore, we conducted a literature review on sustainability design and software. This step allowed us to establish a strong knowledge base as part of the first Rigor Circle. Moving on to the Design Cycle, we created a light-weight assessment framework tailored to address the identified criteria and dimensions. Building on the knowledge base and foundations established in the Relevance Cycle, we formulated a set of questions aimed at various roles within the company to address our research goal. We followed the problem-centered interview model defined by Mayring [15] to refine the questions and ensure their effectiveness in capturing essential aspects of sustainable software engineering processes. Subsequently, we identified criteria for sustainability assessment through collaborative discussions and agreement on a final set. The list of criteria provides answers to RQ1. Moreover, we systematically mapped each criterion to sustainability dimensions defined by Duboc et al. [8], thereby ensuring a comprehensive evaluation framework. The refined questions now aligned with criteria and dimensions, effectively addressing RQ2. We understand our results as worthwhile, but also as initial results. As our work progresses, further cycles will be necessary to refine and enhance the framework. As part of the evaluation within the Design Cycle, we first conducted an initial evaluation with our industry partner. We sought qualitative feedback from the Head Competence Center Digital on the identified assessment criteria. They provided valuable insights, validating the relevance and understandability of the criteria. This feedback from industry experts enhances the practical applicability and effectiveness of our assessment framework. Furthermore, and more importantly, we applied the assessment framework to evaluate the sustainability of the software engineering processes used by the company. This step allowed us to gain first-hand experience and insights into the framework's performance and its ability to assess and eventually improve the sustainability of real-world development processes.

3 IDENTIFIED SUSTAINABILITY CRITERIA

In this section, we investigate RQ1: What aspects have an influence on the sustainability of software engineering processes?

As initially mentioned in Section 2 of our paper, in our investigation, we made ourselves familiar with the existing software engineering process of our industry partner by talking to different employees about the process and a collection and analysis of the process documentation provided. This was done by the first and the third author of the paper. Additionally, the first and second author conducted a comprehensive, but informal literature review on sustainability design, sustainable software engineering, stakeholder characterization, sustainable project management, digital sustainability, sustainable deployment infrastructures, and energy consumption analysis within the field of software engineering. This review served as the foundation for our knowledge base, providing

insights into existing frameworks and theories. As part of the first Design Cycle, both inputs (process and literature analysis) were used to define relevant sustainability criteria, which are observable characterizing traits that contribute to the sustainability of the software engineering process. Criteria were identified by the first and second author of the paper individually, and refined by individually discussing the identified criteria and agreeing on a final set. In particular, criteria that both authors have identified were added to the final list without further discussion. For criteria that only one author had identified, a discussion was started, and a final agreement of the other author was needed to add the criterion to the list. In a last step, the concrete names of the criteria were jointly defined by both authors. This list was finally reviewed and validated by the third author of the paper and the industry partner.

So far, we have identified 38 sustainability criteria. The list of all criteria with all details and scientific references would quadruple the size of this paper. Therefore, we have uploaded a document, which presents all criteria in detail, to an archival repository [22]. We have also created a website[1] where we show the criteria as cards (see Figure 1) with the affected dimensions of sustainability, a description, possible consequences if the criterion is neglected, different levels of maturity at which the criterion can manifest, the references to the scientific articles and books that motivated the criteria, and examples of best practices (on the back of the card).

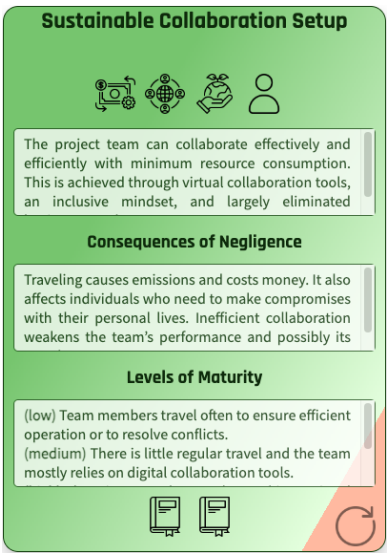

Figure 1: Graphical representation of sustainability criteria

We cluster our sustainability criteria in 5 groups:

Software Engineering Best Practices. We have found that many best practices in software engineering have a positive impact on sustainability. We present the identified criteria for this category in Table 1.

[1]https://criteria.greensoftware.ch/

Implications of Software Operations. Software operations impacts important aspects of sustainability such as energy consumption and cost. Therefore, a sustainable development process must consider operational aspects (which is, for example, done in DevOps). We list the identified criteria in Table 2.

Sustainable Team Culture. Two dimensions of sustainability (*social* and *individual*) directly concern human aspects. Therefore, it is important that the culture of the software development team embraces sustainability. We list the identified criteria for this category in Table 3.

Sustainability Awareness. For any software engineering process to be sustainable, it is important that all involved stakeholders are aware of how their decisions and actions influence sustainability. Our criteria in Table 4 therefore concern process aspects that are important for any technical or non-technical stakeholder in the development process.

Sustainable Procurement and Governance. The criteria in this category are motivated by the fact that a software engineering process can only be considered sustainable if it takes place in a sustainable environment. This requires a sustainable supply chain and effective governance mechanisms. The criteria are shown in Table 5.

