Blueprints for Sustainability: A Software Architecture Viewpoint on Building Sustainable Systems

Benedikt Engel

Institute of Information Security and Dependability (KASTEL)
Advisor: Dipl.-Inf. Martina Rapp-Sieger, M.Sc. Ralf Sieger

In today's digital era, we face the significant challenge of developing technologies that are not only effective and resource-efficient, but also sustainable and future-proof. However, we frequently encounter applications and architectures that fail to meet the demands of sustainability and the needs of future generations. This issue raises questions that go far beyond the mere technological advancement of software, touching on ecological, social, individual, and economical aspects.

This seminar paper aims to explore the key factors contributing to the lack of sustainability in current technological solutions and introduces solution approaches found in literature, from an architectural perspective. Additionally, it provides an introduction to Green IT, explaining its focus points and presenting approaches to increasing sustainability in IT. Through this exploration, the paper seeks to highlight the importance of sustainable practices in technology development and offer practical strategies for achieving this goal from the literature.

1 Introduction

In our highly digitized society, the extensive use of software applications is integral to our daily lives. As society grows, the demands on these applications increase, leading to complex architectures that must remain available while being cost-efficient, continuously maintained and developed [48].

An estimated 50%-70% of the costs associated with a software application over its lifecycle are allocated to maintenance and development. The architecture of a system largely determines these aspects, since aspects like maintainability, modifiability, resilience, and others depend on it [48]. However, changing design preferences, Architectural Knowledge (AK) Vaporization, and the application of tacit AK can lead to poor architectural decisions, resulting in unmaintainable software and wasted resources [48]. In addition, developing sustainable software architectures is challenging due to the lack of established metrics and frameworks. While the number of metrics to estimate architecture sustainability is increasing [49] it is hard to determine which metrics to use in different contexts [48]. This complicates developers' tasks, making it difficult to consider and measure all facets of sustainable software, especially in complex applications where future impacts are not immediately evident [37].

Data centers and server farms, necessary for the perpetual availability of systems, combined with inefficient data structures and devices, have significant economic and environmental impacts due to high energy consumption [33]. The rebound effect, where improvements in software efficiency lead to higher usage and demand, further exacerbates this issue [45]. Blueprints and frameworks for sustainable architectures or certain parts of an architecture can benefit these issues, as later shown in Chapter 3 and 4.

This seminar paper will provide an overview of the aforementioned topics with the following structure: Chapter 2 provides a classification of sustainable software. Chapter 3 introduces challenges in developing sustainable software and architecture. Chapter 4 presents conceptual solutions for the problems discussed in Chapter 3. Chapter 5 provides an overview of Green IT, offering insights into what Green IT is, what challenges it addresses, and approaches to these challenges. Chapter 6 concludes the work and gives an outlook.

2 Classification of Sustainable Software

This chapter explains key terms frequently used in this work and explores different perspectives on software sustainability.

2.1 Software Architecture

According to Venters et al. "software architectures are the foundation of any software system and provide a mechanism for reasoning about core software quality requirements" [48]. Therefore, it is important to have a common understanding of what software architecture is. In this work, the term *software architecture* follows the description of the IEEE/ISO/IEC 42010:2022 standard, where software architecture is defined as: "[...] Fun-

damental concepts or properties of an entity in its environment and governing principles for the realization and evolution of this entity and its related life cycle processes" [21].

Now that we have a shared understanding of software architecture, the following section will introduce and compare different perspectives on software sustainability.

2.2 Software Sustainability

Software Sustainability itself is a complex and broad concept, which makes it challenging to find a definition that meets all needs. There are two terms often used in this context: Software sustainability and sustainable software. *Software sustainability* is the concept that encompasses the principles, practices and methodologies used to achieve and maintain *sustainable software*. Sustainable software is software that adheres to these principles. Therefore, definitions addressing sustainable software also define software sustainability to a large degree, and will therefore also be taken into account [10]. To better understand these terms, I summarized definitions for software sustainability and sustainable software from the literature:

Definition 1: Koziolek et al. state that "[...] Software systems are sustainable if they can be cost-efficiently maintained and evolved over their entire lifecycle" [24]. The quality of the software architecture plays a crucial role in achieving this [24].

Definition 2: Zdun et al. define Software sustainability as the decisions made during the design process and their ability to remain valid in the future [52].

Definition 3: Becker et al. define that Software "Sustainability is at its heart a systemic concept and has to be understood on a set of dimensions, including social, individual, environmental, economic, and technical" [7].

Definition 4: Penzenstadler et al. state that the concept software sustainability can be seen in two ways: "(1) the software code being sustainable, agnostic of purpose, or (2) the software purpose being to support sustainability goals, i.e. improving the sustainability of humankind on our planet. Ideally, both interpretations coincide in a software system that contributes to more sustainable living" [39].

Definition 5: Dick et al. state that "Sustainable Software is software whose direct and indirect negative impacts on economy, society, human beings, and the environment resulting from development, deployment, and usage of the software is minimal and/or has a positive effect on sustainable development" [13].

There are many different definitions for software sustainability, each emphasizing different key aspects. For example, Definition 2. from Zdun et al. emphasizes the sustainability of design decisions made during the software design process, whereas Definitions 3 and 5 provide a holistic approach by dividing software sustainability into dimensions such as economic, social, individual, and environmental. Definition 3 also includes the technical

aspect, which is not considered in Definition 5. In contrast, Definition 1 focuses solely on the economic aspect of cost efficiency. Definition 4 highlights the importance of the context in which software sustainability can be viewed, indicating that software which does not contribute to sustainability in every dimension introduced in Definitions 3. and 5. can still be considered sustainable.

