VU VRIJE UNIVERSITEIT AMSTERDAM

VU Research Portal

Towards a Sustainability-aware Software Architecture Evaluation for Cloud-Based Software Services

Fatima, Iffat; Lago, Patricia

2023

DOI (link to publisher)
10.1007/978-3-031-66326-0_13

document version
Peer reviewed version

document license
CC0

Link to publication in VU Research Portal

citation for published version (APA)
Fatima, I., & Lago, P. (2023). *Towards a Sustainability-aware Software Architecture Evaluation for Cloud-Based Software Services*. 200-216. https://doi.org/10.1007/978-3-031-66326-0_13

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 05. Oct. 2024

Towards a Sustainability-aware Software Architecture Evaluation for Cloud-Based Software Services*

Iffat Fatima[1 [0000−0002−2430−0441]] and Patricia Lago[1 [0000−0002−2234−0845]]

Vrije Universiteit Amsterdam, The Netherlands
{i.fatima,p.lago}@vu.nl

Abstract. The ubiquity of digital solutions integrating cloud-based software services necessitates sustainability awareness of such solutions. The integration of sustainability needs evaluation early in the software development life-cycle; preferably at the software architecture level. Although extensive literature is available for software architecture evaluation, not so much is observed for sustainability awareness, in general, and specifically for cloud-based software service architectures. In this study, we aim to create a blueprint of a software architecture evaluation method that has the potential to guide the sustainability-aware software architecture evaluation of cloud-based software services. We use the knowledge extracted from existing software architecture evaluation methods from our previous work and identify the steps to develop a generic blueprint for software architecture evaluation. Our blueprint of software architecture evaluation methods consists of 11 general steps divided into 3 phases. We discuss the challenges of software architecture evaluation and provide recommendations; in general and in the context of sustainability. We further present additional sub-steps that supplement the software architecture evaluation blueprint with a sustainability-focused trade-off analysis, impact analysis and prioritization. The software architecture evaluation blueprint can be used and customized for the evaluation of cloud-based software service architectures based on cloud-based software service-specific evaluation criteria (quality attributes/metrics) for sustainability awareness. Future work involves identifying sustainability metrics and evaluating the blueprint in an industrial setting for cloud-based software services.

Keywords: software architecture, sustainability, architecture evaluation, cloud-based software services

1 Introduction

The ubiquity and rapid growth of cloud-based digital solutions reinforce the need to study the sustainability of digital ecosystems. In the context of software, the sustainability needs of these ecosystems should be ensured as part of the Software Development Lifecycle (SDLC).

Designing Software Architecture (SA) is one of the most crucial stages of the SDLC as it lays a foundation for the whole software. Bass *et al.* [5] describe the importance of SA in terms of its role in inhibiting or enabling quality attributes (QAs).

* This publication is part of the SustainableCloud (OCENW.M20.243) project from the research program Open Competition which is (partly) financed by the Dutch Research Council (NWO).

Hence, evaluating SA for the fulfillment of required QAs is the first checkpoint to ensure conformance to quality.

Lago *et al.* [23] create a case for treating sustainability as a property of software quality. In order to ensure the conformance of software to sustainability quality requirements, sustainability awareness needs to be evaluated. Therefore, a SA evaluation method would aid in ascertaining the fulfillment of the sustainability quality of a software system.

The literature includes a number of studies that present a comparative analysis of SA evaluation methods [35, 40]. None of these methods, however, consider sustainability as a criterion for SA evaluation. Koziolek [20] presents a review of SA evaluation methods for sustainability and catalogs architecture-level sustainability metrics. However, some of these metrics are determined by source code analysis and hence, the analysis is at a more granular level. These metrics can directly be mapped to the technical dimension of sustainability, however, further analysis and validation are needed to evaluate how these metrics may contribute to sustainability across other dimensions (social, economic, environmental) and if new metrics are required to fulfill the sustainability requirement.

Our objective is to explore current SA evaluation methods' support in analyzing SA of CBSSs. Building upon our previous work [14], which conducted a systematic literature review of SA evaluation methods. In this study, we leverage its findings and data to build an overarching SA evaluation blueprint for assessing the sustainability of CBSS architectures. Our vision for this blueprint is to assist practitioners in integrating sustainability into the existing SA evaluation process and make informed decisions in the context of the impact of design decisions on sustainability.

The paper is structured as follows: Section 2 presents the background knowledge for our research context. Section 3 presents related works. Section 4 presents our methodology. Section 5 presents the results of the study and the discussion of our findings. Section 6 presents the main threats to the validity of our study and how they are mitigated. Section 7 presents the conclusion with future work.

2 Background

In the context of software engineering research, software sustainability has a lot of variation in terms of its definitions. One of the early mentions of software sustainability in literature appears in a study by Robert *et al.* [37], who introduce a sustainability attribute to measure sustainability in software. The said measurement is proposed in terms of software quality, its potential for evolution and maintainability. Further, Koziolek [20] defines sustainability as a long-living system operating cost-effectively. Penzenstadler *et al.* [33] compare sustainability definitions in the literature and conclude that defining sustainability is relative to the context of the system and the researcher. Calero *et al.* [6] define sustainability along two aspects: sustainability *in* software (when the software itself is sustainable), and sustainability *by* software (when software aids in achieving sustainability through its operation). Lago *et al.* [23] aim to address sustainability with a broader scope, and provide a comprehensive definition of sustainability dimensions embracing the aspects covered in previous definitions under *4D-sustainability* with economic, environmental, social and technical dimensions.