3.1 Feedback from Industry Partner

After having documented our sustainability criteria, and as evaluation within the Design Cycle, we asked our industry partner for feedback and in particular discussed four key questions:

1. Are the criteria and their associated details understandable*?* According to our industry partner, the criterion "*Usage of tools to assess sustainability*" requires that there be metrics in place that can be measured and thresholds for these metrics be defined. We clarified this criterion, focusing on the presence of tools and their frequency of use [22] instead on the availability of metrics.

2. Do you consider the criteria as relevant *for assessing the sustainability of your software engineering process?* Our industry partner suggested the criterion "*Sustainability Incentive*" to be removed because the relation between costs and sustainability was not clear. In the initial version of this criterion, this relation was not sufficiently explained. We therefore improved the details of this criterion and are convinced that with this further explanation, the criterion is relevant.

3. Do you consider the list of criteria as complete*? If not, what are you missing?* Our industry partner suggested an additional criterion "*Sustainability in Deployment Planning*". We explained that this is covered by the existing criteria "*Implications of Software Operations*" and "*Development for Efficient Execution*" and improved the details of these two criteria to reduce misunderstandings in future assessments.

4. Do you have any additional comments or ideas? Our industry partner also provided 11 comments on additional ideas. Most of these ideas are about the evidence that the development process needs to provide in order to meet the respective criterion. We used these ideas to refine the levels of maturity for those criteria [22].

Table 1: Sustainability Criteria in *Software Engineering Best Practices*

Title	Description
Automatic Quality Checks	Automatic quality checks determine metrics and calculate KPIs that reflect the code quality.
Business Continuity of the Development Environment	The development environment (e.g., frameworks, libraries, operating systems) and process can be effectively adapted to new or changing technologies, business requirements, or regulations. It is important to notice that these are major changes and in the practice only occur in rare occasions.
Code Maintainability	The code has a high degree of maintainability.
Software Engineering Best Practices	The process uses best practices in software engineering (such as configuration management, continuous integration and delivery, or devops).
Sustainability Quality Attributes	The requirements document explicitly states sustainability of the product as a quality attribute, which is further refined.
Sustainability in Different Process Phases	Sustainability considerations are done in different phases of the process.
Sustainability in Release Planning	Sustainability considerations are part of release planning.
Sustainable Data Structures	The architect team considers needs of the environment, individuals, and society, as well as economical aspects while deciding on which data structures to use. They are optimized with respect to sustainability (e.g., energy consumption or extensibility).
Sustainable Design Decisions	Design decisions are written in a sustainable way. For each decision, its impact on the sustainability of the software is explicitly documented.
Sustainable Test Management	There is a policy in place to ensure that the right tests are run at the right time

Table 2: Sustainability Criteria in *Implications of Software Operations*

Title	Description
Development for Efficient Execution	The development process involves optimization steps to reduce the energy consumption of the software under development.
Implementation of Resource-Intensive Operations	For resource-intensive operations (e.g., computation-intensive calculations), measures are in place to assess the resource consumption and optimize it where needed.
Implications of Software Operations	The effects on sustainability of the operational tools, processes and mechanisms for the software are transparent.

Table 3: Sustainability Criteria in *Sustainable Team Culture*

Title	Description
Multidisciplinarity of the Development Team	Different disciplines have been included in the development of the software.
Participation of the Team	The proposals and strategies to improve the sustainability of the software engineering process of the development team should be heard and be evaluated by a governing body.
Sustainability Incentive	If the development team acts efficiently and stays under budget, it is not punished by having its budget reduced in the future.
Sustainable Collaboration Setup	The project team can collaborate effectively and efficiently with minimum resource consumption. This is achieved through virtual collaboration tools, an inclusive mindset, and largely eliminated business travel.
Sustainable Team Culture	Team members have an understanding of the importance of ethics, social communication, and respect for their team members.
Transparency of Communication	Project documentation, including minutes of meeting, is available to all stakeholders.

Another suggestion was that several sustainability criteria are dependent on other criteria. As an example, the criterion "*Value of Sustainability*" requires the presence of sustainability awareness in the organization. This feedback has helped us to group our criteria, which can be exploited to make assessments leaner; if the assessors determine the absence of any sustainability awareness, the assessment of subordinate criteria such as "*Value of Sustainability*" can be skipped. Based on this feedback, we improved the description and structure of our sustainability criteria before we carried out our assessment.

