Variability Features: Extending Sustainability Decision Maps *via* an Industrial Case Study

Markus Funke, Patricia Lago, Roberto Verdecchia
Vrije Universiteit Amsterdam, Netherlands

***Abstract*—Over the years, various thinking frameworks have been developed to address sustainability as a quality property of software-intensive systems. Notwithstanding, which quality concerns should be selected in practice that have a significant impact on sustainability remains a challenge.**

In this experience report, we propose the notion of variability features, *i.e.,* specific software features which are implemented in a number of possible alternative variants, each with a potentially different impact on sustainability. We extended sustainability decision maps to incorporate these variability features into an already existing thinking framework. Our findings were derived from a qualitative case study and evaluated in an industrial context. Data was collected by analysing a real-world application and conducting working sessions together with expert interviews.

The variability features allowed us to identify and evaluate alternative usage scenarios of one real-world software-intensive system, enabling data-driven sustainability choices and suggestions for professional practices. By providing concrete measurements, we can support software architects at design time, and decision makers towards achieving sustainability goals.

***Index Terms*—Software Architecture, Software Sustainability, Case Study, Variability Features, Lessons Learned.**

I. Introduction

Now that the energy consumed by software is exceeding the one of the entire avionic sector [1], the sustainability impact of software-intensive systems is no longer negligible. To address this issue, during the years numerous approaches have been proposed to improve the sustainability of software products. Related literature tackles software sustainability from different angles. While some works focus on refactoring source code to improve energy efficiency [2]–[4], others design architectural tactics to address software sustainability more holistically [5], [6]. Despite such efforts, how to concretely select what sustainability concerns to address in practice remains an open question. As a potential solution, in recent years, thinking frameworks were devised to reason about the sustainability of software-intensive systems, fostering communication on the sustainability of software products, and highlighting the most important (or most promising) sustainability concerns to be addressed case by case. In this work, we report on the experience of applying Decision Maps (DMs), a software sustainability thinking framework proposed by Lago [7], to an industrial software product developed by our industrial partner. The application of DMs, conducted in collaboration with the company, aimed to empowering our collaboration partner to gain insights into (i) the sustainability of their software product, (ii) the measurement plans which could be used to measure software sustainability, and (iii) the strategies that could be devised to improve it. While environmental concerns such as energy consumption are only one example, in this work we also cover the broader scope of sustainability in all four dimensions (*i.e.,* technical, environmental, social and economic [7]). The main **contributions** of this study are: (i) the introduction of the notion of variability features and variants; (ii) an extension to sustainability decision maps; (iii) the application and evaluation in an industrial context; (iv) suggestions and strategies for professional practice.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 871342.

II. Study design and execution

In this section we describe the research methodology used to apply the DMs in the context of the industrial partner collaborating on the project. An overview of the research process used for this study is depicted in Figure 1, and further described below.

Working session: In the first phase of the research process, a working session was held between the researchers who authored this paper and the industrial partner. During this session, the knowledge on DM of the researchers, and the knowledge on the software product under consideration, named Zahori[1], was combined to model a DM. Specifically, the industrial participants provided information on the Zahori product, and the researchers modeled such information into the DM, inquiring about further data, clarifications, and confirmations when needed. As background information about the context, Zahori is a robotic process automation tool used in software testing, capable of capture-and-replay testing, variable customization, and log reporting.

In contrast to previous work [8], where the knowledge on DMs was first taught to industrial participants, who could then independently apply it, the methodology used in this study allowed to efficiently construct a DM, while ensuring both the completeness of the information modeled, and the syntactic/semantic correctness of the DM. The output of this phase was a DM of Zahori, capturing its main features, sustainability concerns, and related dependencies (see Section III-A for the concrete DM and its documentation).

Manual code inspection: In the second phase, the source code of the product under consideration (Zahori), which is available as open-source[1], was manually inspected

[1]https://github.com/zahori-io/. Accessed: 05-12-2022.