This work adopts the approach of dividing sustainability into dimensions. The following subsection summarizes what these dimensions are and presents different approaches to defining sustainability in terms of dimensions.

2.3 Dimensions of Software Sustainability

As discussed in Section 2.2, some definitions of software sustainability use a set of sustainability dimensions. The literature presents several dimensions for defining sustainability. Below, I provide an overview and comparison of different approaches.

According to Robert Goodland, sustainability consists of four dimensions — Human (also referred to as individual dimension in literature, e.g., [7, 38]), Social, Economic, and Environmental — which he explains as follows [17]:

1. **Economic sustainability** aims at maintaining capital and adding value.

2. **Environmental sustainability** aims at maintaining renewable and non-renewable resources as well as managing future impacts of human actions.

3. **Individual (Human) sustainability** aims at maintaining human capital which is "health, education, skills, knowledge, leadership and access to services".

4. **Social sustainability** aims at maintaining social capital which is the "Cohesion of community for mutual benefit, connectedness between groups of people, reciprocity, tolerance, compassion, patience, forbearance, fellowship, love, commonly accepted standards of honesty discipline and ethics" according to Robert Goodland.

The social, economic, and environmental dimensions date back to the Brundtland report from 1987 [8]. It is important to note that Robert Goodland's and the Brundtland report's approaches were proposed from a general perspective on sustainability and not specifically for software sustainability. While the set of three dimensions from the Brundtland report is the most broadly used in defining sustainability [10], Penzenstadler et al. emphasize that the three-dimensional concept from the Brundtland report and even the four-dimensional concept proposed by Robert Goodland does not consider the longevity and long-term viability of technical systems, which is important for sustainable software. Therefore, Penzenstadler et al. added the technical dimension to the four dimensions from Robert Goodland and defined it as [38]:

5. **Technical sustainability** aims to make systems long-living, efficiently maintainable, and evolvable in changing environments over time

These sustainability dimensions are not distinct from each other. Condori-Fernandez et al. demonstrated the interdependencies by evaluating the contribution of quality requirements derived by experts to sustainability dimensions, revealing that some quality requirements contribute to multiple dimensions [11]. That means dimensions must share goals addressed by the respective quality requirements. Penzenstadler et al. also proposed a sustainability model to map project requirements to said sustainability dimensions. This model concretizes the abstract approach of sustainability dimensions and makes it more tangible for developers [38] and helps understand possible trade-offs. By trade-offs, I mean situations where a method improves sustainability in one dimension but harms it in another. However, some approaches use just a subset of the five dimensions, add dimensions, or evolve this approach.

Calero et al. stick with the original three dimensions from the Brundtland report. They argue that software sustainability is the same as sustainable software, and therefore the three dimensions also apply to software sustainability [10]. Lago et al. stick with a four-dimensional approach, consisting of the three dimensions based on the Brundtland report and including the technical dimension [26]. Different literature proposes new dimensions, e.g. Aljarallah et al. use five dimensions for software sustainability: environmental, social, economic, technical, and political. They argue that, especially in developing countries, societal guidelines and regulations are needed, justifying a political dimension in software sustainability [4]. A similar direction is found in the work of Al Hinai et al., who propose the inclusion of cultural and religious dimensions. They argue that these aspects are less addressed in the dimensions of the Brundtland report [2].

Various perspectives exist on the dimensions relevant to software sustainability, as seen above. This work adopts the dimensions outlined by Penzenstadler et al., which is widely referenced in the literature (e.g. by the Karlskrona Manifesto [7]). However, the primary focus of this work is on the environmental, economical, and technical dimensions of sustainable software. In addition to understanding software sustainability, it is crucial to explore which attributes characterize sustainable software.

2.4 Quality Attributes of Sustainable Software

Similar to the definition of software sustainability, there are various perspectives on what characteristics sustainable software must have. Calero et al. consider sustainability as non-functional requirement. Non-functional requirements reflect the quality requirements for a software product. Consequently, the quality attributes from the ISO/IEC 25010 [46] Quality model are also connected to sustainable goals [9]. The ISO/IEC 25010 comprises nine quality attributes, each with various sub-characteristics, as illustrated in figure 1 with sustainability as the symbolic tenth quality attribute:

SOFTWARE PRODUCT QUALITY									
FUNCTIONAL SUITABILITY	PERFORMANCE EFFICIENCY	COMPATIBILITY	INTERACTION CAPABILITY	RELIABILITY	SECURITY	MAINTAINABILITY	FLEXIBILITY	SAFETY	SUSTAINABILITY
FUNCTIONAL COMPLETENESS FUNCTIONAL CORRECTNESS FUNCTIONAL APPROPRIATENESS	TIME BEHAVIOUR RESOURCE UTILIZATION CAPACITY	CO-EXISTENCE INTEROPERABILITY	APPROPRIATENESS RECOGNIZABILITY LEARNABILITY OPERABILITY USER ERROR PROTECTION USER ENGAGEMENT INCLUSIVITY USER ASSISTANCE SELF-DESCRIPTIVENESS	FAULTLESSNESS AVAILABILITY FAULT TOLERANCE RECOVERABILITY	CONFIDENTIALITY INTEGRITY NON-REPUDIATION ACCOUNTABILITY AUTHENTICITY RESISTANCE	MODULARITY REUSABILITY ANALYSABILITY MODIFIABILITY TESTABILITY	ADAPTABILITY SCALABILITY INSTALLABILITY REPLACEABILITY	OPERATIONAL CONSTRAINT RISK IDENTIFICATION FAIL SAFE HAZARD WARNING SAFE INTEGRATION	

iso25000.com

Figure 1: Software Quality attributes. Adapted from: ISO/IEC 25010 Quality standard [46]; Calero et al. [9]

However, literature does not clearly define or agree on which sub-characteristics fall under the sustainability quality attribute.

