# Assessment Criteria for Sustainable Software Engineering Processes

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

Norbert Seyff
University of Applied Sciences and Arts Northwestern Switzerland
norbert.seyff@fhnw.ch

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

December 2023

Abstract

We present a collection of sustainability criteria for assessing the sustainability of software engineering processes. These criteria were identified in a research project with an industry partner. The results of this project can be found in our research paper [WSSRed]. We recommend reading the research paper first before delving into this report because this report is meant as an appendix to the research paper and not a stand-alone document.

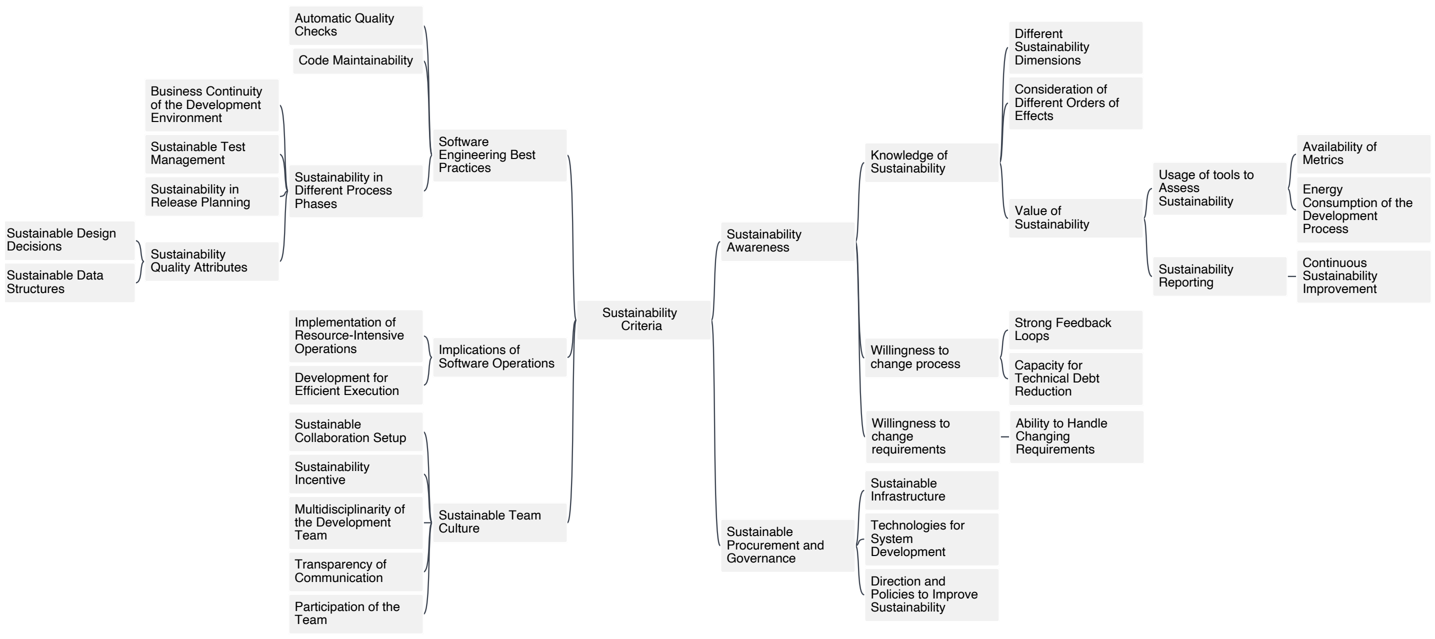
## Contents

# 1 Introduction

This report contains all details of our sustainability criteria for assessing the sustainability of software engineering processes that we present in our paper at ICSE 2024 [WSSRed]. The contents of this report are also available on a website [WSSR23], which gets extended when additional sustainability criteria are identified.

## 1.1 Overview

The sustainability criteria presented in this report are organized in 5 sections, as outlined in Figure 1. For more details, see our ICSE paper [WSSRed].



Sustainability Criteria
Software Engineering Best Practices
Automatic Quality Checks
Code Maintainability
Sustainability in Different Process Phases
Business Continuity of the Development Environment
Sustainable Test Management
Sustainability in Release Planning
Sustainability Quality Attributes
Sustainable Design Decisions
Sustainable Data Structures
Implications of Software Operations
Implementation of Resource-Intensive Operations
Development for Efficient Execution
Sustainable Team Culture
Sustainable Collaboration Setup
Sustainability Incentive
Multidisciplinarity of the Development Team
Transparency of Communication
Participation of the Team
Sustainability Awareness
Knowledge of Sustainability
Different Sustainability Dimensions
Consideration of Different Orders of Effects
Value of Sustainability
Usage of tools to Assess Sustainability
Availability of Metrics
Energy Consumption of the Development Process
Sustainability Reporting
Continuous Sustainability Improvement
Willingness to change process
Strong Feedback Loops
Capacity for Technical Debt Reduction
Willingness to change requirements
Ability to Handle Changing Requirements
Sustainable Procurement and Governance
Sustainable Infrastructure
Technologies for System Development
Direction and Policies to Improve Sustainability

Figure 1: Concept map of sustainability criteria

For each criterion, we display the dimensions of sustainability that are affected by it as defined in [DPP$^+$20]: economic, environmental, individual, social, and technical.

## 1.2 Disclaimer

The icons used in this report are created by kmg design[1]. The illustration on the cover page is AI generated.

[1] https://www.flaticon.com/authors/kmg-design

# 2 Category: Software Engineering Best Practices

## 2.1 Software Engineering Best Practices

The process uses best practices in software engineering (such as configuration management, continuous integration and delivery, or devops).

**Consequence of Negligence** Without software engineering best practices, developers might not be aware that the software development is using outdated techniques. This will lead to problems in the maintenance phase of the life cycle of the software.

**Levels of Maturity**

> low
>
> The development process uses at least one best practice.

> medium
>
> The development process uses several best practices.

> high
>
> There is continuous improvement in place to assess and improve the software development practices used.

**Example** This maturity level can be measure through existing development systems such as DevOps maturity model.

**References** [Kel07] [ARA+22] [AS10]

## 2.2 Automatic Quality Checks

Automatic quality checks determine metrics and calculate KPIs that reflect the code quality.

**Consequence of Negligence** The quality of the code base may deteriorate over time, which can lead to a large number of bugs or hard-to-maintain code.

#### Levels of Maturity

> **low**
>
> Automatically quality checks (e.g., change frequency of each class, cyclomatic complexity of functions) regularly calculate metrics and KPIs from the code.

> **medium**
>
> Thresholds are defined and actions are taken when the thresholds are exceeded.

> **high**
>
> Automatic actions are in place.

**Example** In the implementation phase, it is important to track metrics and define KPIs to obtain a quantitative value for code quality. This can be done in different ways such as mining the change history in the version management system to predict changes or to detect logical coupling.

**References** [GO10]

## 2.3 Code Maintainability

The code has a high degree of maintainability.