To perform a sustainability assessment of CBSS architectures, it needs to be evaluated across all dimensions considering the over-time impacts (i.e. direct, enabling, and systemic) [16], of design decisions. Several architecture evaluation methods exist that can aid in SA assessment like Architecture Trade-off Analysis Method (ATAM) [18] and Software Architecture Analysis Method (SAAM) [17]. However, these methods do not explicitly provide support for sustainability assessment. Using sustainability criteria as a quality requirement of the system can enable sustainability assessment at the architectural level. In the context of our research, evaluation criteria specific to CBSS architectures need identification. Moreover, the SA evaluation needs to be carried out while making informed trade-offs considering the over-time impacts of design decisions on 4D-sustainability, in the context of CBSS architectures.

3 Related Work

In this section, we present work related to ours, summarising what is missing for the sustainability evaluation of CBSSs.

Comparison of state of the art in SA evaluation.

Several SA evaluation methods are available in the literature for analyzing SA for certain QAs [4, 27, 35, 39, 40]. These works reveal extensive literature on SA evaluation methods. Scenario-based approaches, often rooted in ATAM, are more prevalent. Existing methods offer detailed insights into evaluation steps, artifacts, and the overall process. This information enables the creation of a generic blueprint for SA evaluation, adaptable for assessing sustainability in CBSSs. Most studies focus on a few QAs, but as QAs increase, trade-off decisions become more complex. Thus, the SA evaluation method's capacity for accommodating trade-off analysis with a larger set of QAs needs study, specifically for sustainability. However, there is insufficient information for sustainability-aware SA evaluation.

State of the art in sustainability evaluation of SA. Koziolek *et al.* [21] employ a metric-based SA evaluation concluding that addressing architectural erosion requires analysis by code metrics at the architectural level while requirement and technology-based changes need to be addressed through scenario analysis. Condori-Fernandez *et al.* [10] employ a software sustainability assessment framework (SAF) to evaluate a software product based on a reference architecture. The study identifies, (i) the QAs contributing to four sustainability dimensions, and (ii) the unaddressed QAs that can potentially trigger product evolution. The results of the study show that a criterion is needed, to qualify a QA for its over-time impact (direct, enabling, or systemic). Furthermore, all types of inter-QA effects (positive, negative, and undecided) need evaluation for a well-rounded trade-off analysis.

From the related work, we observe an emerging trend in the literature for the sustainability evaluation of SA. However, the sustainability evaluations are limited to the technical dimension of sustainability. Hence, the conformance to sustainability across other dimensions needs to be studied and appropriate metrics need to be identified. No study evaluates the long-term sustainability impact of architectural decisions on the SA over time.

SA Evaluation of specific architecture types. Some SA evaluation methods are designed specifically for certain SAs. For instance, Nurani *et al.* [30] present

an SA evaluation method specifically for Service Oriented Architecture using a metric-based approach to quantify QAs. Liu *et al.* [26] present a method to evaluate middleware architecture and use a quality rating scale, to prioritize architectural alternatives against priority QAs. Measurements from prototypes of the SA alternatives are used to quantify the QAs. Adjepon-Yamoah *et al.* [1] present cloud-ATAM, an adaptation of ATAM for evaluating cloud-specific QA trade-offs (performance vs availability). They perform a case study on a cloud-based reactive architecture for global software development. However, other cloud-specific QAs need exploration. Based on the particular type of SA, a specific QA and its associated metric may be required for its evaluation.

4 Methodology

In this section, we provide an overview of our research objectives, highlighting the identified problems, the goals we aim to achieve, the research questions we seek to answer, and the methodologies employed for each question.

Problem Statement. Current SA evaluation processes require a mechanism for sustainability inclusion as a criterion for SA evaluation. CBSS architectures need further exploration for applying this evaluation criterion specifically to such SAs.

P1: Lack of uniformity in SA evaluation artifacts: pre-requisite and post-requisites.

P2: Lack of criteria for sustainability-aware SA evaluation.

P3: Lack of SA evaluation methods specialized for CBSS architectures.

Goals. To elaborate on the possibilities of solving this problem, we present the following four goals as a guide for our research methodology:

G1: To identify the requirements for sustainability-aware SA evaluation.

G2: To exploit the available knowledge about SA evaluation methods for sustainability awareness.

G3: To present a blueprint of a SA evaluation method equipped with sustainability-aware SA evaluation of CBSS architectures.

Research Questions. Based on above goals, we define a set of research questions (RQs) and describe the methodology used to answer each RQ.

RQ1: What are the required input and output artifacts for SA evaluation? From the data[1] obtained from our preliminary study, use open coding [41] to extract the elements about the inputs and outputs of SA evaluation methods. This step aims to identify what artifacts are used for SA evaluation and how evaluation results are represented.

RQ2: What are the steps of existing SA evaluation methods? We use axial coding [41] and a visualization-based technique to systematically categorize the steps of the existing SA evaluation methods. We illustrate these steps in activity diagrams[1] and color-code them based on similarity. Upon need, colors are merged to form a single generic category. The steps are grouped into categories per color, where each category represents the steps of the SA evaluation blueprint. A similar, technique is used to identify the order of steps and iteration patterns.