There are other approaches to defining what characterizes sustainable software. E.g., Naumann et al., propose a framework with distinct characteristics necessary for sustainable software products called *the GREENSOFT model* [36], but since the ISO/IEC 25010 standard is an established software Quality standard, this work adheres to it and the argumentation of Calero et al. [9].

3 Challenges in Designing Sustainable Software

This chapter will introduce several challenges in developing sustainable software and software architecture, affecting certain quality attributes and sustainability dimensions.

3.1 Technical Debt and Sustainability Debt

In software architecture, technical debt arises from design decisions affecting coupling and cohesion. Coupling is "the measure of the strength of association established by a connection from one module to another" [47]. Cohesion is "the degree of connectivity among the elements of a module" [47]. This leads to code smells, architectural brittleness, and increased maintenance costs [48]. Therefore, it affects sustainability in the technical and economical dimension according to the definition from Section 2.3, since it increases cost and decreases the lifetime of the system.

3.2 Tacit Architectural Knowledge and Knowledge Vaporization

Another hurdle in designing sustainable systems and architecture is the usage of tacit Architectural knowledge [48]. An example of tacit architectural knowledge is design decisions made based on the experience of the current staff, which remain undocumented. Such knowledge is easily vaporized as soon as the staff leaves. A general explanation for tacit knowledge would be "[...] knowledge that a human is not able to express explicitly, but is guiding the behavior of the human, e.g., how to ride a bike [6]. This challenge affects the social and individual dimension because it leads to a loss of Human and social

Capital by witholding knowledge. Additionally, problems solved in the past are possibly tackled again, leading to the waste of capital and therby affecting the economic dimension according to its definition 2.3.

3.3 Energy Efficiency

The ICT (Information and communications technology) sector's electricity consumption exceeds 10% of global electricity generation, necessitating energy-efficient software to mitigate climate impacts [31, 30, 41, 37]. According to Noman et al. and the definition of the environmental dimension, energy efficiency of systems belongs to the environmental dimension of software sustainability [37].

3.4 Sustainable Design Decisions

Ensuring architectural decisions remain effective over time is crucial for software architectures because it must endure evolution throughout the lifecycle of the system. Challenges include the high effort of comprehensive decision documentation, traceability to other artifacts, and the need for precise justifications [48, 52]. This challenge addresses the technical and social dimension, according to the definition in Section 2.3, since it affects longevity, maintainability and standards among developers.

3.5 Guidelines, Frameworks and Metrics for Sustainability

Balancing immediate impacts with long-term viability requires frameworks and guidelines integrating sustainability into software design [37]. Aside from that, evaluating the sustainability of software architecture and uncover issues early in the design process is crucial. Though, existing established metrics for architecture sustainability remain limited [35]. Research in this field is strongly context-dependent, making it difficult to identify a well fitting set of metrics for the sustainability dimensions and to understand how these dimensions connect among themselves [49].

3.6 Electronic Waste from Mobile Devices

Mobile devices have limited computational resources and battery life [3]. Computationally intensive tasks often necessitate frequent recharging, which degrades batteries and other components, leading to the need for replacements. Our society's tendency to dispose used or slightly damaged devices instead of repairing them exacerbates hazardous electronic waste even further [12, 33]. This waste impacts the environmental dimension of software sustainability, as defined in Section 2.3.

I adress the above presented problem areas by presenting different solution approaches from research in Chapter 4.

4 Conceptual Solutions for Sustainable Software Development

This chapter provides a number of interesting conceptual solutions to tackle problems or partial problems of software sustainability. They provide blueprints for certain aspects of software architecture, tools software architects can use during the design process and software architectures that can be used by developers to increase the sustainability of the project. The approaches are mapped to challenges from Chapter 3 and increase sustainability by tackling these challenges. Since this work focuses mainly on the technical, environmental, and economic dimensions, the proposed concepts will focus on these dimensions.

4.1 The GREENC5 Architecture

Concerning the energy consumption of the ICT sector addressed in Section 3.3, selecting the optimal data structure significantly impacts energy consumption and system performance. This is because every data structure is designed for different tasks and workloads [23, 18, 30]. For example, some data structures are fast at extracting data, while others are at inserting data. For the implementation of an algorithm that has to insert data frequently, a data structure that supports fast insertion is better suited.

Michanan et al. [30] explored the impacts on energy consumption of data structures in the *C5 collection.* This is a C# dynamic data structure library which is used for .NET development. Based on the C5 collection, Michanan et al. developed the GreenC5 architecture, an adaptive dynamic data structure architecture that intelligently changes the data structure implementation depending on the current workload. GreenC5 consists of two main components [30]:

1. **CRUD(Create, Retrieve, Update and Delete)-based Wrapper of C5 Collection:** Dynamically changes the data structure implementation at runtime.
2. **The "Green" Component:** Comprising four subcomponents:
   - **Event Listener and X-Value Translator:** Observes "[...] activities, operation execution and states of the CRUD-based C5 Collection component, and translating them into meaningful feature values for the Classifier".
   - **Classifier:** An Artificial Neural Network trained to classify C5 data structures by energy efficiency for a sequence of operations.
   - **Predictor:** Uses classifier output to predict the next appropriate data structure for upcoming tasks.
   - **Decision Maker:** Decides when to switch to a new data structure based on predictions.