**Consequence of Negligence** Changing the code is error-prone and time-consuming. The onboarding of new team members is tedious. Bugs are difficult to fix.

#### Levels of Maturity

> **low**
>
> All code follows the same coding style guidelines (e.g., maximum length of functions).

medium

The code follows clean code guidelines (e.g., descriptive names for functions).

high

Maintainability metrics for the code are monitored and the code is regularly refactored to keep these metrics low.

**Example** When writing code, the development team should follow high standards for code quality. It is well known that high-quality code results in fewer bugs and better maintainability, which can help to avoid extensive code revisions. This influences the economic and technical dimensions of sustainability.

**References** [Kol16] [VCB+18]

## 2.4 Sustainability in Different Process Phases

Sustainability considerations are done in different phases of the process.

**Consequence of Negligence** When sustainable considerations are done in few phases of the process, issues concerning sustainability are treated in isolation resulting in an incoherent understanding of what is a sustainable software development as a whole system.

### Levels of Maturity

low

Sustainability considerations are done in different phases of the process, but there is no documentation that is being updated during each phase.

medium

Stakeholders communicate among each other about sustainable goals and values. A sustainability monitoring process is still in its first stages.

high

Stakeholders communicate effectively among each other about goals, values and principles of sustainability in each phase of the process. Collaborative communication, iterative evolution of incremental and open-ended documentation are key factors implemented in the iterative process of the development.

**Example** A global community of research and practice stakeholders met at a workshop held at the Third Int. Workshop on RE for Sustainable Systems (RE4SuSy), in Karlskrona, Sweden. This group created an "open manifesto for forward-thinking sustainable software design, drafted collaboratively in an open and sustainable process". The Karlskrona manifesto can be used as a basis to effective communicate key goals, values, issues and principles of sustainable design.

**References** [NKDJ15]

## 2.5 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.

**Consequence of Negligence** Inflexibility can hinder the use of the most sustainable solutions available.

**Levels of Maturity**

low

Development team documents the development environment.

medium

Development and business team work together to effectively make the development environment flexible.

high

Development and business teams regularly review the sustainability of the development environment and replace parts for which a more sustainable solution is available.

**Example** Several technology startups start with an inflexible development environment in order to quickly develop a product. As a result, the product may not be sustainable in terms of energy consumption, maintainability, or availability of security updates.

**References** [BV06] [HCH09]

## 2.6 Sustainable Test Management

There is a policy in place to ensure that the right tests are run at the right time

**Consequence of Negligence** Unnecessarily running tests increases feedback times and increases energy consumption.

### Levels of Maturity

low

No policy exists, all tests are executed after every change.

medium

Tests are grouped by the module to which they are associated and they are executed only when the module changes.

high

Tests are associated with fine-grained components such as source code files, and mechanisms exist to execute the tests that correspond only to the modified components.

**Example** Before nightly regressions testing starts, the test server checks each test for changes. If the test is unchanged, it checks if the code that is covered by this test is changed. If neither has been changed, the test is skipped.

**References** [RW85] [RHD00] [MA13]

## 2.7 Sustainability in Release Planning

Sustainability considerations are part of release planning.

**Consequence of Negligence** If the release of the software is done without taking sustainability considerations into account, production and use of information of the software can lead to negative effects on society or the environment.

### Levels of Maturity

> **low**
>
> Stakeholders analyze how the software is currently running and take measures for the next development cycle.

> **medium**
>
> While the software is used and applied, second order effects are analyzed, i.e., a review of what and how the software release is changing situations, factors and creating new capabilities is created.

> **high**
>
> Before releasing the software, measures of consumptions patterns such as energy consumption, and other materials and products consumptions are analyzed and new improvements are to be fixed before the next release.

**Example** Increasing environmental and social implications of ICT infrastructures in the global economy is expanding (According to Hilty et al. (2010)). Thus, it is important to analyze after the release of the software, if it is triggering a change towards a sustainable economy in the long run, or else, under what conditions this might be possible. How can the software help to reduce greenhouse gas emissions? How can new improvements in the new releases can be offered, so the software facilitates adaptation to climate change?

**References** [BBC+15]

## 2.8 Sustainability Quality Attributes

The requirements document explicitly states sustainability of the product as a quality attribute, which is further refined.

**Consequence of Negligence** The software under development may not meet important requirements, which lowers its value. Reworks may be required, which consume resources and delay market introduction.

**Levels of Maturity**

> **low**
>
> Sustainability of the product is a requirement.

> **medium**
>
> The requirements address several dimensions of sustainability.

> **high**
>
> The requirements document is explicit about which dimensions and aspects are covered by the requirements and which are not.

**Example** From a sustainability perspective, it is important that the sustainability of the system under development is a well-documented quality attribute. However, with sustainability being such a broad term, it is important that this quality attribute is refined into a set of concrete and detailed quality attributes that cover individual aspects of sustainability. For example, a data-heavy cloud service could be required to find an efficient solution (in time and space) for data storage and retrieval. As another example, a web-based service application for a municipality could be required to be accessible to everyone in the municipality, which may include people in different age groups or reading skills.

**References** [Rob16] [VLG+14]

## 2.9 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.

**Consequence of Negligence** The sustainability of the software product may decrease in one or more dimensions (e.g., the individual domain if performance is favored over data privacy). The sustainability of the development process may decrease (e.g., if simplicity is favored over extensibility).

### Levels of Maturity

> **low**
>
> For major design decisions, their impact on sustainability is documented.

> **medium**
>
> The architectural design decisions involve strategic consequences (future operations and maintenance effort) based on experience and knowledge. They should be measurable and manageable according to objective criteria. These decisions should be achievable and realistic as well as grounded in domain-specific architecture experience and context.

> **high**
>
> The rationale of each decision shows the impact of the decision on the software development process and the software product.

**Example** Conflicting requirements pose a challenge to software architects, who often need to make trade-offs in their architectural decisions. These decisions should themselves be sustainable. To this end, Zdun et al. devised five criteria that make these decisions more sustainable: they are strategic, measurable and manageable, achievable and realistic, rooted in requirements, and timeless. Furthermore, the architectural design document should be augmented with rationale about how the design decisions impact the sustainability of the software.

**References** [ZCTZ13] [BBC+15] [Rob16]

## 2.10 Sustainable Data Structures

The architect team considers needs of the environment, individuals and society, as well as economic aspects while deciding on which data structures to use. They are optimized with respect to sustainability (e.g., energy consumption or extensibility).

**Consequence of Negligence** A wrong choice of data structures can lead to a high complexity in time and/or space, which consumes unnecessary resources.

### Levels of Maturity

> low
>
> Data structures are designed considering the amount of energy consumption or extensibility properties.

> medium
>
> A benchmark of energy consumption (and/or other important factors considered for the development team) is conducted purely applied to the data structures used in the software development.

> high
>
> New designs and requirements are implemented in the following releases of the software product. This process is iterative and should be applied in a cradle-to-grave approach.