[1] Online Material [https://bit.ly/SAEvalMethods-OnlineMaterial]

RQ3: What are the challenges associated with SA evaluation; in general and in the context of sustainability? For RQ3, we reflect on (i) the results of RQ1 and RQ2, (ii) sustainability literature, and (iii) the discussion of the related works. Based on this reflection we identify limitations and recommendations for developing a sustainability-aware SA evaluation blueprint for CBSSs.

RQ4: How can existing SA evaluation method be tailored for 4D-sustainability evaluation of CBSS architectures? To answer this RQ, we identify the QAs specific to the quality of CBSSs. We run a Google Scholar search with the following search string: `software AND cloud AND ("quality attribute" OR "non-functional requirement")`. As a result of this search query, we filter relevant studies by title and analyze the full text of those studies and identify the quality attributes within those studies, important for CBSSs. The data is available in the Online Material[1]. We use this information as an exercise for our future work, where we aim to develop comprehensive CBSS-specific evaluation criteria. To enable a sustainability-aware evaluation, we leverage the reflections from RQ3 to provide supplementary sub-steps to enable sustainability evaluation as an additional part of the SA evaluation blueprint.

5 Results and Discussion

In this section, we present our findings per each research question (RQ). Based on these findings, we present a blueprint of the SA evaluation method and identify the challenges and recommendations, in general, and specifically in the context of the 4D-sustainability of CBSS architectures.

RQ1: Input & Output Artifacts.

We answer RQ1, in light of the findings from our preliminary work consisting of a systematic literature review [14]. We identify the common input and output artifacts of the SA evaluation methods, their limitations and possible improvements.

Most SA evaluation methods are not clear about the type of input information they need for the initiation of the SA evaluation process. We identify these inputs from the description of SA evaluation methods in the reviewed studies (see Table 1). The completion of the SA evaluation process produces output artifacts, which are normally included in an evaluation report (see Table 2).

Limitations. Many methods skip initial SA evaluation steps and require an input that might be an output of one of the SA evaluation steps. For example, scenarios are produced as a part of the SA evaluation process. However, some methods rely on scenarios as input without performing scenario identification [31]. Output artifacts need to include details of the decision-making process so it can aid the architects in the future to understand the rationale of design decisions before incorporating change requirements.

A structured template for the inputs/outputs would aid in structuring the evaluation process. To make it systematic, inputs/outputs per step could be further specified. Additional reporting could be useful during SA evaluation. E.g., trade-off decisions, their rationale and impacts, and the strategy to handle future changes, if included in the report as an output, would aid architects in better decision-making.

Table 1. Types of inputs used for SA Evaluation

Input Type	**Study ID**	**Input Type**	**Study ID**
SA Description	S1, S5, S7, S8, S11, S17, S20, S21, S22, S30, S33, S36, S37, S39, S42, S44, S48, S55, S62, S64, S66, S69, S70	Fault Domain Model	S30
Scenarios	S5, S6, S7, S10, S12, S15, S21, S34, S37, S38, S41, S42, S50, S51, S65	Architecture Views	S42
Requirements	S24, S25, S28, S39, S44, S45, S48, S50, S53, S64	Context Diagrams	S31
QAs	S6, S24, S35, S36, S46, S59, S60, S68	Usage Profile	S10
Goals	S5, S6, S15, S19, S33, S39, S59	Risks	S15
SA Specification	S2, S16, S26, S29, S43, S52, S57	Usability Profile	S27
UML	S14, S31, S40, S49, S63	SA Drivers	S20
Source Code	S4, S23, S44, S47	SA and SC Models	S13
Quality Concerns	S8, S17, S18	SA Strategies	S65
Documentation	S9, S32	Simulation Model	S67
Design Properties	S56, S57	UI Prototype	S3
SA Candidates	S46, S68	Execution profiles of UML scenarios	S3
Metrics	S11, S54		

SC=Source Code, See Online Material[1] for Study IDs

Table 2. Types of Outputs of SA Evaluation

Output Type	**Study ID**	**Output Type**	**Study ID**
Quantified QAs	S37, S49, S54, S55, S56, S57, S59, S60, S61, S63, S67	Probability of achieving quality	S40, S43
Metric Values	S19,S53,S29,S30,S23,S26	Design goal violations	S4
NFRs/QAs Conformance	S2, S12, S50, S48, S36	Dependency Model	S23
Risks	S1, S6, S38, S58	Improvements	S8
Architectural Approaches	S32, S33, S69, S70	Strong and weak points	S9
Component Interactions	S14, S16, S18	Acceptance Levels	S10
Cost Benefit	S15, S34, S65	SA Rating	S17
Impact of Architectural Decisions	S25, S28, S66	Scenario classification/ Ranking	S7, S39
Risk Factors	S3, S31	Evolvability Guidelines	S20
Architectural Recommendations	S11, S66	Prioritized Architectural Views	S46
Uncertainty Levels	S35, S68	Prioritized ASRs	S45
Evaluation Report	S13, S44	Impact of modifiability	S5
Decision Rationale	S15, S16	Dependency Graph	S47
Scenario Description	S51, S71	Warnings	S52
Evolvability Points	S20	Suitability	S68

NFR=Non-functional requirement, ASR = Architecturally Significant Requirement
See Online Material[1] for Study IDs

RQ2: SA Evaluation Steps

To answer RQ2, we analyze the studies presenting SA evaluation methods, to elicit the evaluation steps and the process organization.