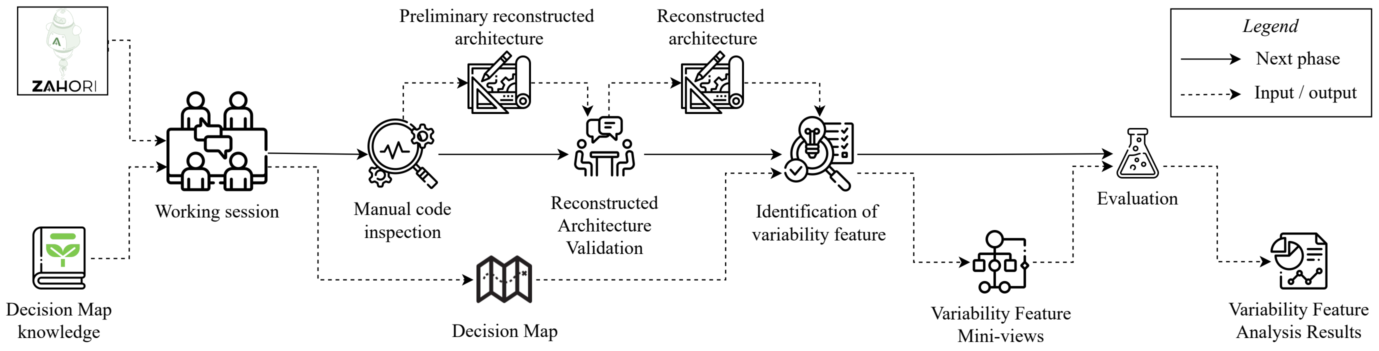

Legend
Next phase
Input / output
ZAHORI
Preliminary reconstructed architecture
Reconstructed architecture
Working session
Manual code inspection
Reconstructed Architecture Validation
Identification of variability feature
Evaluation
Decision Map knowledge
Decision Map
Variability Feature Mini-views
Variability Feature Analysis Results

Fig. 1: Study design overview

in order to reconstruct its architecture. During this phase, three researchers manually analyzed the Zahori code base to identify the architecture elements and their connections. The output of this phase was a preliminary version of the reconstructed architecture of Zahori, to be validated with the developers of Zahori in the subsequent phase.

Reconstructed architecture validation: In the third phase of the study, a series of interviews and follow-up questions were conducted with the developers of Zahori. This process was executed to validate the reconstructed architecture and terminated with a documentation of the reconstructed architecture of Zahori, as obtained via the manual inspection and subsequent validation phase. An overview of the resulting reconstructed architecture is reported in Section III-B.

Identification of Variability Feature: In this phase, both the reconstructed architecture and the DM were utilized to identify where in the Zahori architecture the different sustainability aspects of the DM can be measured. We exemplify this identification process by focusing on a specific Zahori sub-feature having a potential impact on sustainability. The sub-feature selection was driven by both the envisioned impact of the metric, *i.e.,* its relevancy; and the effort required to implement its measurement plan, *i.e.,* its measurability. The process and the sub-feature itself are outlined below.

First, we mapped the involved sub-elements of the DM and the reconstructed architecture to each other. This process was conducted to identify, in the architecture, where the sustainability concerns could be measured and evaluated. In addition to the mapping of architecture elements to DM ones, metrics suitable to assess the sustainability concerns were identified as derived product.

Second, we identified one concrete *variability feature*, *i.e.,* a specific feature modeling a variation point[2] in the DM which has already been implemented in a number of possible alternative variants, each with a potentially different impact on the sustainability of the application at hand.

Based on the results of the working session, the derived DM and its analysis (described further in Section III-C), we selected the Zahori feature for creating ‘evidences’ as a potential variability feature, which is simply referred to as ‘Evidences’ feature from this point onward. The Evidences feature provides Zahori the capability to report to the tester evidences on the end status of tests, in the form of either logs, screenshots, textual documents, or videos.

The final output of this phase were two *mini-views*[3] zooming into the selected variability feature. In particular, the first mini-view focuses on the mapping between the feature-related DM elements and architecture elements, and the second mini-view visualises the variability feature together with its variants and sustainability concerns.

Evaluation: Finally, we conducted an evaluation in terms of a controlled experiment on the impact that varying the selected Evidences feature entailed on the related sustainability concern (as mapped in the mini-view DM). The output of this phase is empirical evidence on the impact that each variant of the variability feature has on the sustainability concerns. An overview of the inspection of the variability feature is reported in Section III-D.

III. Results

In this section we report on the results of the industrial collaboration. First, we present the DMs that emerged from the workshop sessions. Then, we present the reconstructed architecture of the software under study. Finally, a typical software function is examined in more detail, *i.e.,* our variability feature, to derive a measurement plan in order to showcase how software sustainability can be measured in practice.

A. Decision Map

As outlined in Section II, the workshop session aimed to elicit and uncover sustainability concerns related to the Zahori software product. The workshop resulted in an initial DM version, which was revised in an iteration by the researcher and re-evaluated together with the software developers, resulting in the final DM, as shown in Figure 2. The DM identifies the two technical features (i) Automated Regression Testing and (ii) Dataset Creation and Maintenance. Since Zahori is defined as

[2]We borrowed from Lago *et al.* [9] the notions of variation point and variants applied to features in software architecture, where the first (see definitions in Pohl *et al.* [10]) is in our case an architecture element that can be realized in different ways, *i.e.,* the variants.