Table 4: Sustainability Criteria in *Sustainability Awareness*

Title	Description
Ability to Handle Changing Requirements	The process is able to adapt to changing requirements.
Availability of Metrics	There are metrics available that allow sustainability to be measured.
Capacity for Technical Debt Reduction	The software engineering process must provide capacity for reducing technical debt (e.g., refactoring, updating/complementing documentation).
Consideration of Different Orders of Effects	Different orders of effects are considered when analyzing the sustainability of the process.
Continuous Sustainability Improvement	There is a regular assessment of the sustainability of the development process and potential gaps are addressed.
Different Sustainability Dimensions	The process has means to investigate sustainability in different dimensions.
Energy Consumption of the Development Process	There are policies in place to assess and reduce the energy consumption in software development.
Knowledge about Sustainability	The stakeholders of the project possess sufficient knowledge about sustainability.
Strong Feedback Loops	The process uses feedback loops to regularly monitor external feedback and feeds it into the process to improve quality.
Sustainability Awareness	All stakeholders must be aware of the need for sustainability and its potential implications on the process and the product. They must also be aware how their role influences the sustainability of the software engineering process.
Sustainability Reporting	Sustainability is a distinct part of project reporting.
Usage of tools to assess sustainability	There are tools in use to assess sustainability in a software system development
Value of Sustainability	The company puts a value on sustainability. This is a prerequisite for investing money and making tradeoffs between requirements that are related to sustainability and those that are not.
Willingness to Change Requirements	There is a willingness to change the software system requirements to make the system more sustainable.
Willingness to Change the Process	There is a willingness to change the process towards higher sustainability.

Table 5: Sustainability Criteria in *Sustainable Procurement and Governance*

Title	Description
Direction and Policies to Improve Sustainability	The development process of a software product must be directed by regulatory obligations of sustainable digitalization, stakeholders expectations and business needs.
Sustainable Infrastructure	The choice and decision of the infrastructure on which the software is built takes sustainability into account. Such design decisions (e.g., PaaS, IaaS) are well documented.
Sustainable Procurement and Governance	Procurement and governance are driven by sustainability principles.
Technologies for System Development	The project uses and/or adopts sustainable technologies for system development.

3.2 Lessons Learned

In this section, we present key lessons learned of investigating sustainability criteria.

There might not be a complete set of sustainability criteria, but what we have is good enough for now. During our investigation of sustainability criteria, we discussed whether it was possible to create a set of criteria that we can consider as *complete*, i.e., any other criteria imaginable would be subsumed by the previously identified criteria. The short answer is "no", but the long answer shows that this does not make our approach any less valuable.

We consider the set of criteria we have identified as a snapshot of the cross-product of what is covered in the literature and what is relevant to our industry partner. The important thing for us is that these are the criteria that allow our industry partner, and, ideally, other companies who also consider them relevant, to grow and improve their software engineering process towards sustainability. We expect that working with other companies and also working together with our industry partner for a longer period of time will allow us to identify further criteria.

Focus on development, but don't forget about runtime and the software. Our criteria focus on software development and cover the whole life cycle. In the discussions with our industry partner, it surfaced that the operations phase has strong ties with the sustainability of the software product itself, where there is already a lot of research work available (e.g., on energy consumption in cloud computing [4] or mobile computing [10]). We conclude that the sustainability of the operations phase (in particular, the sustainability of the software *product*) needs to be considered together with the sustainability of the development *process* to see the larger picture.

We suggest that a sustainable software engineering process

should incorporate the continuous analysis of the software's effects on various sustainability dimensions. Approaches like the Sustainability Awareness Framework (SusAF) [8] aim to provide an initial analysis of potential effects based on the system vision. However, we argue for a continuous analysis of effects, even during system runtime, which necessitates further research and the development of novel approaches and tools to support practitioners in this endeavor.

Sustainability strategies are available, but are not holistic. Today, many companies have a publicly available sustainability strategy. From what we have seen at our industry partner, their strategy covers several dimensions of sustainability. However, it was not clear to us (or, them) how their sustainability strategy affects software development and, conversely, how sustainable software development, as anticipated, affects the general sustainability strategy. Our key learning is that for the future, there needs to be traceability between the sustainability strategy of companies and their software engineering process(es). This traceability should go in both directions.

We suggest that practitioners use the criteria presented in this paper and trace them to their sustainability strategies. As an example, if the company as a whole wants to reduce energy consumption, the criteria "Sustainable Collaboration Setup" or "Implementation of Resource-Intensive Operations" are strongly related.

Companies can learn from software engineering. While conducting our research, we have come to the conclusion that certain aspects of software engineering knowledge and practices already contribute to fostering sustainability within the software engineering process. We foresee that some of these practices can not only support sustainable software engineering, but also serve as a model for implementing sustainability in non-technical processes and practices within the company. In particular, agile development [2] suggests principles that improve the social (e.g., through participation of the customer), individual (e.g., through focus on trust), technical (e.g., through focus on delivering working software), and economical (e.g., through focus on value and constant pace) dimensions of sustainability. As an example, agile enterprises should use customer feedback as "the driver that steers the engine of customer value" [16]. We suggest that researchers and practitioners explore how aspects and practices that support sustainable software engineering can also be applied to other processes within the company.

4 ASSESSMENT OF THE SOFTWARE ENGINEERING PROCESS

In this section, we provide first answers to RQ2: How can software engineering processes be assessed regarding their sustainability? Hence, an initial assessment, based on the 38 sustainability criteria, was conducted with our industry partner. As described in Section 2, this is part of our evaluation activities within the Design Cycle and was conducted by the first and third author of the paper. To answer the question, we had to consider a few specifics.

First, our criteria are predominantly *qualitative* and therefore, they must be assessed manually based on the personal judgment of the assessors. This makes the assessment prone to subjectivity and difficult to replicate, at least to some extent. However, such a manual assessment (e.g., through a series of interviews) can provide a richer context and insights, capturing nuances that quantitative criteria may miss.