Types of Processes. Our analysis shows that existing SA evaluation methods can be grouped into 3 categories based on the type of process they follow to carry out the evaluation activities.

1. *Sequential.* All evaluation steps are performed in a sequence, one-time each.
2. *Iterative.* All steps or a set of steps, are performed iteratively. An iteration is usually confined to the steps of the analysis phase. This iterative nature of the SA evaluation methods enables the evaluators to ponder over the implications of the design decisions. It further aids them in performing an improvement-driven trade-off- and impact- analysis.
3. *Phase-based.* Such methods divide the evaluation process into multiple phases. Based on the type of method, the steps can be either, (i) *Unique.* All steps are divided into several phases, or (ii) *Common.* The same set of steps is repeated in each phase with a different goal/architectural artifact for each phase. E.g., to analyze Product Line Architectures (PLAs), Olumofin *et al.* [32] use one phase to analyze the core SA and the next phase to analyze the SA of a specific product; using the same set of steps in both phases.

We also observe that certain methods use a combination of iterative and phase-based processes. Iteration can be either of the phases or of the steps within a phase.

SA Evaluation Steps. We identify the SA evaluation steps from the studies presenting SA evaluation methods. These steps are further analyzed in terms of (i) Commonalities in steps, (ii) Phases of evaluation, and (iii) Sequence of steps and iteration patterns. The data of the SA evaluation steps can be found in Online Material[1].

Common Steps. In our analysis, we observe that ATAM [18] is the baseline that, to various extents, other SA evaluation methods follow. Based on the analysis of SA evaluation steps and their comparison with steps of ATAM, we identify 11 common steps and present them as a blueprint for SA evaluation (see Fig. 1).
Phases of Evaluation. Inspired by ATAM [18], we divide the common steps of SA evaluation methods into three phases, (i) Pre-Evaluation (ii) In-Evaluation (iii) Post-Evaluation (see Fig. 1 for the steps within each phase).

Sequence of Steps and Iteration Patterns. In Fig. 1, we illustrate the SA evaluation steps divided into multiple phases. We further represent the iteration and change in the sequence of steps during evaluation.

SA evaluation process naturally starts with a `Pre-Evaluation` phase consisting of two steps; `Preparation` and `Requirement Identification`. However, these steps are found missing in the majority of SA evaluation methods.

Next, comes the `In-Evaluation` phase. The first three steps in this phase are quite consistent in their placement among the sequence of steps in all evaluation methods. These are `Goal Identification`, `Method Presentation` and `Architecture Presentation`. Instead of the `Pre-Evaluation` phase, many methods use at least one or all of the above three steps, as a starting point for the SA evaluation.

In our analysis, two types of `Prioritization` steps are observed.

*(a) Prioritization **of** evaluation criteria.* It refers to an early prioritization of QAs, requirements, or evaluation perspectives (e.g. a specific architectural view). This

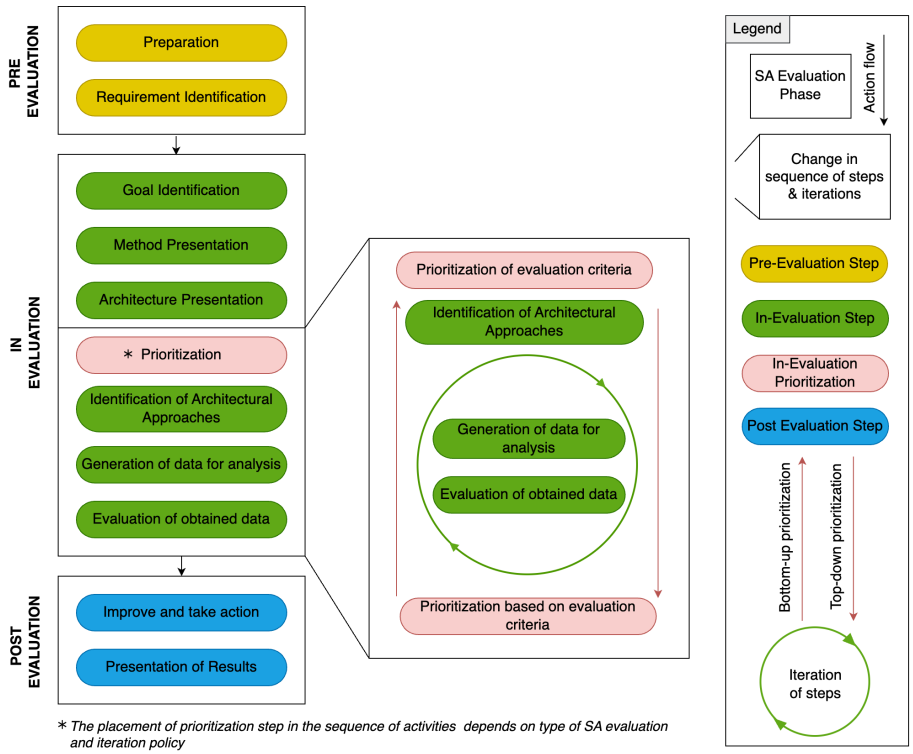

Fig. 1. Blueprint of SA Evaluation Methods

prioritization is performed to narrow down the scope and focus of SA evaluation to high-priority QAs/requirements etc.