**Example** At Cornell University, an open-source framework for concurrent data-structures and benchmarks called Synch was developed. It provides both high-performant concurrent data-structures and tools for designing and benchmarking these data-structures.

**References** [MDR17] [Kal21]

# 3 Category: Implications of Software Operations

## 3.1 Implications of Software Operations

The effects on sustainability of the operational tools, processes and mechanisms for the software are transparent.

**Consequence of Negligence** The operation of the software may be unnecessarily difficult, expensive, or insecure.

**Levels of Maturity**

> **low**
>
> The effects on the sustainability of the software operation are estimated and documented.

> **medium**
>
> The effects on the sustainability of the software operation are regularly assessed, documented, and improved.

> **high**
>
> Optimizing the effects on the sustainability of the software operation is a continuous target in the development process.

**Example** The carbon and energy footprint of tools used for monitoring and control of the software are known and considered in the overall sustainability characteristics of the system.

**References** [BCD+15]

## 3.2 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.

**Consequence of Negligence** Without measures for resource-intensive operations, the resource consumption is unknown and the software development cannot be claimed to be sustainable.

**Levels of Maturity**

> **low**
>
> The organization has no knowledge about the computing capacity required by different processes.

> **medium**
>
> Categories of computing capacity needs are known which allows a certain amount of planning and cost prediction.

> **high**
>
> A model of computing capacity needs exists allowing the organization to predict needs in detail and optimize the computing resource utilization accordingly.

**Example** The computing needs of a software platform are understood, allowing the organization to reserve computing capacity with their cloud provider, which enables them to benefit from lower cost and preferential energy consumption characteristics.

**References** [SPG09] [WG11]

## 3.3 Development for Efficient Execution

The development process involves optimization steps to reduce the energy consumption of the software under development.

**Consequence of Negligence** Neglecting the opportunity to reduce energy consumption of the software under development leads to an environmentally unsustainable software product.

## Levels of Maturity

**low**

The orders of magnitude of the energy consumption of the software under development are known.

**medium**

There are optimization steps to reduce the energy consumption of the software under development from an architectural point of view.

**high**

A ”white box” approach is implemented to find resource intensive parts of the code and improve them.

**Example** The two most common methods are the black box and the white box method. The black box method uses benchmarking and individual measurement. The white box method is used when inefficiencies in the code need to be detected and improved.

**References** [MPC14]

# 4 Category: Sustainable Team Culture

## 4.1 Sustainable Team Culture

Team members have an understanding of the importance of ethics, social communication and respect for their team members.

**Consequence of Negligence** The project may miss important features or quality attributes. Team performance may degrade if individual stakeholders lose motivation.

**Levels of Maturity**

> **low**
>
> Regular surveys among stakeholders take place.

> **medium**
>
> The results of these surveys are attended by IT management and are analyzed to improve different aspects of social and individual issues that might appear from the information in the surveys.

> **high**
>
> The team itself actively tracks the status of its team culture and continuously takes efforts on improving it.

**Example** When discussing stakeholders and team setup, two dimensions of sustainability play a particularly important role: the social and the individual dimensions. To obtain sustainability in the social dimension, it is important (among other things) that the team has a sense of community and trust. For the individual dimension, personal health and safety is one of the example qualities that must be present in the process. Both dimensions are also covered by the agile manifesto.

**References** [SMDK21] [RO13] [WZZ17]

## 4.2 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.

**Consequence of Negligence** Traveling causes emissions and costs money. It also affects individuals who need to make compromises with their personal lives. Inefficient collaboration weakens the team's performance and possibly its morale.

### Levels of Maturity

> **low**
>
> Team members travel often to ensure efficient operation or to resolve conflicts.

> **medium**
>
> There is little regular travel and the team mostly relies on digital collaboration tools.

> **high**
>
> There is no regular travel. Travel is restricted to improving social aspects of the team (e.g., joint kick-off meetings or celebrations).

**Example** The individual members in a software development process consume energy (e.g., through heating and ventilation of buildings), produce waste (e.g., through phased out laptops), and create emissions (e.g., through commuting). While all of these are important sustainability concerns that must be addressed, they are not specific to software development and thus, they are out of scope of this paper.

However, there are people-related emissions that are specific to software development. Since most software development projects do not involve any physical assets (with cyber-physical systems being a prominent exception), they can be carried out in a virtual fashion with a globally distributed team. Such a team can cross language borders, cultural borders, or time zones. Therefore, it is important the development process involves communication tools and structures that allow the team to collaborate efficiently and with a minimal environmental footprint (which, e.g., could be caused by frequent air travel between development locations).

**References** [GCM17] [HCAF06]

## 4.3 Sustainability Incentive

If the development team acts efficiently and stays under budget, it is not punished by having its budget reduced in the future.

**Consequence of Negligence** The team may lose the motivation to operate more efficiently.

**Levels of Maturity**

> **low**
>
> Teams are encouraged to not make unnecessary expenses, e.g., business trips.

> **medium**
>
> There are no automatic budget adaptations for subsequent years if the team stays under budget.

> **high**
>
> Unspent budget is invested in further sustainability improvements with a focus on the well-being of the employees.

**Example** When motivation gains among the team are improved, coordination gains are also improved. This is a sustainability incentive that enhances positively social and individual dimensions of the development of a software. Because when the group situation is built on, the individual attitude of a member of the team goes in a positive direction and might increase his efforts, initiative and perseverance. From a technical perspective, team members should be aware of how they can contribute to saving cost, e.g., by switching off running VMs in the cloud that are not needed.

**References** [Mik16]

## 4.4 Multidisciplinarity of the Development Team

Different disciplines have been included in the development of the software.

**Consequence of Negligence** A narrowly-focused team can undermine efforts to create solutions with different perspectives

**Levels of Maturity**

> **low**
>
> The involved stakeholders communicate to gain insights of different perspectives. Developers, Architects and Project Managers use the information from the stakeholders and add new requirements or ideas into the development of the software.

> **medium**
>
> A workshop held among stakeholders of different disciplines reunite and participate by providing ideas. Requirements or processes may appear or change for developers, architects or project managers to review.

> **high**
>
> Multidisciplinarity development teams ensure that all the necessary disciplines are involved on the development process.

**Example** Creation of a Stakeholder analysis using the Power Interest Grid (a chart in which we classify all stakeholders by their power over the project and by their interest in the project) is very practical and helps to integrate all stakeholders with different backgrounds and disciplines informed according to their needs and expectations. The goal is to create a group that can complement each other, that occupies different positions, that brings different qualities to the table.

**References** [Fon03] [NKDJ15]

## 4.5 Transparency of Communication

Project documentation, including minutes of meeting, is available to all stakeholders.

**Consequence of Negligence** A centralized group of stakeholders that hold relevant information from the team creates apathy and untrust from the rest of the group; thus, giving up responsibilities on the project.

### Levels of Maturity

> **low**
>
> Meeting minutes should be accessible to all participants.