Second, our industry partner is interested in the aforementioned context and insights for improving their software engineering process rather than obtaining a score that would allow them to advertise their software development with a sustainability certificate.

Third, our industry partner is just starting their journey towards sustainability in software development. Therefore, they are not only interested in an initial assessment, but also in repeating the assessment and continuously monitoring their efforts to improve the sustainability of their software engineering process.

Due to these facts, we decided with our industry partner to aim for and develop a lightweight, open assessment framework that can be easily applied by software companies even without the involvement of external experts. This might stand in contrast to established approaches such as lifecycle assessments (LCA) or established maturity models (e.g., CMMI).

Our light-weight framework consists of three steps. In the following paragraphs, we describe how we applied the assessment framework to initially assess the sustainability of the software engineering process of our industry partner. This, at the same time, discusses the three steps we recommend for performing an assessment and shows how we actually performed the assessment.

The assessment was done within the scope of their key software development project, where they are building a complex and large piece of industrial software.

4.1 Step 1: Data Collection

Carrying out our first assessment, we involved 7 stakeholders involved in the software engineering process at our industry partner. This included software engineers, but also the product owner.

Participants in the assessment were asked to rate the sustainability of their software engineering processes based on the criteria we have identified. This rating was based on their knowledge and perception of the process.

As a tool to conduct the assessment, we developed a questionnaire basically listing the criteria and an explanation. Furthermore, there were 6 possible response choices for each criterion, with four response choices on the level of maturity (NONE, LOW, MEDIUM, HIGH) and two with the NOT APPLICABLE and DON'T KNOW responses. Next to each response choice, a follow up text box was available, so participants could write the reason for their selection. The questionnaire was made available to the participants via Microsoft Forms. This also allowed for easy data collection, as each response was stored separately in an Excel document.

Each session of data collection was conducted during an online video call with one or two individual participants, where they gave their independent assessments. In each call, at least one of the authors of this paper was present to provide guidance and support. This mainly included providing an introduction, explaining the procedure, and answering questions about the criteria. There otherwise was no discussion amongst the people in the call.

4.2 Step 2: Results Analysis

In the next step, we wanted to create a ranking of the criteria based on the responses in the assessment. Such a ranking allows

Table 6: Average score and variance of each criterion

Criterion	avg	σ^2
Multidisciplinarity of the Development Team	3.00	1.14
Software Engineering Best Practices	2.86	4.81
Capacity for Technical Debt Reduction	2.50	1.81
Sustainable Collaboration Setup	2.50	3.14
Sustainable Team Culture	2.50	0.81
Ability to Handle Changing Requirements	2.33	1.14
Code Maintainability	2.00	1.14
Strong Feedback Loops	2.00	1.47
Willingness to Change the Process	2.00	0.67
Transparency of Communication	2.00	1.14
Automatic Quality Checks	1.80	1.00
Business Continuity of the Development Environment	1.67	2.14
Willingness to Change Requirements	1.57	1.14
Implementation of Resource-Intensive Operations	1.50	0.47
Sustainable Test Management	1.50	0.47
Continuous Sustainability Improvement	1.33	1.00
Participation of the Team	1.33	0.47
Sustainability in Different Process Phases	1.29	0.81
Sustainable Design Decisions	1.25	1.14
Sustainability Reporting	1.25	0.67
Implications of Software Operations	1.00	1.14
Sustainability Awareness	1.00	1.47
Value of Sustainability	1.00	3.47
Availability of Metrics	1.00	0.67
Sustainable Procurement and Governance	1.00	3.14
Knowledge about Sustainability	0.86	2.14
Development for Efficient Execution	0.80	1.33
Sustainability in Release Planning	0.80	1.00
Sustainable Data Structures	0.75	1.81
Sustainability Incentive	0.75	0.33
Sustainability Quality Attributes	0.60	2.14
Usage of tools to assess sustainability	0.50	1.00
Energy Consumption of the Development Process	0.25	1.33
Different Sustainability Dimensions	0.00	2.81
Consideration of Different Orders of Effects	0.00	3.47
Direction and Policies to Improve Sustainability	0.00	1.25
Sustainable Infrastructure	0.00	0.58
Technologies for System Development	0.00	0.58

us to easily identify the strengths and weaknesses of the software engineering process in terms of sustainability. To this end, we calculated a score for each criterion by awarding points for the different levels of maturity: 0 points for NONE, 1 point for LOW, 2 points for MEDIUM, and 3 points for HIGH. For NOT APPLICABLE and DON'T KNOW responses, 0 points were awarded. Because of the relatively low number of responses, the median response can be ambiguous. Therefore, we decided to calculate the average score and the variance σ^2, indicating how far the responses are spread out. A low value for σ^2 thus indicates disagreement between the respondents, whereas a high value means that the respondents agree to a large extent. Table 6 shows the criteria sorted by the average score in a descending way.