[3]Mini-view is a simple term introduced to illustrate an architecture model that according to the ISO/IEC/IEEE 42010:2011 standard [11] is used to focus on specific elements or aspects of a view.

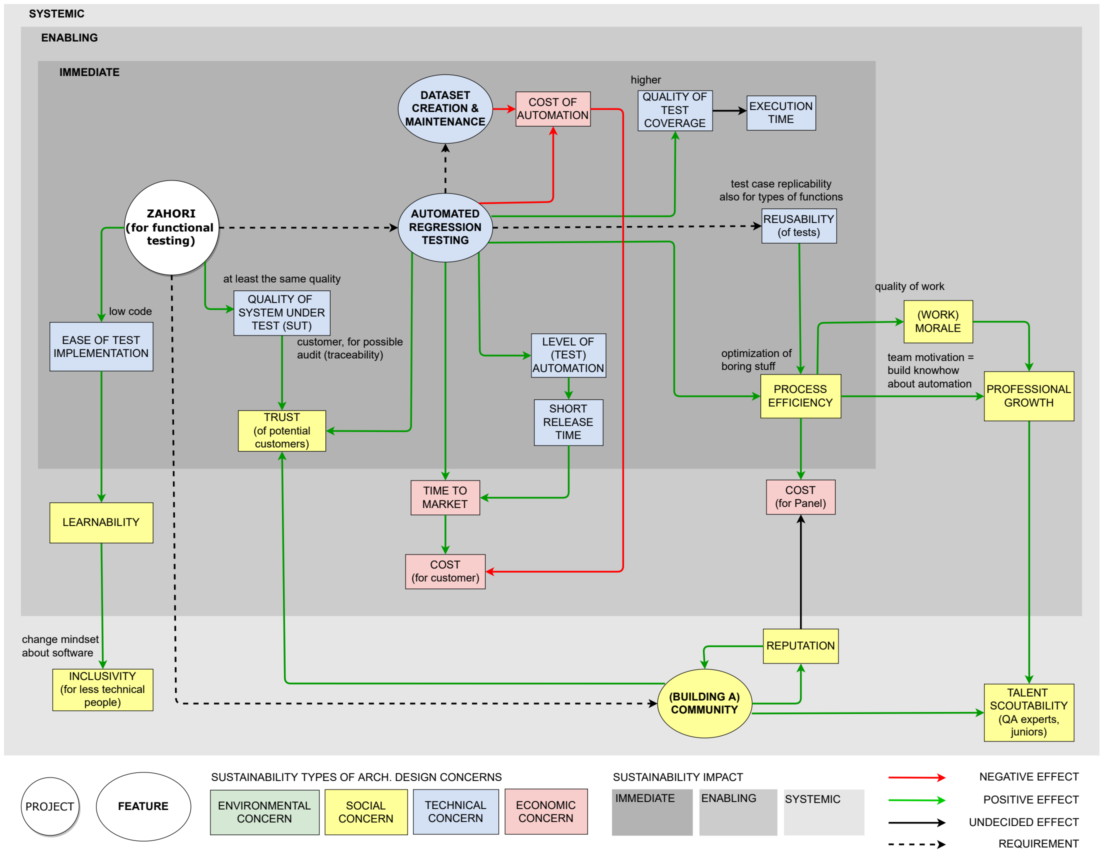

SYSTEMIC
ENABLING
IMMEDIATE
DATASET CREATION & MAINTENANCE
COST OF AUTOMATION
higher
QUALITY OF TEST COVERAGE
EXECUTION TIME
test case replicability also for types of functions
REUSABILITY (of tests)
ZAHORI (for functional testing)
AUTOMATED REGRESSION TESTING
at least the same quality
QUALITY OF SYSTEM UNDER TEST (SUT)
low code
EASE OF TEST IMPLEMENTATION
customer, for possible audit (traceability)
LEVEL OF (TEST) AUTOMATION
quality of work
(WORK) MORALE
optimization of boring stuff
team motivation = build knowhow about automation
PROCESS EFFICIENCY
PROFESSIONAL GROWTH
TRUST (of potential customers)
SHORT RELEASE TIME
TIME TO MARKET
COST (for Panel)
LEARNABILITY
COST (for customer)
REPUTATION
change mindset about software
INCLUSIVITY (for less technical people)
(BUILDING A) COMMUNITY
TALENT SCOUTABILITY (QA experts, juniors)
PROJECT
FEATURE
SUSTAINABILITY TYPES OF ARCH. DESIGN CONCERNS
ENVIRONMENTAL CONCERN
SOCIAL CONCERN
TECHNICAL CONCERN
ECONOMIC CONCERN
SUSTAINABILITY IMPACT
IMMEDIATE
ENABLING
SYSTEMIC
NEGATIVE EFFECT
POSITIVE EFFECT
UNDECIDED EFFECT
REQUIREMENT