(b) Prioritization ***based on*** *the evaluation criteria.* It involves prioritizing architectural decisions for scenario ranking, QA ranking, trade-off decisions, and impact analysis. For example, design decisions with positive effects on high-priority QAs may receive higher priority, or trade-off decisions may be prioritized based on their impact on critical requirements like sustainability.

The position of prioritization steps can oscillate between different steps, on an as-needed basis, facilitating a continuous improvement of the SA.

After presenting the SA, the `Identification of Architectural Approaches` takes place. The architectural approach representation can range from abstract (e.g., textual description, component diagram) to concrete (e.g., sequence diagram, source code). Data is generated from these representations, such as utility trees for scenario-based methods, metric computations for metric-based methods, prototypes for simulation-based methods, and Petri-net generation and execution for formal-modeling-based methods

For `Evaluation of obtained data`, the data can be in the form of scenarios, metric values, etc. The evaluation is based on the (prioritized) requirements and goals. At this stage, some methods perform prioritization *based on* evaluation

criteria. For instance, one design decision may be prioritized over the other *based on* the priority *of* a scenario or a quality requirement.

Once this prioritization is finished, either the evaluation is finalized, or another iteration is performed. This iteration can go back to either generation of data or the evaluation of obtained data. This process continues until all goals have been met or the evaluators have agreed on the achieved trade-offs (see Fig. 1 - Right). For SA evaluation, prioritization can be conducted in one of two ways: (i) *Top-down* - where evaluation criteria are prioritized first, followed by prioritizing design decisions based on the established criteria, or (ii) *Bottom-up* - where the evaluation is initially carried out without prioritizing any specific criterion. Subsequently, design decisions are prioritized based on the analysis results, potentially leading to updates in the priorities of evaluation criteria.

In the `Post-evaluation` phase, analysis results are interpreted and appropriate actions are taken if needed, such as modifying the SA in the `Improve and take action` step. The `Presentation of results` concludes the process, generating a comprehensive report with all necessary output artifacts.

RQ3: Challenges and recommendations

To answer RQ3, we identify challenges for defining an SA evaluation method in general, and for the 4D-sustainability of CBSSs in particular. We further present recommendations that can aid in streamlining the current SA evaluation approaches.

Representation of 4D-sustainability needs to be ensured by equitable stakeholder representation. Our results indicate that most methods overlook ATAM (Architecture Trade-off Analysis Method) [18] Phase 0 steps, namely Partnership and Preparation. These steps involve stakeholder identification, document examination, and training. By skipping this crucial step, stakeholder identification is disregarded in the evaluation process, jeopardizing equitable stakeholder representation. In our state-of-the-art review, we found that most studies only consider technical and business stakeholders in SA evaluation. This raises concerns about the inclusion of other indirect stakeholders [3] during evaluation. This bias in the evaluation process towards technical and economic dimensions undermines sustainability assessment. Neglecting stakeholders from social sustainability domains impedes the comprehensive 4D-sustainability awareness of SA. For example, if software users' concerns are not explicitly analyzed during SA evaluation, it can lead to unfair trade-offs and disregard the impact of design decisions on social sustainability.

An SA evaluation method can be made stakeholder inclusive by explicitly representing stakeholders in the SA evaluation process. 4D-sustainability representation across all four dimensions would also help identify indirect and possibly hidden stakeholders.

Heavy-weight methods are prone to non-acceptance in practice. In agile software organizations, incorporating heavy-weight SA evaluation methods like ATAM, which typically take at least three days for evaluation [18], can hinder fast delivery. Instead, a lightweight and continuous SA evaluation approach, as proposed by Pooley *et al.* [34], would be more suitable. To enhance consistency and structure in SA evaluation, standardizing the SA representation format within a project or organization can facilitate rapid adaptation to changes, while automation tools such as SAVE [19], CSAFE [2] and SDMetrics [30] can help reduce evaluation time by understanding architecture relationships, extracting architectural properties, and conducting metric-based software design analysis, respectively.

Using automation and tool-based solutions for at least some steps of SA evaluation process can aid in accelerating the evaluation process. A systematic SA evaluation process with clear templates and guidelines can also facilitate a fast(er) evaluation.

Choice of abstraction level for quantification is complex. Scenario-based and experience-based methods are carried out at a high abstraction level (typically architecture level). Metric-based methods are carried out at a lower abstraction level (design level or code level) involving definition and collection of metrics [9, 11, 36]. Formal-model-based methods are also carried out at a low abstraction level as they require the representation of architectural components at a granular level (e.g. with ADLs or Petri-nets). Simulation-based methods evaluate the SA using ADLs (e.g. data flow diagram [38], CS2ADEL [13]) or source code [7], also at a lower level of abstraction.

Choosing an abstraction level for SA evaluation is complex. Higher levels risk overlooking important details, while lower levels increase complexity with more elements and dependencies. The challenge lies in finding a middle ground for a quantitative SA evaluation without compromising quality. Hilty *et al.* [16], in turn, emphasize the difference between sustainability analysis at the micro- and macro-levels. They argue against performing sustainability analysis of actions at a micro-level, so to avoid missing the macro-level impact of those actions. In the context of SA, we observe this phenomenon frequently with QA trade-offs (e.g. reliability vs performance). Hence, the impact of micro-level decisions needs to be analyzed for macro-level impact; systematically.