From these results in Table 6, we can draw several conclusions:

(1) 84 % of all criteria (32 out of 38) already show some level of maturity (score ≥ 0.5). We consider this a respectable result for a first assessment.
(2) Several of the criteria at the bottom of the list (i.e., with a low score) might relate to criteria where more fundamental knowledge regarding sustainability (e.g., sustainability dimensions, orders of effects) is required in order to receive higher scores. As mentioned earlier, the software engineering team at the industry partner has not had any formal training on sustainability. We are convinced that by making sustainability an official process goal and providing basic knowledge and training in this regard to all stakeholders, the results can be dramatically improved.
(3) Criteria with a variance $\sigma^2 \geq 1.0$ show a sufficient level of agreement among the respondents. For criteria with $\sigma^2 < 1.0$, we suspect that the low variance could also be attributed to differing notions of sustainability among practitioners. However, this requires further investigation.
(4) Despite such open questions and a rather small sample size of interviewees, the results can be used to recognize strengths and weaknesses and to define improvement measures.

4.3 Step 3: Recommendations

Striving for continuous improvement is an important aspect in sustainable software engineering processes. Therefore, we see the value of such a sustainability assessment not only in computing an average score, but rather in deriving recommendations for improving the status quo.

The results clearly show some of the strengths and weaknesses of the software engineering process with respect to sustainability. As can be seen, the software engineering team already uses many best practices for sustainable software engineering although it has never received training on the concept of sustainability. In a first follow-up discussion with our industry partner, we decided that the currently used practices (such agile software engineering and virtual collaboration across countries) should be continued. Since several of the identified weaknesses concern a lack of formal training on sustainability, we suggest including basic sustainability training for all project stakeholders and raising sustainability awareness at regular intervals, e.g., at sprint planning meetings. We suspect that such training will help to establish a common understanding of sustainability in the team. This, in turn, could help to increase the variance σ^2 in the responses for future sustainability assessments and likely lead to improved sustainability practices.

4.4 Feedback from Industry Partner

After having performed our sustainability assessment, and as evaluation within the Design Cycle, we asked our industry partner for feedback and in particular discussed three key questions:

1. What are your own conclusions on your weaknesses and strengths?
Our industry partner is happy to see that agile principles and sustainability go hand-in-hand, which resulted in most criteria being at least partially fulfilled. They will continue to foster best practices in software development, having a good team culture, and reduce business travel to what is needed. Therefore, it was not a surprise to them that their process scored high values for the corresponding

criteria. On the other hand, they expected relatively low scores regarding the knowledge and awareness of conceptual aspects of sustainability because their software engineering team has never had a formal training on these aspects. They now see the value in such a training, and they are planning to roll out training sessions on sustainability as integral part of their Learning Management System (LMS).

2. How do you respond to the conclusions the researchers made upon evaluation? Our industry partner thinks that our conclusions are comprehensible and valid. Despite the rather small number of participants in the assessment, they think that it is quite accurate having had representatives of each stakeholder group participating in the assessment (from developers through business analysts to management).

3. How are you going to move forward from there? Our industry partner has already added sustainability as a "development value" to their software development principles and values. As one of the outcomes, the use case templates were updated accordingly. As a next step, they are going to implement additional improvement measures such as sustainability awareness trainings to selected stakeholders and having regular sustainability assessments as a part of their software release process to monitor their ongoing sustainability initiatives. They are also planning to share the research and their findings with their industry peers through existing digital cluster meetings.

4.5 Lessons Learned

We have learned valuable lessons regarding the *assessment and the communication of the results*:

Limited sustainability awareness, but "implicit" measures were already present. While assessing the software engineering process of our industry partner, we observed that stakeholders lacked substantial sustainability awareness regarding software engineering and software products. Moreover, those who recognized a connection tended to have a limited perspective, primarily focusing on the environmental dimension of sustainability. This finding is consistent with previous research [6], which highlighted a similar lack of awareness in the industry. However, we also noticed that following a mature and quality-focused software engineering process several aspects of their process were already implicitly contributing to the sustainability of the process according to our expectations. However, we also noticed that by following a mature and quality-focused software engineering process, several aspects of their operations were already implicitly contributing to sustainability.

We suggest that practitioners ensure that *all* project stakeholders have a basic *knowledge* of the different sustainability dimensions and how their work affects these, as well as different order of effects.

Industry would like to have a certificate, but there is the danger of "greenwashing". Although our framework can be used to calculate a sustainability score for each criterion and a weighted total average score, we do not intend to use it to grant sustainability certificates. The main reason is that our sustainability criteria are largely qualitative and the assessment rather subjective. Although we encourage the introduction of quantitative criteria that can be objectively measured, we are afraid that our framework could be misused by software development teams to stop improving their sustainability once a desired score has been reached and use the framework for greenwashing, i.e., using the assessment for green marketing instead of a starting point for further improvements.

We suggest that practitioners use our sustainability criteria to stimulate discussions and improve their processes. They should not use the assessment results to simply reach a high score or compare themselves with competitors, but to benchmark their processes with previous iterations of these processes with the goal of continuous improvement.

Sustainability in software engineering is a process, not a state. We consider our assessment criteria and framework as a helpful companion that guides software development teams on their journey toward increased sustainability. There is little value in carrying out a single sustainability assessment to evaluate the status quo with respect to sustainability. In other words, achieving sustainability in the context of software engineering is not a fixed or static condition that can be reached and then forgotten. Instead, it is an ongoing and dynamic journey or series of actions and practices that need to be continually maintained and improved.