The key distinction from the original C5 collection lies in its internal mechanisms, which imbue it with intelligence, adaptability, and energy awareness [30]. Michanan et al. describe the GreenC5 workflow as follows [30]: First, an energy model is created once to train the classifier. Secondly, the Green component monitors each instance of the adaptive data structure in real-time. When necessary, it notifies the'wrapper to switch to a more

suitable implementation. The implementation to which it will be switched is determined by the predictor based on the Classifier's output.

In conclusion, this approach assists .NET developers in constructing energy-efficient systems by providing an energy-efficient dynamic data structure architecture and library, thereby contributing to sustainability. Michanan et al. suggest that their work should serve as a foundation for integrating similar approaches into other dynamic data structure libraries [30].

Another approach to the challenges from section 3.3 and additionally section 3.6 is Cyberforaging for Surrogate Provisioning.

4.2 Cyber-foraging

Cyber-foraging is used in mobile applications and extends mobile devices capabilities by leveraging external resources such as cloud or local servers, so-called surrogates. There are two main approaches to cyber-foraging: computation offloading and data staging. Computation offloading is the practice of offloading computationally expensive tasks to external mediums. Data staging is the practice of temporarily staging transmitted data on external mediums. Both aim at enhancing battery life and computational power [3]. A significant challenge in cyber-foraging is the volatile nature of operating environments. The connections to external resources can be unstable, unavailable or inappropriate for certain tasks. Surrogate provisioning addresses this issue. Surrogate provisioning involves the deployment and management of surrogates that handle the offloaded tasks or data throughout different architectural tactics [3]. The two main tactics are [3]:

- **Static provisioning:** Surrogates are predefined (pre-provisioning).
- **Dynamic provisioning:** Surrogates are selected and assigned at runtime based on current conditions.

Moghaddam et al. evaluate these surrogate provisioning tactics to show their impact on resilience and energy-efficiency. Dynamic provisioning adapts better to diverse conditions, dynamically switching surrogates in case of errors. Static provisioning offers faster response times and quick reactions to environmental changes, making it more resilient [3]. Moghaddam et al.'s research also demonstrates that cyber-foraging architectures, in general, are more energy-efficient compared to alternative architectures, regardless of the surrogate provisioning approach used [3].

To conclude, Cyber-foraging architectures contribute to sustainability in mobile applications by improving the quality attributes resilience, efficiency, and flexibility. Additionally, it reduces e-waste and device degradation for mobile devices.

During the development of sustainable architectures and software, it is important to make sustainable decisions over the whole development process, which can endure over time. It is equally important to document those decisions to avoid tacit architectural knowledge and knowledge Vaporization, as explained in sections 3.4 and 3.2.

4.3 Sustainable Architectural Design Decisions

To address the challenges from Section 3.2 and 3.4, Zdun et al. introduce five criteria for assessing decision sustainability [52]:

1. **Strategic:** Consider the impacts of decisions during the decision-making process.
2. **Measurable and Manageable:** Limit decision granularity and use numeric criteria to measure outcomes.
3. **Achievable and Realistic:** Pragmatically justify how solutions address problems.
4. **Rooted in Requirements:** Consider domain experience, company context, project requirements, and team capabilities.
5. **Timeless:** Rely on enduring experience and knowledge.

Zdun et al. also introduce solutions for each aspect mentioned in Section 3.2 and 3.4, including the (WH)Y approach, the SOAD (Service-Oriented Architecture Design) decision model, the ADvISE (Architecture Decision Value-driven Identification, Selection, and Execution) framework, and Rational Guidelines which I briefly explain below:

1. **(WH)Y Approach:** Provides a format for minimalistic documentation statements, addressing documentation effort [52].
2. **SOAD Decision Model:** A framework for analyzing and designing service-oriented architectures [53].
3. **ADvISE Framework:** Uses questionnaires to streamline documentation of recurring decisions, reducing effort [52].
4. **Rational Guidelines:** Guidelines for precise decision rationales, avoiding vague statements, and specifying decision drivers based on actual project requirements [52].

This approach benefits software sustainability by ensuring that decisions made during architectural design and development stay valid over time [52]. Additionally, Zdun et al. propose an architectural guidance model for SOAs (Service Oriented Architectures), successfully applied in various projects.

Aside from the importance of sustainable architectural design decisions, it is important to analyze the sustainability of the project to derive sustainable guidelines and uncover issues early, as mentioned in Section 3.5.

4.4 Sustainability Analysis Framework

To address the challenge introduced in Section 3.5, Noman et al. developed the Software Sustainability Analysis Framework (SSAF) to evaluate software sustainability across the development process. While the SSAF does not specifically target software architecture;

it is applicable during both the development and design stage, therefore applicable by software architects. Noman et al. focus on the social, individual, environmental, and technical dimensions but state that the SSAF is also usable with all five dimensions of software sustainability [37].

The SSAF comprises four steps [37]:

- **Step 1: sustainability analysis**
  - **Input**: Software system
  - **Process:** Assess the system using an initial list of parameters of concern (PoCs) to identify potential negative impacts regarding sustainability dimensions early.
  - **Output:** Applicable PoCs (i.e. POCs which are applicable to the input system) for each sustainability dimension.
- **Step 2: impact analysis**
  - **Input**: applicable POCs from step 1
  - **Process:** identify the impacts of unaddressed PoCs the system has on sustainability.
  - **Output:** Impact on sustainability of the system
- **Step 3: goal derivation**
  - **Input**: Impacts on sustainability from step 2
  - **Process:** Derive goals to mitigate these impacts.
  - **Output:** Sustainability goals.
- **Step 4: feature extraction**
  - **Input**: sustainability goals
  - **Process:** Determine features to achieve these goals.
  - **Output:** Software features.