> **medium**
>
> All stakeholders can access any documentation of the project.

> **high**
>
> The meaning of transparency of communication is defined and documented. Documentation is shared with all stakeholders. Practices and initiatives on how to establish dialogue and critical conscience are implemented and shared with all stakeholders on the team.

**Example** Sustainability requires long term thinking rather than a short period of time. Therefore, participation of stakeholders need to be not only at one stage of the software development, but rather at an iterative process over time. In practice, this participation can be organized in teams representing a diverse group of stakeholders that are aware of the effects on the sustainability of the system software. This group should document and make it available to all. Transparency should be a priority as a long term requirement.

Transparency is considered not only at the beginning of the software process development, but also throughout the entire process of development. As transparency has been implemented as one of the strategy alignments, in this stage transparency becomes a development strategy capable of stimulating a healthy, self-reliant and participative software development community group. This lead that participants know about the progress and problems of the development and can contribute on the growth of the software development with solidarity, and the growth of all stakeholders as whole persons can be reconciled.

**References** [TC02] [TSK+21]

## 4.6 Participation of the Team

The proposals and strategies to improve the sustainability of the software development process of the development team should be heard and be evaluated by a governing body.

**Consequence of Negligence** When participation of the development team is denied or ignored by any entity within the organization, valuable sustainability improvements may be missed.

### Levels of Maturity

**low**

Participation initiatives are encouraged by the organization to its stakeholders involved in the software development.

**medium**

After reviewing the feedback of the stakeholders, a guiding coalition is developed to better communicate the vision of the next sustainable strategies on the software development process.

**high**

Sustainability proposals are discussed as part of the iterative cycle review according to the software development methodology used by the development team. Organizational structure provides flexibility to the stakeholders of the software development to express their opinions in a constructed way without undermining the new initiatives. Big obstacles that prevent a sustainable transformation of the software development processes are confronted and removed.

**Example** A successful sustainable transformation on the software development process starts by involving most of the stakeholders involved in the development as the process progresses. Stakeholders are encouraged to try new approaches, to develop new ideas, and to provide leadership. The only constraint is that the actions fit withing the broad parameters of the overall vision of sustainability. However, communication is never sufficient by itself. To guide the software development to a sustainable approach, often it is required to remove obstacles. These obstacles can be from the perspective point of view of an individual or the organizational structure of the software development team. The Team should work to remove them and not make stakeholders choose between the new sustainable vision and their own self-interest.

**References** [BCD+15]

# 5 Category: Sustainability Awareness

## 5.1 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 development process.

**Consequence of Negligence** Unawareness can lead to decisions that can make the development process unsustainable.

### Levels of Maturity

> **low**
>
> There is at least one key stakeholder with a solid sustainability awareness.

> **medium**
>
> Sustainability is discussed and documented in the project's regular meetings. Project members should be made aware. For long-running projects, new project members need to be briefed.

> **high**
>
> There are documented sustainability goals for the project, which are reviewed at regular intervals meeting.

**Example** Most software development projects involve numerous stakeholders. These stakeholders can be within the development team (e.g., a software architect) our outside (e.g., a customer). Thus, the background and motivation of stakeholders can be very diverse. What all the stakeholders have in common is that each of them makes project-related decisions.

Therefore, all stakeholders have an influence on the sustainability of the software development project. Since their influence can be both positive and negative, it is crucial that each stakeholder is aware of it. Sustainability awareness can be created verbally, e.g., in the kick-off meeting. It should also be manifested in the project documentation. During the project, the stakeholders of the project must have access to necessary sustainability documentation. To this end, many companies have defined sustainability strategies and goals, which often are also publicly available.

**References** [DPP+20] [VLG+14]

## 5.2 Knowledge of Sustainability

The stakeholders of the project possess sufficient knowledge about sustainability.

**Consequence of Negligence** When there is a lack of knowledge about sustainability in the organization, a systematic way to obtain sustainable processes or products is not possible to achieve.

**Levels of Maturity**

> **low**
>
> Stakeholders know general concepts on sustainability and are starting to think about the possibility to align their business models towards sustainable goals.

> **medium**
>
> Key stakeholders (such as product owners) have a solid knowledge on several dimensions of sustainability.

> **high**
>
> All stakeholders involved have a solid knowledge of all five dimensions of sustainability.

**Example** To follow a common approach of documenting sustainable goals and progress into reports, stakeholders should be educated on sustainability. Although existing systems and projects are not to be changed, any new systems and projects would conform the new sustainable goals and standards. This initiative is essential for improving the future systems development capability of the IT organization towards sustainability.

**References** [BCD+15]

## 5.3 Different Sustainability Dimensions

The process has means to investigate sustainability in different dimensions.

**Consequence of Negligence** Even though the process is somewhat sustainable, important aspects of sustainability may be missed. As an example, the process may focus on being environmentally friendly, but its social aspects are neglected.

### Levels of Maturity

**low**

Dimensions of sustainability in a life cycle development process of the software system are understood. Stakeholders work together to design and implement a process in the software development to investigate sustainability in the environmental and social dimensions on the software system.

**medium**

A life cycle assessment (LCA) has been already conducted. A group of stakeholders in the organization is working to align technical, environmental and economic dimensions in the software development of the system.

**high**

A life cycle social assessment LCSA is conducted, software development complies with the relevant ESG rules and regulations concerning the organization.

**Example** It is not enough to include some of the five dimensions of sustainability in the software development life cycle. In the Apple and Conflict Minerals: Ethical Sourcing for Sustainability report written by IBS Center for Management Research (ICMR), Apple Inc (Apple) was criticized for its lack of ethical sourcing for sustainability. Organizations need to know about the Environmental, Social and Governance (ESG) risks and implemented a ESG risk management process in the software development life cycle to create a resilient software system.

**References** [BCD$^+$15] [LKCP15]

## 5.4 Consideration of Different Orders of Effects

Different orders of effects are considered when analyzing the sustainability of the process.

**Consequence of Negligence** Whereas first-order effects may be considered, second-order and third-order effects may be neglected.

**Levels of Maturity**

> **low**
>
> A basic knowledge of the three orders of effects (immediate, enabling, and structural effects) on sustainability is done by a group of stakeholders.

> **medium**
>
> The team is informed by the three orders of effects and workshops are conducted to enhance participation of all stakeholders to define specifically the orders in the software development.

> **high**
>
> An specified ”n” number of sprints should be defined, so every ”n” sprints in the agile process of the software development the three order of effects are reviewed. They are documented and updated.

**Example** Several authors explain the three orders of effects on a software development system. For example, in the sustainability design and software: the Karlskrona Manifesto explains the three orders of effects on a software development process.

**References** [BCD+15] [LKCP15]

## 5.5 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.

**Consequence of Negligence** Sustainability may be neglected in favor of other quality attributes or process metrics (e.g., cost or delivery time). Thus, anticipation of disruptive environmental conditions, proactive management risk and overall the gain of other sustainable operations are missed. As a result, a reduction of ROI can be expected.