We suggest that practitioners repeat assessments regularly, e.g., at quarterly retrospective meetings, define follow-up actions, and measure the results of these actions.

Be aware of weaknesses, do not ignore strengths. When being evaluated, people and organizations may focus on their weaknesses (and sometimes, even get defensive). This has two consequences: First, they may oversee their strengths, which may even outweigh their weaknesses. Second, improving on one's weaknesses is typically more difficult than further refining one's strengths.

We suggest that practitioners take each sustainability assessment as an opportunity to discuss the current situation, put weights on their strengths and weaknesses, and carefully craft an improvement plan. This will allow them to approach sustainability in a holistic way and achieve the biggest progress.

5 DISCUSSION

In this section, we discuss next steps for our research, summarize the current limitations of our approach in the form of threats to validity, and present related work.

5.1 Next Steps

As our research continues, we plan to further validate our framework through additional case studies in different organizational settings and software development approaches, such as agile and plan-driven methods. This iterative process following Design Science allows us to continuously refine and enhance the framework, ensuring its reliability and effectiveness in assessing the sustainability of software engineering processes.

We consider our assessment framework for sustainable software engineering processes as a starting point for further exploration, which should be a joint endeavor of researchers and practitioners working together and continuously extending the suggested set of sustainability criteria based on their knowledge, experience, and

research done in the field of software engineering. For such an exploration, we consider three actors as important.

Our industry partner. Our industry partner states that our sustainability criteria have given them valuable insights about the maturity of their software development process related to sustainability. As the next step, they are going to discuss which of the identified weaknesses have the biggest perceived negative impact for them and start working on improving them. They will continue the discussion with us on their progress an jointly improve the criteria catalogue based on the findings.

For now, we are planning to continue our collaboration with our industry partner, offering guidance and mentorship in their pursuit of a more sustainable software engineering process based on the first evaluation results. This in particular could include educational interventions, such as training sessions or workshops focused on the foundational principles of sustainability to also improve awareness. This exchange supports further understanding our partner's expectations and requirements, contributing to the Relevance Cycle.

The authors. As authors of this research, we commit to an ongoing exploration of sustainability within the context of software engineering and digitalization. In particular, we want to use our assessment framework with other companies across different industries. This will help us to extend our current set of sustainability criteria with new insights, tailor it towards specific use cases, improve the assessment process, and align it with existing quality assurance and maturity assessment processes in the respective company. These activities can be seen as new Design Cycles and outputs will contribute to Rigor Cycles.

Software engineering community. In recognition of the collaborative and open nature of scientific research, we intend to release our sustainability criteria under a Creative Commons license and invite the software engineering community to contribute. This will help to ensure that the criteria cover a wide range of software engineering processes across many industries, standardize the terminology and levels of maturity, and create a higher awareness in the software engineering community. This emphasizes rigor through open collaboration and contribution to further refine and validate the sustainability criteria.

5.2 Threats to Validity

In this paper, we have presented the first results of our ongoing research. We are aware of several threats to validity, including internal, external, conclusion, and construct validity [24], especially also because of the exploratory nature of this paper. Key threats to validity regarding the identified assessment criteria include:

Internal validity. It can be affected by the method used to identify potential assessment criteria, which was done by the authors of this paper in cooperation with the industry partner. The authors consider themselves to be experts in the field of software engineering, and a first validation of the identified criteria confirms their relevance. However, further validation is needed. Furthermore, using a non-systematic literature review method to establish our knowledge base may introduce a threat to the research study's internal validity, particularly in terms of incomplete coverage.

External validity. So far, our focus is on our industry partner and their software engineering process. We are positive that the identified criteria are also relevant for other companies and other software engineering processes, and we are planning to validate our results within different use cases. Furthermore, the type of software developed (e.g., mobile apps), may have an influence on our work and needs to be considered in the planned studies.

Another issue is that most sustainability criteria that we have identified are qualitative. This means that a manual assessment is necessary, and the accuracy of the assessment depends on the fidelity of the levels of maturity. Thus, there is room for subjectivity in the assessment. More quantitative criteria can help to obtain more accurate and less subjective assessments.

Conclusion validity. The completeness of the identified assessment criteria is an issue. We argue that the list of identified criteria is not complete. By studying more development processes and refining sustainability targets, more criteria will eventually be identified. It is an open question if at any time, the list of criteria can be considered complete. It is also an open question if some properties of our criteria (such as their weight) will change over time.

The sample size of the assessment with our industry partner was of 7. This sample size is rather small and might not be representative. However, for this specific assessment (with an entire team size of around 50 people) we needed specialized populations, and interesting results can be drawn even from this small sample size.

Construct validity. In order to minimize the threats to construct validity, the criteria were identified based on questions used for an early analysis of the software process and the software product of our industry partner. These questions defined the scope and guided the identification of assessment criteria.

5.3 Related Work

To the best of our knowledge, there is no research that provides criteria to assess the sustainability of software engineering processes. Our work aims to fill this gap. Furthermore, we consider the participation of industry professionals and having a real-world case study as highly relevant to explore our assessment framework. Here is related work that complements our approach.