Noman et al. demonstrated the SSAF by deriving sustainability guidelines for the online shopping portal Daraz and Microsoft PowerPoint, providing practical examples of its application [37].

The approach coming closest to the term "Blueprints" are sustainable or sustainability-enforcing or supporting reference architectures that provide a solution to the challenges from sections 3.2, 3.5 and 3.1.

4.5 Sustainable Reference Architectures

Reference architectures serve as foundational software blueprints, encapsulating best practices and knowledge for designing systems within specific domains. They standardize development, facilitate reuse and support system evolution through a clear framework and terminology [50, 34, 15]. The sustainability and robustness of these architectures are vital for addressing broader sustainability dimensions. For example, AUTOSAR [5] in the automotive sector exemplifies how architectural compliance can ensure technical sustainability, even within a changing environment (shift towards electric vehicles) [51, 49]. The necessity for reference architectures extends to other domains, such as agriculture [16] and manufacturing [32]. These reference architectures provide reusable design decisions that help ensure the longevity and adaptability of software systems. By adhering to established design principles and regularly updating documentation to incorporate new requirements, reference architectures support the creation of flexible, resilient software systems that can evolve over time without significant structural changes, thereby contributing to their sustainability [48]. Venters et al. explain that continuous updates are key for the sustainability of reference architectures. Notably, reference architectures supported by strategic consortia, such as AUTOSAR, showcase longevity and adaptability through extensive documentation and regular updates [49].

This chapter explored various conceptual solutions for enhancing software sustainability through architectural approaches, tools and frameworks, focusing on technical, environmental, and economic dimensions. Solutions proposed included the energy-aware GreenC5 architecture for dynamic data structures, cyber-foraging for efficient resource utilization, sustainable architectural decision frameworks, and the SSAF for comprehensive sustainability assessments. Reference architectures further provide blueprints that enforce sustainability principles across diverse domains, ensuring long-term adaptability and resilience in software systems.

5 Green IT

Information Technology (IT) significantly impacts the environment at every stage of a device's lifecycle due to high energy consumption, CO2 emissions, and toxic waste, necessitating a shift towards sustainable practices. Green IT addresses these concerns [33].

5.1 What is Green IT?

Green IT, also called Green Computing, aims to minimize IT's environmental impact while optimizing resource usage. This involves reducing energy use and emissions, managing electronic waste, encouraging recycling and designing eco-friendly technologies. Green IT also aims to align organizational goals with environmental sustainability [33, 29, 44].

5.2 Focus Points of Green IT

According to Murugesan in 2008, Green IT addresses the environmental and economic dimensions, aiming to reduce disposal costs, recycling expenses, ownership costs, and electricity costs. Key areas of Green IT include [33]:

- "design for environmental sustainability
- energy-efficient computing
- power management
- data center design, layout, and location
- server virtualization
- regulatory compliance
- green metrics
- assessment tools and methodology
- environment-related risk mitigation
- use of renewable energy sources
- ecolabeling of IT products"

In more recent works, such as the ones from Soomro et al. from 2012 [44], Saha et al. from 2018 [40] or Dhaini et al. from 2021 [12] those points were adopted to a large part showing that the focus of Green IT research has stayed similar in many regards. However, there are some differences that have emerged over time.

Recent studies emphasize the sustainability of mobile devices and cloud computing due to the increasing number of smartphone users. Compared to 2012, where about 46% of U.S. adults owned a smartphone [43] the number in 2018 had already risen up to 81% according to the Pew Research Center [25]. Therefore, the effect mobile devices have on the environment (e.g., hazardous electronic waste from batteries) [12] is raising more concerns. The high number of smartphone owners also directs the focus of research towards cloud computing for mobile devices (like Cyber-foraging from Section 4.2), which is used to offload heavy tasks from mobile devices onto the cloud to reduce power consumption [12]. Along with this is increasing data center size and the amount of data traffic between users and data Centers, leading to a higher resource usage [12]. Aside from that, there is also increased effort made towards raising awareness about green practices in computer usage in the education sector. Although one could argue that these points are already included in Murugesan's focus points, I think it is important to mention them separately to highlight their increasing importance over time.

5.3 Approaches in Green IT

In the following section, I will briefly present approaches regarding a number of focus points addressing challenges mentioned in Chapter 4 of Green IT introduced in section 5.2.

5.3.1 Energy Consumption and Efficiency in Data Centers

A significant concern in Green IT is the energy consumption and efficiency 3.3, especially of data centers. In 2022, data centers accounted for 4% of the total electricity demand in the U.S. [1]. Examples of strategies to reduce resource utilization in the literature are:

- Hu et al. proposed the Dynamic Time Scale based Server Provisioning (DTSP) method, which optimizes resource utilization by adjusting the number of active servers according to the workload [20].
- The Energy Aware VM Available Time (EAVMAT) algorithm by Loganathan et al. minimizes energy consumption by optimizing ressource usage through job scheduling [28].

According to Friis et al., current research in this area focuses on advanced cooling systems, server virtualization, Data Center Infrastructure Management (DCIM) tools, edge computing, AI-driven data center management, and quantum computing [14]. Edge computing shifts computational processes closer to data sources, reducing latency and optimizing energy consumption [42]. AI frameworks, such as the one proposed by Lin et al., manage power consumption in data centers while maintaining performance [27].