### Levels of Maturity

> **low**
>
> Stakeholders recognize that sustainability is a necessity for society, but they prioritize other immediate impacts or risks while developing software.

> **medium**
>
> Stakeholders not only recognize that sustainability is a necessity for society, but they have also analyzed the benefits to adopt a sustainable software development. Qualities of sustainability are considered during or after the design of the system.

> **high**
>
> Stakeholders recognize and analyze the needs of the environment, society, and individuals, while implementing new technologies and considering the economic aspect from the design of the software through its software development life cycle. Qualities of sustainability are considering at the beginning of the creation of the design of the system.

**Example** As a consequence of the rapidly economic development, problems of high energy consumption and environmental pollutions have increased. These problems have disrupted the pace of individuals, societies, and economic grow for several communities. To alleviate these problems, the academic community has conducted substantial research on the mutual substitution between environmental protection and economic growth. For example, Grossman and Krueger conducted an empirical research that showed an inverted U curve relationship between per capita income and pollution levels. The environmental Kuznets curve (EKC) suggests that there may be a positive relationship between environmental protection and economic growth.

**References** [SBL$^+$22] [Bau09]

## 5.6 Usage of tools to assess sustainability

There are tools in use to assess sustainability in a software system development

**Consequence of Negligence** Without proper tools, knowledge of the sustainability of the development of the process may be incomplete or even missing.

**Levels of Maturity**

> **low**
>
> No specialized LCA tools are being used.

> **medium**
>
> Specialized LCA tools are being used occasionally.

> **high**
>
> Specialized LCA tools are being used regularly throughout the process.

**Example** There are several LCA tools that an organization can use to assess its software development process: GaBi (https://sphera.com/gabi-academy/), Simapro (https://pre-sustainability.com/solutions/tools/simapro/), and Open Source LCA (https://www.openlca.org/).

**References** [BCD+15] [Int06]

## 5.7 Availability of Metrics

There are metrics available that allow sustainability to be measured.

**Consequence of Negligence** When sustainability metrics are not known, the size, amount or degree of how sustainable the software system is unknown, as a result governance issues about sustainability may grow and become a problem.

#### Levels of Maturity

**low**

Sustainability metrics are occasionally evaluated.

**medium**

Sustainable software metrics are taken into consideration during the whole life cycle of software development (from cradle to grave).

**high**

An iterative process of searching and implementing new sustainability metrics is embedded in the software development process.

**Example** Several public databases such as Ecoinvent are good examples of researching sustainable metrics. Governments like Switzerland provide more information of Eco-factors according to the Ecological Scarcity Method used in Ecoinvent. Eco-factors measure the environmental damage in eco-points (UBP) per unit of quantity. The ecological scarcity methods make it possible to assess the impact of emissions as part of a life cycle assessment.

**References** [MMS+10]

## 5.8 Energy Consumption of the Development Process

There are policies in place to assess and reduce the energy consumption in software development.

**Consequence of Negligence** Unnecessary energy consumption increases emissions and costs.

#### Levels of Maturity

**low**

Policies and regulations of energy consumption in personal computers, smart phones, and other devices necessary for the development of the software are implemented and documented. These policies are circulated to all the stakeholders involved in the development of the software.

medium

A TPC-Energy benchmark runs recursively in a defined period of time to have an energy-related result of the consumption on the software development life cycle. The switching off process is reviewed and waste of unused resources process is designed and starts.

high

An improved TPC-Energy benchmark which can be run with workloads is implemented. The result of this benchmark will ideally be run with workloads to get a more realistic consumption of energy. With these results, stakeholders are aware of the energy consumption of the system and will take measures to reduce energy consumption. Waste of unused resources process aligns with the circular economy strategy for the company.

**Example** Informing ourselves about what kind of architecture requires less energy help stakeholders to lower the total energy consumption on the communication network. Kern et al. developed procedure models that can be used by stakeholders "to support the optimization of the different processes focusing on green and sustainable software engineering". Also, switching off unused resources process is encouraged.

**References** [CEK+20] [PNV+10]

## 5.9 Sustainability Reporting

Sustainability is a distinct part of project reporting.

**Consequence of Negligence** When there is a lack in the documentation of the progress towards sustainability, the direction in which an organization needs to move might get lost.

**Levels of Maturity**

low

Documentation of sustainable goals are included in the reports of the software development processes.

medium

Sustainability is incorporated into project reports. A fundamental change management analysis that includes sustainable concepts and ideas is implemented while quality, costs and time as well as acceptance, and adjustment has been taken into account.

high

Additionally, an operatives project report where the project phases towards sustainability are documented. Unfreeze, Change and Refreeze stages of the project report are expected to be known.

**Example** The university of Oldenburg developed in 2010 a web interface sustainability reporting software to create and evaluate sustainability reports. According to the description of the project, "most of the reports are still delivered in terms of a monologue, dialogue is rarely going beyond preliminary stakeholder round tables". The aim of the "Corporate Environmental Reporting for Business Related Affiliates" open source software is to enable a dialogue-driven sustainability reporting for all types of companies. (https://uol.de/vlba/lehre/projektgruppen/cerebral)

**References** [SSvBG11]

## 5.10 Continuous Sustainability Improvement

There is a regular assessment of the sustainability of the development process and potential gaps are addressed.

**Consequence of Negligence** The sustainability of the process may decrease over time.

**Levels of Maturity**

**low**

A first sustainable assessment is conducted. Measures are created to ensure that results of the sustainable assessment are analyzed. New rules and regulations are created.

**medium**

New information has been stored provided by the implementation of the rules and regulations. Every two sprints a partial sustainable assessment of the software development process is done.

**high**

Sustainability assessments and improvements are a part of regular retrospectives.

**Example** The U.S. Department of Energy Office of Science among other research groups started a project called Interoperable Design of Extreme-scale Application Software (IDEAS, https://ideas-productivity.org/). The goal of this project is to improve developer productivity and software sustainability while ensuring continued scientific success. In their Exascale Computing project report of January 2020, they developed a productivity and sustainability improvement planning (PSIP) lightweight workflow, "to identify their most urgent software bottlenecks and work to overcome them, as a key aspect of increasing overall scientific productivity". Even though the PSIP framework is not "meant to be an assessment or evaluation tool", it helps software development teams to improve the quality of the project "encouraging frequent iteration and reflection" and "aggregates team capabilities into best practices, introduce the application of appropriate resources, and encourage teams to adopt a culture of process improvement."

**References** [NKDJ15] [BBC+15] [HMB+20]

## 5.11 Willingness to Change Process

There is a willingness to change the process towards higher sustainability.

**Consequence of Negligence** The reluctance to change the processes of the software development towards higher sustainability can result in higher costs, lower quality, unmotivated employees, or operations disruptions.