Lami et al. [14] investigate software process sustainability. Their contributions are on sustainability management, capability levels, and the environmental dimension of sustainability. Our work is focused on sustainability criteria across several dimensions and how these resonate with an industry partner. Their structured approach to sustainability management could be combined with our diverse set of criteria to build a more complete assessment approach.

The book from Calero and Pattini [5] covers many aspects of sustainable software engineering such as definitions, processes, or roles while covering multiple dimensions of sustainability. The knowledge in this book can be augmented with our sustainability criteria to obtain more concrete guidelines for practitioners. As an example, the chapter from Kern et al. [12] presents a process model for sustainable software engineering that can be integrated

into common engineering processes such as SCRUM. Similar to our criteria, this model covers the whole SDLC. Therefore, both approaches could be combined to obtain a holistically sustainable software engineering process.

More recently, the *social* and *individual* dimensions of sustainability have moved into the focus of software engineering researchers. In [9], Dutta et al. investigate the representation of different social groups in software engineering studies. Although a vast majority of the investigated studies report on the diversity of the participants regarding their *professional* background, there is a general lack of reporting on their *social* background. This may have negative impact on the software engineering process and its sustainability. In [25], Zhao and Young investigate workplace discrimination in software engineering and provide concrete suggestions to help reduce it. They have found out that workplace discrimination does exist and has "various negative impacts on software professions". This is, of course, detrimental to a sustainable team culture and thus, threatens the sustainability of the software engineering process.

Naumann et al. [18] define green and sustainable software and green and sustainable software engineering. They argue that a sustainable software product should have not only a low impact on sustainable development, but also, if it is its purpose, should promote it. The authors also mention the importance of organizations to be aware of the positive and negative impacts on sustainable development. They map sustainability-related quality attributes to the different phases of development (e.g., modifiability takes place in the development phase). Each phase can then be reviewed according to whether it meets the desired attributes [7].

Shenoy et al. [19] present some practices for the different phases of the SDLC to make them more sustainable. We consider some of these practices as not relevant anymore in today's software development (e.g.,"avoid paper") and other practices out of scope (e.g., avoid air-conditioning, see Section 1). However, we believe many best practices are still relevant; thus, we integrate them into our assessment framework.

Taina [20] analyzes a software life cycle and gives estimates of the carbon footprints each step produces. Ardito et al. [1] provide a conceptual framework that unifies strategies, models, and tools for designing and developing greener software. This framework aims to reduce software power consumption through code refactoring and by implementing self-adaptation from the starting point of the design and development of the software.

Venters et al. [21] define software sustainability as a measure of quality attributes such as extensibility, interoperability, maintainability, portability, reusability, and scalability. They propose that software sustainability can be achieved through software architectures and architectural evaluation methods. Additionally, they argue that software sustainability is a non-functional requirement which helps to understand how people can develop sustainable software in the future instead of focusing on how people should sustain existing software. The authors do not mention nor do they explore the five dimensions of sustainability as defined by [8].

Mourão et al. [17] provide a systematic mapping study to encompass state-of-the-art approaches for sustainable software engineering practice. They conclude that more research needs to be done in terms of techniques, tools, and metrics to cover construction (implementation), testing and maintenance of the software, since most of the relevant contributions are in the form of approaches and frameworks in the requirements and design phase of the software engineering process. The authors observe that most primary studies in sustainable software engineering use simulated scenarios and need to be evaluated in practice. They also found little evidence of participation of industry professionals in studies of sustainable software engineering. The authors conclude that industry professionals can enrich the findings of sustainable software engineering studies with their experiences and guide researchers to prioritize their research objectives.

6 CONCLUSIONS

With an estimated 26.9 million people working in software engineering around the world [13], understanding and assessing the sustainability of software engineering processes brings significant benefits to society. In this experience paper, we have introduced sustainability criteria for software engineering processes, presented the first version of an assessment framework, and reported on a first assessment of a software engineering process in the industry. Furthermore, we have presented several lessons learned. This allows the software engineering community to better understand what makes software engineering processes sustainable, how they can be assessed, and how their sustainability impacts society.

The first assessment that we performed has already had a positive impact on our industry partner. It raised sustainability knowledge and awareness of several members of the software engineering team. It also stimulated discussions about the extent to which some of the criteria are already met (or not). Thus, we conclude that our criteria and framework are a good fit for our industry partner. We are looking forward to extending our approach to evaluate its suitability for other companies and adapt and extend it as needed.

With the work conducted so far, we can also present first answers to our research questions. Regarding RQ1, "*What aspects have an influence on the sustainability of software engineering processes?*", we have identified a first set of criteria to assess software engineering processes, which was confirmed as relevant and sufficiently complete by our industry partner. As discussed in Section 5.2, further studies are, however, needed to evaluate and extend the criteria presented.

Regarding RQ2, "*How can software engineering processes be assessed regarding their sustainability?*", we have shown that our suggested initial assessment framework is a first step in the right direction because it has helped our industry partner to obtain a comprehensive view of the sustainability of their software engineering process and define measures to further improve it. As discussed in Section 5.1, performing assessments of the software engineering processes of other industry partners is a key next step for improving and extending our assessment process.

ACKNOWLEDGMENTS

This work is supported by the Zürcher Stiftung für Textilforschung (Project 147: Green and sustainable digitalization for the textile industry). We thank David H. Gehring from our industry partner Rieter for his commitment, valuable input, and for making himself and other key stakeholders available during the project.