Moreover, it is important to align organizational goals with green goals. This can be done through eco labeling and waste management approaches.

5.3.2 Eco-Labeling and Waste Management

Electronic waste is a rising issue in our society as Eco-labeling means to introduce policies favoring sustainable design, allowing compliant companies to receive an eco-label, thereby encouraging Green IT practices among organizations. Existing eco-labels can be found in the Eco label index [22]. Some companies are addressing e-waste independently. For instance, HP has initiated programs to manage e-waste by offering convenient return options and collecting products for resale and recycling [19].

5.3.3 Green Software and the GREENSOFT Model

The GREENSOFT reference model by Naumann et al. offers a more holistic approach to green software, covering the entire software lifecycle with sustainability metrics, criteria and guidelines for sustainable design and development [36].

By integrating these approaches, Green IT seeks to mitigate the environmental impact of IT operations and leverage technology to support broader sustainability goals, fostering a more sustainable and eco-friendly future.

6 Conclusion

This seminar paper provides an overview of sustainability in software architecture, examining its environmental, technical, social, economic, and individual dimensions. In this context, the connection between the ISO/IEC25010 quality standard and software sustainability has been introduced to gain knowledge about what characterizes sustainable software. Moreover, it addresses current challenges such as technical debt, loss of architectural knowledge, scarcity of sustainable frameworks, and the need for energy-efficient software, presenting their impacts on sustainability. It presented conceptual solutions from an architecture viewpoint, such as an energy-efficient architecture for dynamic data structure libraries, a framework for analyzing software sustainability, and improved decision-making approaches (as summarized in chapter 4) to tackle the aforementioned challenges. The solutions presented were assigned to the sustainability dimensions explained in Chapter 2 through the challenges they address. Emphasizing Green IT, the paper explains the focal points of Green IT as a greater concept and discusses strategies for reducing energy consumption and minimizing the negative environmental impact of IT. The necessity of integrating sustainable practices into software development and organizational structures is underscored, advocating for systems that are sustainable in the long term across multiple dimensions. This should ultimately lead to more sustainable applications that consume fewer resources and remain usable for longer.

However, ambiguities remain. As mentioned in Section 2.4, it is not quite clear what attributes characterize sustainable software. Additionally, it was not clear from the literature which sub-characteristics define the "SUSTAINABILITY" Quality attribute related to the ISO/IEC25010 quality standard, indicating the need for an update of this standard to fully characterize sustainable software. It is also not quite clear how to handle interdependencies between the software sustainability dimensions, as well as the subsequent trade-offs when providing solutions to challenges in a sustainability context. Additonally, the rebound effect is an issue when proposing approaches to increase efficiency like the GreenC5 architecture or Cyber-foraging.

Further research is needed to address these gaps. Given our rapidly evolving society, new challenges will frequently emerge and existing challenges will gain importance. For researchers, obtaining a comprehensive understanding is crucial to identify and address emerging issues effectively, thereby enhancing system sustainability.

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 18.07.2024

. . . . . .
(Benedikt Engel)

References

[1] International Energy Agency. *Electricity 2024 Analysis and forecast to 2026*. URL: https://iea.blob.core.windows.net/assets/6b2fd954-2017-408e-bf08-952fdd62118a/Electricity2024-Analysisandforecastto2026.pdf (visited on 06/13/2024).

[2] Maryam Al Hinai. "Quantification of social sustainability in software". In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 2014, pp. 456–460. DOI: 10.1109/RE.2014.6912298.

[3] Fahimeh Alizadeh Moghaddam et al. "Empirical validation of cyber-foraging architectural tactics for surrogate provisioning". In: *Journal of Systems and Software* 138 (2018), pp. 37–51. DOI: https://doi.org/10.1016/j.jss.2017.11.047.

[4] Sulaiman Aljarallah and Russell Lock. "An exploratory study of software sustainability dimensions and characteristics: end user perspectives in the kingdom of Saudi Arabia (KSA)". In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '18. Oulu, Finland: Association for Computing Machinery, 2018, pp. 1–10. ISBN: 9781450358231. DOI: 10.1145/3239235.3239240.

[5] *AUTOSAR*. URL: https://www.autosar.org/ (visited on 06/15/2024).

[6] Muhammad Ali Babar et al. *Software Architecture Knowledge Management*. Berlin: Springer, 2009. ISBN: 978-3-642-02373-6. DOI: 10.1007/978-3-642-02374-3.

[7] Christoph Becker et al. "Sustainability Design and Software: The Karlskrona Manifesto". In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. 2015, pp. 467–476. DOI: 10.1109/ICSE.2015.179.

[8] Gro Harlem Brundtland, World Commission on Environment, and Development. *Our common future*. Oxford paperbacks. Hier auch später erschienene unveränderte Nachdrucke. Oxford: Oxford University Press, 1987. URL: https://www.are.admin.ch/are/en/home/media/publications/sustainable-development/brundtland-report.html (visited on 06/10/2024).

[9] Coral Calero, Manuel F. Bertoa, and María Ángeles Moraga. "Sustainability and Quality: Icing on the Cake". In: *RE4SuSyRE*. 2013. URL: https://api.semanticscholar.org/CorpusID:15399765 (visited on 05/25/2024).

[10] Coral Calero, Mª Ángeles Moraga, and Mario Piattini. "Introduction to Software Sustainability". In: *Software Sustainability*. Springer International Publishing, 2021, pp. 1–15. ISBN: 978-3-030-69970-3. DOI: 10.1007/978-3-030-69970-3_1.