An approach that aggregates micro-level 4D-sustainability to macro-level 4D-sustainability is needed. Micro-level 4D-sustainability can be ensured through sustainability metrics. Macro-level 4D-sustainability can be ensured through Key Performance Indicators (KPIs) and impact analysis of micro-level design decisions. These metrics need aggregation for a KPI representation. Over time sustainability impact needs to be ensured through continuous impact analysis of design decisions.

Lack of sustainability-specific evaluation criteria. The related work shows that only Koziolek [20] presents sustainability metrics that are defined at the code level and categorized based on modularization design principles, corresponding to the technical sustainability. In order to provide a well-rounded view of 4D-sustainability quality, metrics are needed which provide an equitable representation of all sustainability dimensions. Liu *et al.* [25] present two metrics (Impact on the SA - IOSA and Adaptability Degree of SA - ADSA) to quantify the adaptability of SAs. In the context of over-time sustainability, similar metrics can be helpful in identifying the impact of change requirements on SA and the capacity of SA to integrate this change based on the degree of SAs adaptability and evolution.

Identification and measurement of comprehensive 4D-sustainability metrics is needed for evaluating SA; over time and across the economic, environmental, social and technical dimensions.

In the context of over-time 4D-sustainability, architectural decisions resulting in QA trade-offs need to be analyzed for the impacts (i) on other QAs, (ii) on the sustainability dimension itself, (iii) on other sustainability dimensions, and (iv) over time, i.e., direct, enabling and systemic.

RQ4: Sustainability-aware SA evaluation of CBSS architectures

To answer RQ4, we identify CBSS specific QAs and supplement the blueprint with additional sub-steps to ensure sustainability.

Supplementing SA evaluation with CBSS-specific Quality Attributes Lee *et al.* [24] present a difference between conventional software and cloud software-as-a-service (SaaS) through the notion of quality attributes. To evaluate CBSS architectures, we identify cloud-specific QAs by conducting a small-scale systematic search as outlined in Section 4. Our results show that certain QAs are discussed specifically in context for CBSSs, as having a priority (see Table 3). These CBSS-specific QAs can serve as base evaluation criteria for CBSS architectures. Our future work will delve into exploring these QAs through associated metrics for quantification and design principles for ensuring their conformance.

Table 3. Quality Attributes significant for CBSSs

QAs	**References**	**QAs**	**References**
Availability	[24],[28],[42],[12], [24], [1]	Portability	[28][8]
Resource Utilization	[24], [28],[42],[29]	Decentrality	[8],[15]
Functional Correctness	[42]	Recoverability	[24]
Maintainability	[24],[28], [1]	Usability	[42]
Fault Tolerance	[12], [24]	Compliance	[8],[42]
Interoperability	[42],[8],[24], [28]	Configurability	[42]
Security	[12],[42], [1]	Operability	[1]
Scalability	[24],[28],[42], [1], [1]	Elasticity	[22]
Response Time	[24], [28], [42], [29]	Reliability	[1]

Supplementing SA evaluation with sustainability. To supplement sustianability, we propose to specify the *Evaluation of obtained data* step (in Fig. 1) with 3 sub-steps (see Fig. 2). The goal of introducing this supplementary support is to make these steps explicit to ensure sustainability evaluation. The steps are (i) Trade-off analysis, (ii) Prioritization of trade-offs based on evaluation criteria, and (iii) Impact analysis. In these steps, the chosen design decisions are analyzed for possible trade-offs. The choice of selecting an appropriate trade-off is carried out using a prioritization mechanism. Next, impact (direct, enabling and systemic) of the chosen trade-off, on sustainability is identified. This impact analysis is further classified into 3 types. (i) *Inter-QA impact* of trade-offs between CBSS-specific prioritized QAs, (ii) *Inter-dimension impact* of QA trade-offs across four sustainability dimensions, (iii) *Intra-dimension impact* of QA trade-offs within one sustainability dimension. Prioritization is a crucial sub-step here as the results of impact analysis may indicate requiring a change in prioritization of QAs or their trade-offs.

This process continues in an iterative cycle until all design decisions have been agreed upon by mutual consensus of the involved stakeholders. Although ATAM already provides a trade-off analysis mechanism, it lacks an explicit mechanism for impact analysis and continuous improvement. This mechanism is provided by adding these supplementary sub-steps to the blueprint.

In our future work, we aim to apply the SA evaluation process (see Fig.1) in the context of CBSS architectures to validate the method for its applicability.

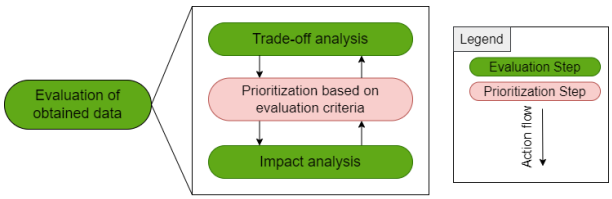

Fig. 2. Supplementary sub-steps for SA Evaluation

6 Threats to Validity

In this section, we discuss threats to validity and the employed mitigation strategies.