Levels of Maturity

low

Once the software development process is defined and implemented, changes towards higher sustainability are reviewed. However, relevant stakeholders do not establish a great enough sense of urgency to transform the processes towards sustainability.

medium

If a transformation of a process towards sustainability is required, further changes are accepted, creating short-terms wins.

high

The organization has a unifying strategy, structure, culture and management of processes by relying on governance to consolidate improvements and producing still more changes towards sustainability in an iterative and participative approach.

**Example** Stakeholders who communicate and help people see the right connections towards sustainability anchor changes in the corporation culture. In reality, successful change efforts to change processes in general are full of surprises. However, a simple vision towards sustainability is needed to guide people through an acceptance of changes of processes. This vision can reduce error rate while measuring risks and fulfill official environmental, social and economic requirements.

**References** [LSJMTCR10]

## 5.12 Strong Feedback Loops

The process uses feedback loops to regularly monitor external feedback and feeds it into the process to improve quality.

**Consequence of Negligence** The software under development may not meet important requirements, which lowers its value. Rework may be required, which consume resources and delay market introduction. If feedback is ignored, the motivation of project stakeholders to participate in the project may diminish.

Levels of Maturity

low

Feedback is collected occasionally and made available to the rest of the team.

medium

Feedback is collected in a structured way at regular intervals, analyzed, and forwarded to the corresponding departments.

high

Feedback is collected from external stakeholders, internal stakeholders, and generated from the logs of the software in operation (e.g., exceptions that lead to slow degradation). It is ensured that the feedback is available to the respective relevant stakeholder(s).

**Example** Today, most software development processes follow agile principles, which are summarized in the Manifesto for Agile Software Development. The Manifesto has sustainability "built in" ("Agile processes promote sustainable development") and touches several dimensions of sustainability. Most notably, it puts a strong emphasis on the social dimension ("Business people and developers must work together daily throughout the project.") and the individual dimension ("Build projects around motivated individuals."). This motivates the criterion "Strong Feedback Loops".

**References** [ARC+19] [WMW20] [BBVB+01]

## 5.13 Capacity for Technical Debt Reduction

The software development process must provide capacity for reducing technical debt (e.g., refactoring, updating/complementing documentation).

**Consequence of Negligence** An ever increasing amount of technical debt may negatively impact any further development of the system.

Levels of Maturity

low

No capacity is reserved during development to address technical debt.

medium

Fix amounts of time are reserved for reducing technical debt during each development cycle, but without consideration for the amount of work needed. Often the time reserved for technical debt reduction is used to finish other work.

high

Technical debt reduction is an integral part of the development and planning process. Teams are encouraged and given the flexibility to work on reducing technical debt.

**Example** Dedicate 20 % of the capacity in each sprint to refactoring and maintaining documentation.

**References** [FRP+21]

### 5.14 Willingness to Change Requirements

There is a willingness to change the software system requirements to make the system more sustainable.

**Consequence of Negligence** The inflexibility to change the software system requirements to make the system more sustainable can lead to a short-term orientation of an organization and the corresponding configuration of its resources and competences. As a result, this might decrease its competitive advantages in a changing environment.

#### Levels of Maturity

low

Once the requirements of the software system are baselined, changes are reviewed, but no further process is followed.

medium

If new requirements arrive once the requirements are baselined, further changes can be accepted with a light process.

high

The organization has a unifying development and operations philosophy at the culture, practice and tool levels, to achieve accelerated and more frequent deployment of changes of production.

**Example** Developers have included DevOps in an early involvement of the development of the software system. Agile testing has an acceptance criteria. Architects and developers are confident to include new changes in requirements because their agile testing allows them to make new changes before deployment.

**References** [JL18] [NE00]

## 5.15 Ability to Handle Changing Requirements

The process is able to adapt to changing requirements.

**Consequence of Negligence** Changing any requirement can lead to quality degradation, increased costs, or delays.

**Levels of Maturity**

low

There is no process for identifying changing requirements in advance. The organization responds to changes as they come.

medium

A process that identify which requirements will need to be changed is in place. This process has the goal to develop techniques to improve the changing of requirements without compromising the overall development of the software.

high

A requirements pipeline is in place to constantly identify future requirements and plan for them as part of the system development process.

**Example** Agile software development processes typically handle changing requirements well. Changes can be handled in each sprint planning.

**References** [BHE+10] [BBVB+01]

# 6 Category: Sustainable Procurement and Governance

## 6.1 Sustainable Procurement and Governance

Procurement and governance are driven by sustainability principles.

**Consequence of Negligence** The overall process cannot be sustainable if procurement and governance are not sustainable.

**Levels of Maturity**

low

The supply chain for the development project (e.g., outsourcing) is transparent. An extensive survey program is conducted and rules and regulations are established to measure the level of sustainability in the supply chain of the project.

medium

Rules and regulations are implemented by the governance department to analyze the level of sustainability of the suppliers. Governance department organize in-person visits and utilized the services of third-party investigator to audit the level of sustainability in their supply chain stakeholders.

high

The organizations supports its stakeholders in getting itself certified by independent third party auditors.

**Example** Partnerships between research institutes and organizations can be created to conduct surveys, focus groups and exchange knowledge base and practical experiences to support SMEs and large companies to increase the transparency of the supply chain of a software development.

**References** [WMJS12]

## 6.2 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.

**Consequence of Negligence** When the sustainability of the supply chain of a software product and its development is ignored, the software development and the end-product cannot be sustainable.

### Levels of Maturity

> **low**
>
> The organization recognizes that several infrastructure design decisions affect sustainability.

> **medium**
>
> New decisions are taken into account to improve sustainability.

> **high**
>
> Stakeholders use recommendations provided by the results of the life cycle assessment (LCA) of the infrastructure supply chain. New designs, architecture and requirements are added in the next releases of the software.

**Example** Sustainable infrastructure can be achieved by creating synergies with partners that are responsible for infrastructure and technologies. When searching for a sustainable infrastructure, the following characteristics should be considered in a new provider for the infrastructure of the software development: 1.- The supplier of the infrastructure should manage the lifecycle of its products in a sustainable form (i.e., they can provide a sustainable LCA of its products and services. 2.-The supplier should provide a responsible minerals sourcing report. 3.- The infrastructure of the supplier has eco-design devices, power-efficient and low-carbon properties.

**References** [HO21]

## 6.3 Technologies for System Development

The project uses and/or adopts sustainable technologies for system development.

**Consequence of Negligence** When stakeholders responsible for the project do not adopt sustainable technologies, the system development cannot be sustainable.

**Levels of Maturity**

> low
>
> The development team researches which technologies are sustainable (comply with the five dimensions of sustainability: social, individual, economic, environmental and technical dimensions.

> medium
>
> Development and business stakeholders make a decision of how to implement those technologies aligned with business opportunities.

> high
>
> The organization implements rules and regulations describing: the definition of sustainable technologies and the criteria while choosing a new technology for the software development process.