[11] Nelly Condori-Fernandez and Patricia Lago. "Characterizing the contribution of quality requirements to software sustainability". In: *Journal of Systems and Software* 137 (2018), pp. 289–305. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2017.12.005.

[12] Mahdi Dhaini et al. "Green Computing Approaches - A Survey". In: *Informatica* 45.1 (2021), pp. 1–12. DOI: 10.31449/inf.v45i1.2998.

[13] Markus Dick, Stefan Naumann, and Norbert Kuhn. "A Model and Selected Instances of Green and Sustainable Software". In: *What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 248–259. ISBN: 978-3-642-15479-9. DOI: https://doi.org/10.1007/978-3-642-15479-9_24.

[14] Simon Friis, Line Maria Pyndt Larsen, and Sarah Renée Ruepp. "Strategies for Minimization of Energy Consumption in Data Centers". In: *Proceedings of Twenty-Second International Conference on Networks*. 2023, pp. 17–22. ISBN: 978-1-68558-036-0. URL: https://orbit.dtu.dk/en/publications/strategies-for-minimization-of-energy-consumption-in-data-centers (visited on 06/09/2024).

[15] Matthias Galster and Paris Avgeriou. "Empirically-grounded reference architectures: a proposal". In: *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*. QoSA-ISARCS '11. Boulder, Colorado, USA: Association for Computing Machinery, 2011, pp. 153–158. ISBN: 9781450307246. DOI: 10.1145/2000259.2000285.

[16] Görkem Giray and Cagatay Catal. "Design of a Data Management Reference Architecture for Sustainable Agriculture". In: *Sustainability* 13.13 (2021). DOI: 10.3390/su13137309.

[17] Robert Goodland. "Sustainability: human, social, economic and environmental". In: (2002). URL: https://www2.econ.iastate.edu/classes/tsc220/hallam/TypesOfSustainability.pdf (visited on 06/06/2024).

[18] R. Horvick. *Data Structures Succinctly Part 1*. CreateSpace Independent Publishing Platform, 2017. ISBN: 9781542835688. URL: https://books.google.de/books?id=EcdbnQAACAAJ (visited on 05/27/2024).

[19] *HP Statement on E-Waste & Used Electronic Equipment*. URL: https://h20195.www2.hp.com/v2/getpdf.aspx/c06971467.pdf (visited on 06/14/2024).

[20] Cheng Hu et al. "Improve the Energy Efficiency of Datacenters With the Awareness of Workload Variability". In: vol. 19. 2. IEEE, 2022, pp. 1260–1273. DOI: 10.1109/TNSM.2022.3144508.

[21] "IEEE/ISO/IEC International Standard for Software, systems and enterprise–Architecture description". In: *ISO/IEC/IEEE 42010:2022(E)* (2022), pp. 1–74. DOI: 10.1109/IEEESTD.2022.9938446.

[22] Big Room Inc. *ECOLABEL INDEX*. 2024. URL: https://www.ecolabelindex.com/ecolabels/?st=category=electronics (visited on 06/12/2024).

[23] Todd King. "Dynamic data structures: Theory and application". In: USA: Academic Press Professional, Inc., 1992. ISBN: 0124075304. URL: https://dl.acm.org/doi/abs/10.5555/129369 (visited on 06/01/2024).

[24] Heiko Koziolek. “Sustainability evaluation of software architectures: a systematic review”. In: *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*. QoSA-ISARCS ’11. Boulder, Colorado, USA: Association for Computing Machinery, 2011, pp. 3–12. ISBN: 9781450307246. DOI: 10.1145/2000259.2000263.

[25] Laura Silver Kyle Taylor. *Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally*. 2019. URL: https://www.pewresearch.org/global/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/ (visited on 06/12/2024).

[26] Patricia Lago et al. “Framing sustainability as a property of software quality”. In: *Commun. ACM* 58.10 (2015), pp. 70–78. DOI: 10.1145/2714560.

[27] Xue Lin, Yanzhi Wang, and Massoud Pedram. “A Reinforcement Learning-Based Power Management Framework for Green Computing Data Centers”. In: *2016 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2016, pp. 135–138. DOI: 10.1109/IC2E.2016.33.

[28] Shyamala Loganathan, Renuka Saravanan, and Saswati Mukherjee. “Energy Aware Resource Management and Job Scheduling in Cloud Datacenter”. In: *International Journal of Intelligent Engineering and Systems* 10 (Aug. 2017), pp. 175–184. DOI: 10.22266/ijies2017.0831.19.

[29] Hanuv Mann, Gerald Grant, and Inder Jit Singh Mann. “Green IT: an implementation framework”. In: 121. 2009. URL: https://aisel.aisnet.org/amcis2009/121 (visited on 06/10/2024).

[30] Junya Michanan, Rinku Dewri, and Matthew J. Rutherford. “GreenC5: An adaptive, energy-aware collection for green software development”. In: *Sustainable Computing: Informatics and Systems* 13 (2017), pp. 42–60. DOI: https://doi.org/10.1016/j.suscom.2016.11.004.

[31] Mark P Mills. “The cloud begins with coal”. In: *Digital Power Group* 1 (2013). URL: https://www.tech-pundit.com/wp-content/uploads/2013/07/Cloud_Begins_With_Coal.pdf (visited on 05/28/2024).

[32] Mohsen Moghaddam et al. “Reference architectures for smart manufacturing: A critical review”. In: *Journal of Manufacturing Systems* 49 (2018), pp. 215–225. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2018.10.006.