Construct Validity. The application of open coding and axial coding yields the risk of subjective bias. To mitigate this risk, the categorization is cross-reviewed by all authors. In case of inconsistencies, consensus is established through discussion, with eventual adjustments to the blueprint.

Reliability. The initial data for analysis was obtained through a SLR [14]. To ensure objectivity, we utilize a systematic approach to categorize and analyze SA evaluation steps, identifying common patterns and generating a blueprint. The Online Material[1] includes the data used and visualizations for categorization.

Internal Validity. To minimize internal bias, the evaluation begins with an analysis of general SA evaluation methods, which is then contextualized within the framework of sustainability and CBSS architectures. Cross-validation by the co-author is conducted to ensure objectivity and prevent any subjective bias.

External Validity. We introduce a generic blueprint for SA evaluation, which is then enhanced with additional steps for sustainability-aware SA evaluation. Additionally, we identify CBSS-specific QAs for evaluating CBSS architectures. The presented blueprint is applicable to various software architectures regardless of the evaluation criteria, allowing for its use with different domains by incorporating specific QAs (e.g., to evaluate the impact of design decisions on a specific QA for mobile applications).

7 Conclusion and Future Work

In this paper, we build upon and leverage the state-of-the-art in SA evaluation methods, and create a blueprint that has the potential to guide SA evaluation. We (i) identify the inputs, outputs and common steps of SA evaluation methods to create a reusable blueprint (cf. RQ1-2); (ii) identify the challenges and present recommendations for a sustainability-aware SA evaluation process (cf. RQ3); (iii) supplement the blueprint with additional sub-steps for trade-offs, impact analysis and prioritization (cf. RQ4); and (iv) present preliminary results towards the evaluation criteria for CBSS architectures through CBSS-specific QAs (cf. RQ4).

In our future work, we will validate the applicability and usefulness of our blueprint. To this aim, we will conduct a case study on a CBSS-based system in the industry. We further aim to (i) identify and use evaluation criteria in terms of 4D-sustainability metrics; (ii) identify the impact of these metrics on 4D-sustainability in terms of possible trade-offs or ripple effects; and (iii) represent the sustainability-quality of CBSS architectures at a macro-level through 4D-sustainability indicators.

References

1. D. E. Adjepon-Yamoah, "cloud-ATAM: Method for Analysing Resilient Attributes of Cloud-Based Architectures," in *Software Engineering for Resilient Systems*. Springer, 2016.
2. N. Admodisastro and G. Kotonya, "An Architecture Analysis Approach for Supporting Black-Box Software Development," in *Software Architecture*. Springer, 2011.
3. R. Alidoosti, P. Lago, E. Poort, M. Razavian, and A. Tang, "Incorporating Ethical Values into Software Architecture Design Practices," in *IEEE 19th Int. Conf. on Software Architecture Companion*, 2022.
4. M. A. Babar and I. Gorton, "Comparison of scenario-based software architecture evaluation methods," in *11th Asia-Pacific Software Engineering Conference*. IEEE, 2005.
5. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 4th ed. Addison-Wesley Professional, 2021.
6. C. Calero, M. Moraga, and F. Garcia, "Software, Sustainability, and UN Sustainable Development Goals," *IT Professional*, vol. 24, no. 1, 2022.
7. E. Cavalcante, J. Quilbeuf, L.-M. Traonouez, F. Oquendo, T. Batista, and A. Legay, "Statistical model checking of dynamic software architectures," in *European Conf. on Software Architecture*. Springer, 2016.
8. M. A. Chauhan and M. A. Babar, "Cloud infrastructure for providing tools as a service: Quality attributes and potential solutions," in *Proc. of the WICSA/ECSA Companion Volume*. ACM, 2012.
9. H. B. Christensen, K. M. Hansen, and B. Lindstrøm, "Lightweight and continuous architectural software quality assurance using the aSQA technique," in *Software Architecture*. Springer, 2010.
10. N. Condori-Fernandez, P. Lago, M. R. Luaces, and A. S. Places, "An action research for improving the sustainability assessment framework instruments," *Sustainability*, vol. 12, no. 4, 2020.
11. U. Dayanandan and K. Vivekanandan, "An empirical evaluation model for software architecture maintainability for object oriented design," in *Proc. of the Int. Conf. on Informatics and Analytics*. ACM, 2016.
12. S. Devata and A. Olmsted, "Modeling Non-Functional Requirements in Cloud Hosted Application Software Engineering," in *The 7th Int. Conf. on Cloud Computing, GRIDs, and Virtualization*. IARIA, 2016.
13. M. Dias and M. Vieira, "Software architecture analysis based on statechart semantics," in *10th Int. Workshop on Software Specification and Design.*, 2000.
14. I. Fatima and P. Lago, "A Review of Software Architecture Evaluation Methods for Sustainability Assessment," in *20th Int. Conf. on Software Architecture Companion (ICSA-C)*. IEEE, 2023.
15. S. P. Gochhayat, S. Shetty, R. Mukkamala, P. Foytik, G. A. Kamhoua, and L. Njilla, "Measuring decentrality in blockchain based systems," *IEEE Access*, vol. 8, 2020.
16. L. M. Hilty and B. Aebischer, "ICT for Sustainability: An Emerging Research Field," in *ICT Innovations for Sustainability*. Springer, 2015.
17. R. Kazman, L. Bass, G. Abowd, and M. Webb, "SAAM: a method for analyzing the properties of software architectures," in *Int. Conf. on Software Engineering*. IEEE, 1994.
18. R. Kazman, M. Barbacci, M. Klein, S. J. Carrière, and S. G. Woods, "Experience with performing architecture tradeoff analysis," in *Int. Conf. on Software Engineering*. IEEE/ACM, 1999.
19. J. Knodel, M. Lindvall, D. Muthig, and M. Naab, "Static evaluation of software architectures," in *Conf. on Software Maintenance and Reengineering*. IEEE, 2006.
20. H. Koziolek, "Sustainability Evaluation of Software Architectures: A Systematic Review," in *7th Int. Conf. on the Quality of Software Architectures and 2nd International Symposium on Architecting Critical Systems*, ser. QoSA-ISARCS '11. ACM, 2011.