**Example** Sustainable technologies are not centered in technical and economic dimensions but also in social and individual dimensions. For example, any technology that does not comply with the sustainable development goals (SDGs) cannot be sustainable.

**References** [XZYT19]

## 6.4 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.

**Consequence of Negligence** The lack of regulatory obligations of sustainable digitization on the development process of a software product can lead to the decrease of an organization's value system.

### Levels of Maturity

> **low**
>
> A corporate governance group is formed within the organization. Based on the general framework of sustainability standards (created by the International Sustainability Standards Board, ISSB), the governance group of the organization will start to create the sustainability disclosure standards of the organization to work on its sustainability report.

> **medium**
>
> Corporate governance together with ICT group had an agreement to implement regulatory obligations of sustainable digitalization according to the ten principles of the UN Global compact and aligned the development process of the software accordingly. This will improve transparency in their reporting.

> **high**
>
> The organization has implemented the regulatory obligations that meet fundamental responsibilities in the areas of human rights, labor, environment and anti-corruption on the software development process and its supply chain (according to the United Nation Global Compact).

**Example** In 2022 new sustainability policies and laws came into effect for all business in the European Union. To improve the sustainability of an organization, every company should know about the following key sustainability regulations: 1) The International Finance Reporting Standard (IFRS) created a general framework of sustainability standard, that will be use as a baseline for every company in the European Union to develop a global sustainability reporting. 2) The European Union created an action plan on financing sustainable growth called EU Taxonomy Delegated Act which is intended to provide information on which economic activities are considered sustainable to relevant stakeholders. 3) A new regulation called Sustainable Finance Disclosure Regulation was created to redirecting capital to more sustainable investments and activities. 4) In 2023 a new regulation called Corporate Sustainability Reporting Directive will be in force to replace the Non-Financial Reporting (NFRD).

References [BCD+15]

## References

[ARA+22] Muhammad Azeem Akbar, Saima Rafi, Abeer Abdulaziz Alsanad, Syed Furqan Qadri, Ahmed Alsanad, and Abdulrahman Alothaim. toward successful devops: a decision-making framework. *IEEE Access*, 10:51343–51362, 2022.

[ARC+19] Ademar Aguiar, André Restivo, Filipe Figueiredo Correia, Hugo Sereno Ferreira, and João Pedro Dias. Live software development: Tightening the feedback loops. In *Companion Proceedings of the 3rd International Conference on the Art, Science, and Engineering of Programming*, pages 1–6, 2019.

[AS10] Bob Aiello and Leslie Sachs. *Configuration Management Best Practices: Practical Methods that Work in the Real World*. Pearson Education, 2010.

[Bau09] Rupert J Baumgartner. Organizational culture and leadership: Preconditions for the development of a sustainable corporation. *Sustainable development*, 17(2):102–113, 2009.

[BBC+15] Stefanie Betz, Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin Venters. Sustainability debt: A metaphor to support sustainability design decisions. In *Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy'15)*, 2015.

[BBVB+01] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. The agile manifesto, 2001.

[BCD+15] Christoph Becker, Ruzanna Chitchyan, Leticia Duboc, Steve Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C Venters. Sustainability design and software: The karlskrona manifesto. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 467–476. IEEE, 2015.

[BHE+10] Muhammad Wasim Bhatti, Farah Hayat, Nadeem Ehsan, Azam Ishaque, Sohail Ahmed, and Ebtisam Mirza. A methodology to manage the changing requirements of a software project. In *2010 International conference on computer information systems and industrial management applications (CISIM)*, pages 319–322. IEEE, 2010.

[BV06] Tommaso Buganza and Roberto Verganti. Life-cycle flexibility: How to measure and improve the innovative capability in turbulent environments. *Journal of Product Innovation Management*, 23(5):393–407, 2006.

[CEK+20] Paolo Ciancarini, Shokhista Ergasheva, Zamira Kholmatova, Artem Kruglov, Giancarlo Succi, Xavier Vasquez, and Evgeniy Zuev. Analysis of energy consumption of software development process entities. *Electronics*, 9(10):1678, 2020.

[DPP+20] Leticia Duboc, Birgit Penzenstadler, Jari Porras, Sedef Akinli Kocak, Stefanie Betz, Ruzanna Chitchyan, Ola Leifler, Norbert Seyff, and Colin C Venters. Requirements engineering for sustainability: an awareness framework for designing software systems for a better tomorrow. *Requirements Engineering*, 25:469–492, 2020.

[Fon03] Patrick SW Fong. Knowledge creation in multidisciplinary project teams: an empirical study of the processes and their dynamic interrelationships. *International journal of project management*, 21(7):479–486, 2003.

[FRP+21] Savio Freire, Nicolli Rios, Boris Perez, Camilo Castellanos, Dario Correal, Robert Ramač, Vladimir Mandić, Nebojša Taušan, Alexia Pacheco, Gustavo Lopez, et al. Pitfalls and solutions for technical debt management in agile software projects. *IEEE Software*, 38(6):42–49, 2021.

[GCM17] Paula Graça and Luís M Camarinha-Matos. Performance indicators for collaborative business ecosystems—literature review and trends. *Technological Forecasting and Social Change*, 116:237–255, 2017.

[GO10] Gregor Grambow and Roy Oberhauser. Towards automated context-aware software quality management. In *2010 Fifth International Conference on Software Engineering Advances*, pages 347–352. IEEE, 2010.

[HCAF06] Helena Holmstrom, Eoin Ó Conchúir, J Agerfalk, and Brian Fitzgerald. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In *2006 IEEE International Conference on Global Software Engineering (ICGSE'06)*, pages 3–11. IEEE, 2006.

[HCH09] Michael L Harris, Rosann Webb Collins, and Alan R Hevner. Control of flexible software development under uncertainty. *Information Systems Research*, 20(3):400–419, 2009.

[HMB+20] Michael A Heroux, Lois McInnes, David E Bernholdt, Anshu Dubey, Elsa Gonsiorowski, Osni Marques, J David Moulton, Boyana Norris, Elaine Raybourn, Satish Balay, et al. Advancing scientific productivity through better scientific software: Developer productivity and software sustainability report. Technical report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2020.

[HO21] Eli Hustad and Dag H Olsen. Creating a sustainable digital infrastructure: The role of service-oriented architecture. *Procedia Computer Science*, 181:597–604, 2021.

[Int06] International Organization for Standardization (ISO). ISO/IEC/IEEE environmental management — life cycle assessment — requirements and guidelines. *ISO/IEC/IEEE ISO14044:2006*, pages 1–46, 2006.

[JL18] Shalinka Jayatilleke and Richard Lai. A systematic review of requirements change management. *Information and Software Technology*, 93:163–185, 2018.

[Kal21] Nikolaos D. Kallimanis. Synch: A framework for concurrent datastructures and benchmarks. *Journal of Open Source Software*, 6(64):3143, 2021.

[Kel07] Diane F Kelly. A software chasm: Software engineering and scientific computing. *IEEE software*, 24(6):120–119, 2007.