REFERENCES

[1] Luca Ardito, Giuseppe Procaccianti, Marco Torchiano, and Antonio Vetro. 2015. Understanding green software development: A conceptual framework. *IT professional* 17, 1 (2015), 44–50.

[2] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. 2001. The agile manifesto.

[3] Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C Venters. 2015. Sustainability design and software: The Karlskrona manifesto. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 467–476.

[4] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. 2010. Energy-efficient cloud computing. *The computer journal* 53, 7 (2010), 1045–1051.

[5] Coral Calero and Mario Piattini. 2015. *Introduction to Green in Software Engineering*. Springer International Publishing, Cham, 3–27. https://doi.org/10.1007/978-3-319-08581-4_1

[6] Ruzanna Chitchyan, Christoph Becker, Stefanie Betz, Leticia Duboc, Birgit Penzenstadler, Norbert Seyff, and Colin C. Venters. 2016. Sustainability Design in Requirements Engineering: State of Practice. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 533–542.

[7] Markus Dick and Stefan Naumann. 2010. Enhancing Software Engineering Processes towards Sustainable Software Product Design. In *EnviroInfo*. Citeseer, 706–715.

[8] Leticia Duboc, Birgit Penzenstadler, Jari Porras, Sedef Akinli Kocak, Stefanie Betz, Ruzanna Chitchyan, Ola Leifler, Norbert Seyff, and Colin C Venters. 2020. Requirements engineering for sustainability: an awareness framework for designing software systems for a better tomorrow. *Requirements Engineering* 25, 4 (2020), 469–492.

[9] Riya Dutta, Diego Elias Costa, Emad Shihab, and Tanja Tajmel. 2023. Diversity Awareness in Software Engineering Participant Research. In *Proceedings of the 45th International Conference on Software Engineering: Software Engineering in Society* (Melbourne, Australia) *(ICSE-SEIS '23)*. IEEE Press, 120–131. https://doi.org/10.1109/ICSE-SEIS58686.2023.00017

[10] J. Flinn and M. Satyanarayanan. 1999. PowerScope: a tool for profiling the energy usage of mobile applications. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*. 2–10. https://doi.org/10.1109/MCSA.1999.749272

[11] Shahryar Habibi. 2017. Micro-climatization and real-time digitalization effects on energy efficiency based on user behavior. *Building and Environment* 114 (2017), 410–428.

[12] Eva Kern, Stefan Naumann, and Markus Dick. 2015. Processes for green and sustainable software engineering. *Green in Software Engineering* (2015), 61–81.

[13] Qubit Labs. 2022. How Many Programmers are there in the World and in the US? https://qubit-labs.com/how-many-programmers-in-the-world/. Online; accessed 2023-09-14.

[14] Giuseppe Lami, Fabrizio Fabbrini, and Mario Fusani. 2012. Software Sustainability from a Process-Centric Perspective. In *Systems, Software and Services Process Improvement*, Dietmar Winkler, Rory V. O'Connor, and Richard Messnarz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 97–108.

[15] Philipp Mayring. 2016. *Einführung in die qualitative Sozialforschung*. Beltz.

[16] E Moreira Mario. 2017. The Agile Enterprise–Building and Running Agile Organization. *Apress. Winchester, Massachusetts* (2017).

[17] Brunna C Mourão, Leila Karita, and Ivan do Carmo Machado. 2018. Green and sustainable software engineering-a systematic mapping study. In *Proceedings of the 17th Brazilian Symposium on Software Quality*. 121–130.

[18] Stefan Naumann, Markus Dick, Eva Kern, and Timo Johann. 2011. The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems* 1, 4 (2011), 294–304.

[19] Sanath S Shenoy and Raghavendra Eeratta. 2011. Green software development model: An approach towards sustainable software development. In *2011 Annual IEEE India Conference*. IEEE, 1–6.

[20] Juha Taina. 2010. How green is your software?. In *Software Business: First International Conference, ICSOB 2010, Jyväskylä, Finland, June 21-23, 2010. Proceedings 1*. Springer, 151–162.

[21] Colin Venters, Lydia Lau, Michael Griffiths, Violeta Holmes, Rupert Ward, Caroline Jay, Charlie Dibsdale, and Jie Xu. 2014. The blind men and the elephant: Towards an empirical evaluation framework for software sustainability. *Journal of Open Research Software* 2, 1 (2014), 1–6.

[22] Michael Wahler, Norbert Seyff, and Maria Susana Soriano Ramirez. 2023. Assessment Criteria for Sustainable Software Engineering Processes. https://doi.org/10.5281/zenodo.10417685

[23] Roel J. Wieringa. 2014. *What Is Design Science?* Springer Berlin Heidelberg, Berlin, Heidelberg, 3–11. https://doi.org/10.1007/978-3-662-43839-8_1

[24] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer, Germany. https://doi.org/10.1007/978-3-642-29044-2

[25] Xin Zhao and Riley Young. 2023. Workplace Discrimination in Software Engineering: Where We Stand Today. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. 188–193. https://doi.org/10.1109/ICSE-SEIS58686.2023.00026