21. H. Koziolek, D. Domis, T. Goldschmidt, and P. Vorst, "Measuring architecture sustainability," *IEEE Software*, vol. 30, no. 6, 2013.
22. M. Kuperberg, N. R. Herbst, J. von Kistowski, and R. H. Reussner, "Defining and quantifying elasticity of resources in cloud computing and scalable platforms," in *Karlsruhe Reports in Informatics*, 2011.
23. P. Lago, S. A. Koçak, I. Crnkovic, and B. Penzenstadler, "Framing Sustainability as a Property of Software Quality," *Commun. ACM*, vol. 58, no. 10, 2015.
24. J. Y. Lee, J. W. Lee, D. W. Cheun, and S. D. Kim, "A quality model for evaluating software-as-a-service in cloud computing," in *7th ACIS Int. Conf. on Software Engineering Research, Management and Applications*, 2009.
25. X. Liu and Q. Wang, "Study on application of a quantitative evaluation approach for software architecture adaptability," in *5th Int. Conf. on Quality Software*, 2005.
26. Y. Liu, I. Gorton, L. Bass, C. Hoang, and S. Abanmi, "MEMS: A Method for Evaluating Middleware Architectures," in *Quality of Software Architectures*. Springer, 2006.
27. M. Mattsson, H. Grahn, and F. Mårtensson, "Software Architecture Evaluation Methods for Performance, Maintainability, Testability, and Portability," in *Int. Conf. on the Quality of Software Architectures*, 2006.
28. P. Nadanam and R. Rajmohan, "QoS evaluation for web services in cloud computing," in *Third Int. Conf. on Computing, Communication and Networking Technologies*, 2012.
29. L. Nogueira, A. Barros, C. Zubia, D. Faura, D. Gracia Pérez, and L. Miguel Pinho, "Non-functional requirements in the elastic architecture," *Ada Lett.*, vol. 40, no. 1, 2020.
30. A. Nuraini and Y. Widyani, "Software with service oriented architecture quality assessment," in *Int. Conf. on Data and Software Engineering (ICODSE)*, 2014.
31. B. Ojameruaye, R. Bahsoon, and L. Duboc, "Sustainability Debt: A Portfolio-based approach for evaluating sustainability requirements in architectures," in *38th Int. Conf. on Software Engineering Companion (ICSE-C)*. IEEE/ACM, 2016.
32. F. G. Olumofin and V. B. Mišić, "A holistic architecture assessment method for software product lines," *Information and Software Technology*, 2007.
33. B. Penzenstadler, "Towards a Definition of Sustainability in and for Software Engineering," in *Proc. of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013.
34. R. Pooley and A. Abdullatif, "CPASA: Continuous Performance Assessment of Software Architecture," in *2010 17th IEEE Int. Conf. and Workshops on Engineering of Computer Based Systems*, 2010.
35. M. Sahlabadi, R. C. Muniyandi, Z. Shukur, and F. Qamar, "Lightweight software architecture evaluation for industry: A comprehensive review," *Sensors (Basel)*, vol. 22, no. 3, 2022.
36. C. Sant'Anna, E. Figueiredo, A. Garcia, and C. J. P. Lucena, "On the Modularity of Software Architectures: A Concern-Driven Measurement Framework," in *Software Architecture*. Springer, 2007.
37. R. Seacord, J. Elm, W. Goethert, G. Lewis, D. Plakosh, J. Robert, L. Wrage, and M. Lindvall, "Measuring software sustainability," in *Int. Conf. on Software Maintenance. ICSM. Proc.*, 2003.
38. L. Sion, D. Van Landuyt, K. Yskout, and W. Joosen, "SPARTA: Security & Privacy Architecture Through Risk-Driven Threat Assessment," in *Int. Conf. on Software Architecture Companion (ICSA-C)*. IEEE, 2018.
39. R. C. Soares, R. Capilla, V. dos Santos, and E. Y. Nakagawa, "Trends in continuous evaluation of software architectures," *Computing*, 2023.
40. D. Sobhy, R. Bahsoon, L. Minku, and R. Kazman, "Evaluation of software architectures under uncertainty," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 4, 2021.
41. A. Strauss, J. M. Corbin, and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, Inc., 1998.
42. M. Younas, D. N. A. Jawawi, M. A. Shah, A. Mustafa, M. Awais, M. K. Ishfaq, and K. Wakil, "Elicitation of nonfunctional requirements in agile development using cloud computing environment," *IEEE Access*, vol. 8, 2020.