[33] San Murugesan. “Harnessing Green IT: Principles and Practices”. In: *IT Professional* 10.1 (2008), pp. 24–33. DOI: 10.1109/MITP.2008.10.

[34] Elisa Yumi Nakagawa, Pablo Oliveira Antonino, and Martin Becker. “Reference Architecture and Product Line Architecture: A Subtle But Critical Difference”. In: *Software Architecture*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 207–211. ISBN: 978-3-642-23798-0. DOI: https://doi.org/10.1007/978-3-642-23798-0_22.

[35] Elisa Yumi Nakagawa et al. “Sustainability and longevity of systems and architectures”. In: *Journal of Systems and Software* 140 (2018), pp. 1–2. DOI: https://doi.org/10.1016/j.jss.2018.02.044.

[36] Stefan Naumann et al. “The GREENSOFT Model: A reference model for green and sustainable software and its engineering”. In: *Sustainable Computing: Informatics and Systems* 1.4 (2011), pp. 294–304. DOI: https://doi.org/10.1016/j.suscom.2011.06.004.

[37] Hira Noman et al. “Towards sustainable software systems: A software sustainability analysis framework”. In: *Information and Software Technology* 169 (2024), p. 107411. DOI: https://doi.org/10.1016/j.infsof.2024.107411.

[38] Birgit Penzenstadler and Henning Femmer. “A generic model for sustainability with process- and product-specific instances”. In: *Proceedings of the 2013 Workshop on Green in/by Software Engineering*. GIBSE ’13. Fukuoka, Japan: Association for Computing Machinery, 2013, pp. 3–8. ISBN: 9781450318662. DOI: 10.1145/2451605.2451609.

[39] Birgit Penzenstadler et al. “Systematic mapping study on software engineering for sustainability (SE4S)”. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. EASE ’14. London, England, United Kingdom: Association for Computing Machinery, 2014, pp. 1–14. ISBN: 9781450324762. DOI: 10.1145/2601248.2601256.

[40] Biswajit Saha. “Green computing: Current research trends”. In: vol. 6. 3. ISROSET: International Scientific Research Organization for Science, Engineering and Technology, 2018, pp. 467–469. URL: https://www.researchgate.net/profile/Biswajit-Saha-3/publication/325360535_Green_Computing_Current_Research_Trends/links/5b37051aaca2720785f8fc7a/Green-Computing-Current-Research-Trends.pdf (visited on 06/09/2024).

[41] Cagri Sahin et al. “Initial explorations on design pattern energy usage”. In: *2012 First International Workshop on Green and Sustainable Software (GREENS)*. IEEE, 2012, pp. 55–61. DOI: 10.1109/GREENS.2012.6224257.

[42] Mahadev Satyanarayanan. “The Emergence of Edge Computing”. In: *Computer* 50.1 (2017), pp. 30–39. DOI: 10.1109/MC.2017.9.

[43] Aaron Smith. *Smartphone ownership-2013 update*. 2013. URL: https://www.pewresearch.org/internet/wp-content/uploads/sites/9/media/Files/Reports/2013/PIP_Smartphone_adoption_2013_PDF.pdf (visited on 06/12/2024).

[44] Tariq Rahim Soomro and Muhammad Sarwar. “Green Computing: From Current to Future Trends”. Version 6384. In: (2018). DOI: 10.5281/zenodo.1062818.

[45] Steve Sorrell and John Dimitropoulos. “The rebound effect: Microeconomic definitions, limitations and extensions”. In: *Ecological Economics* 65.3 (2008), pp. 636–649. DOI: https://doi.org/10.1016/j.ecolecon.2007.08.013.

[46] International Organization for Standardization. *ISO/IEC25010*. URL: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (visited on 05/28/2024).

[47] W. P. Stevens, G. J. Myers, and L. L. Constantine. “Structured design”. In: vol. 13. 2. 1974, pp. 115–139. DOI: 10.1147/sj.132.0115.

[48] Colin C. Venters et al. “Software sustainability: Research and practice from a software architecture viewpoint”. In: *Journal of Systems and Software* 138 (2018), pp. 174–188. DOI: https://doi.org/10.1016/j.jss.2017.12.026.

[49] Colin C. Venters et al. “Sustainable software engineering: Reflections on advances in research and practice”. In: *Information and Software Technology* 164 (2023), p. 107316. DOI: https://doi.org/10.1016/j.infsof.2023.107316.

[50] Tiago Volpato et al. “Two perspectives on reference architecture sustainability”. In: ECSA ’17. Canterbury, United Kingdom: Association for Computing Machinery, 2017, pp. 188–194. ISBN: 9781450352178. DOI: 10.1145/3129790.3129815.

[51] Zhongyuan Wang and Dejun Yin. “Design and Implementation of Vehicle Control System for Pure Electric Vehicle Based on AUTOSAR Standard”. In: *2019 22nd International Conference on Electrical Machines and Systems (ICEMS)*. IEEE, 2019, pp. 1–5. DOI: 10.1109/ICEMS.2019.8921856.

[52] Uwe Zdun et al. “Sustainable Architectural Design Decisions”. In: *IEEE Software* 30.6 (2013), pp. 46–53. DOI: 10.1109/MS.2013.97.

[53] Olaf Zimmermann. “An architectural decision modeling framework for service-oriented architecture design”. In: Berlin: Stuttgart, Univ., Diss., 2009, 2009. ISBN: 978-3-86624-438-2. DOI: http://dx.doi.org/10.18419/opus-2665.