[Kol16] Henning Grimeland Koller. Effects of clean code on understandability: An experiment and analysis. Master's thesis, University of Oslo, 2016.

[LKCP15] Patricia Lago, Sedef Akinli Koçak, Ivica Crnkovic, and Birgit Penzenstadler. Framing sustainability as a property of software quality. *Communications of the ACM*, 58(10):70–78, 2015.

[LSJMTCR10] Raúl León-Soriano, María Jesús Muñoz-Torres, and Ricardo Chalmeta-Rosaleñ. Methodology for sustainability strategic planning and management. *Industrial management & data systems*, 110(2):249–268, 2010.

[MA13] Sara S Mahmoud and Imtiaz Ahmad. A green model for sustainable software engineering. *International Journal of Software Engineering and Its Applications*, 7(4):55–74, 2013.

[MDR17] Junya Michanan, Rinku Dewri, and Matthew J Rutherford. Greenc5: An adaptive, energy-aware collection for green software development. *Sustainable Computing: Informatics and Systems*, 13:42–60, 2017.

[Mik16] Tommi Mikkonen. Flow, intrinsic motivation, and developer experience in software engineering. *Agile Processes in Software Engineering and Extreme Programming*, 104, 2016.

[MMS$^+$10] Manish Marwah, Paulo Maciel, Amip Shah, Ratnesh Sharma, Tom Christian, Virgilio Almeida, Carlos Araújo, Erica Souza, Gustavo Callou, Bruno Silva, et al. Quantifying the sustainability impact of data center availability. *ACM SIGMETRICS Performance Evaluation Review*, 37(4):64–68, 2010.

[MPC14] Irene Manotas, Lori Pollock, and James Clause. Seeds: A software engineer's energy-optimization decision support framework. In *Proceedings of the 36th International Conference on Software Engineering*, pages 503–514, 2014.

[NE00] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46, 2000.

[NKDJ15] Stefan Naumann, Eva Kern, Markus Dick, and Timo Johann. Sustainable software engineering: Process and quality models, life cycle, and social aspects. In *ICT Innovations for Sustainability*, pages 191–205. Springer, 2015.

[PNV+10] Meikel Poess, Raghunath Othayoth Nambiar, Kushagra Vaid, John M Stephens Jr, Karl Huppler, and Evan Haines. Energy benchmarks: a detailed analysis. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 131–140, 2010.

[RHD00] Gregg Rothermel, Mary Jean Harrold, and Jeinay Dedhia. Regression test selection for c++ software. *Software Testing, Verification and Reliability*, 10(2):77–109, 2000.

[RO13] Sharon Ryan and Rory V O'Connor. Acquiring and sharing tacit knowledge in software development teams: An empirical study. *Information and Software Technology*, 55(9):1614–1624, 2013.

[Rob16] Martin P Robillard. Sustainable software design. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 920–923, 2016.

[RW85] Sandra Rapps and Elaine J. Weyuker. Selecting software test data using data flow information. *IEEE transactions on software engineering*, 4:367–375, 1985.

[SBL+22] Norbert Seyff, Stefanie Betz, Dominic Lammert, Jari Porras, Leticia Duboc, Ian Brooks, Ruzanna Chitchyan, Colin Venters, and Birgit Penzenstadler. Transforming our world through software: Mapping the sustainability awareness framework to the un sustainable development goals. In *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering-ENASE, April, 25-26, 2022, Bristol, UK*, pages 417–425, 2022.

[SMDK21] Dalia Streimikiene, Asta Mikalauskiene, Lina Digriene, and Grigorios Kyriakopoulos. Assessment of the role of a leader in shaping sustainable organizational culture. *Amfiteatru economic*, 23(57):483–503, 2021.

[SPG09] Gunther Steenackers, Flavio Presezniak, and Patrick Guillaume. Development of an adaptive response surface method for optimization of computation-intensive models. *Computers & Industrial Engineering*, 57(3):847–855, 2009.

[SSvBG11] Andreas Solsbach, Daniel Süpke, Benjamin Wagner vom Berg, and Jorge Marx Gómez. Sustainable online reporting model: A web based sustainability reporting software. *Information Technologies in Environmental Engineering: New Trends and Challenges*, pages 165–177, 2011.

[TC02] Lars Thøger Christensen. Corporate communication: The challenge of transparency. *Corporate communications: an international journal*, 7(3):162–168, 2002.

[TSK+21] Akash Tayal, Arun Solanki, Richa Kondal, Anand Nayyar, Sudeep Tanwar, and Neeraj Kumar. Blockchain-based efficient communication for food supply chain industry: Transparency and traceability analysis for sustainable business. *International Journal of Communication Systems*, 34(4):e4696, 2021.

[VCB+18] Colin C Venters, Rafael Capilla, Stefanie Betz, Birgit Penzenstadler, Tom Crick, Steve Crouch, Elisa Yumi Nakagawa, Christoph Becker, and Carlos Carrillo. Software sustainability: Research and practice from a software architecture viewpoint. *Journal of Systems and Software*, 138:174–188, 2018.

[VLG+14] Colin Venters, Lydia Lau, Michael Griffiths, Violeta Holmes, Rupert Ward, Caroline Jay, Charlie Dibsdale, and Jie Xu. The blind men and the elephant: Towards an empirical evaluation framework for software sustainability. *Journal of Open Research Software*, 2(1), 2014.

[WG11] Thomas Wirtz and Rong Ge. Improving mapreduce energy efficiency for computation intensive workloads. In *2011 International Green Computing Conference and Workshops*, pages 1–8. IEEE, 2011.

[WMJS12] Helen Walker, Joe Miemczyk, Thomas Johnsen, and Robert Spencer. Sustainable procurement: Past, present and future, 2012.

[WMW20] Titus Winters, Tom Manshreck, and Hyrum Wright. *Software engineering at google: Lessons learned from programming over time*. O'Reilly Media, 2020.

[WSSR23] Michael Wahler, Norbert Seyff, and Maria Susana Soriano Ramirez. Sustainability Cards - Assessment Criteria for Software Processes. `https://criteria.greensoftware.ch/`, 2023. Accessed: 2023-08-08.

[WSSRed] Michael Wahler, Norbert Seyff, and Maria Susana Soriano Ramirez. Exploring assessment criteria for sustainable software engineering processes. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2024 (to be published).

[WZZ17] Guangdong Wu, Jian Zuo, and Xianbo Zhao. Incentive model based on cooperative relationship in sustainable construction projects. *Sustainability*, 9(7):1191, 2017.

[XZYT19] De Xia, Mengjiao Zhang, Qian Yu, and Yan Tu. Developing a framework to identify barriers of green technology adoption for enterprises. *Resources, Conservation and Recycling*, 143:99–110, 2019.

[ZCTZ13] Uwe Zdun, Rafael Capilla, Huy Tran, and Olaf Zimmermann. Sustainable architectural design decisions. *IEEE software*, 30(6):46–53, 2013.