Fig. 2: Zahori sustainability decision map (DM)

an automation tool for software testing, it is apparent that with **automated regression testing**, Zahori aims to improve especially technical aspects such as the *level of (test) automation* or the *reusability of tests*. These technical concerns do also have a positive effect on social sustainability, such as building *trust* among customers by improving the traceability of test cases for potential audits. Further, by automating manual and repetitive tasks, the liberated human resources can now be used for other, more efficient and rewarding work, leading to higher *process efficiency* and thus being defined as a social concern.

The working session uncovered an economic trade-off related to the *cost for customer*. On the one hand, costs decrease due to a shorter *time to market* as tests can be automated and no manual execution or reporting is required. On the other hand, costs can also increase given that test data still needs to be created and maintained manually by the test team. This constraint is attributed to the feature **dataset creation & maintenance**. Nevertheless, these implications do not emerge immediately, but only over time. Hence, the trade-off is defined as an *enabling* impact.

Considering that Zahori was developed as open source software and all resources are publicly available on GitHub, the developers aim to build a social **community** around the software itself. This social feature could help with *talent scouting*. However, building an actual community would require behavioral changes, *i.e., systemic* changes, in the way Zahori is accepted as a software product by both the testing and software development communities. Nonetheless, building a community around Zahori would also lead to an increased *reputation* for the software product itself, and thus for the responsible development organization. An increased reputation would in turn support the building of the actual community; the more the product is accepted by the community, the larger the community becomes. We identify such a “reinforcement” as a positive network effect [12].

B. *Reconstructed Architecture*

Following the working session and the resulting DM, the architecture of Zahori was manually reconstructed. The aim was to create an architectural view that highlights potential architectural patterns and illustrates software components. Such an additional view allows us to map architecture elements to sustainability aspects in more detail. We combined existing documentation with manual source code inspections to arrive at this view. Figure 3 shows the high-level architecture retrieved from the GitHub repository. As shown, the system is divided into the Server and the Automated Processes. The **Server** is responsible for all tasks related to regression testing, such as defining, orchestrating, and monitoring test cases. The server can be accessed either using a dedicated graphical user interface (GUI) or a REST API. The API is also utilized by the Zahori server itself for process communication and enables process discovery together with Eureka [13] , a service registration and discovery integration.

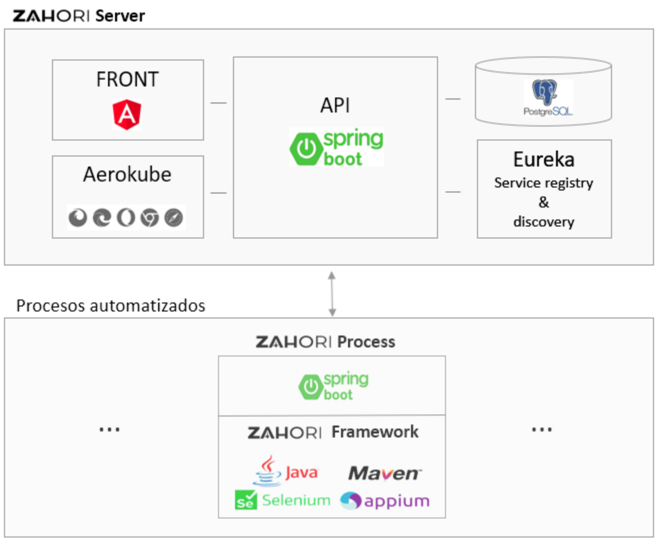


Fig. 3: Zahori components[4]

The actual test cases are implemented and created by the **Process Automation** component. Selenium [14] is used for web browser automation. This enables functional and regression testing for web-based processes and applications. Beyond the testing capabilities, this automated process component is also responsible for the Evidences feature.

Figure 4 outlines the final reconstructed architecture after validating the preliminary reconstructed architecture with the Zahori developers. As shown, the resulting view retains the division between Zahori Server and Process Automation. However, by also including the source code packages in the view , we can unambiguously identify the architectural pattern Model-View-Controller (MVC) for the Zahori server. The MVC pattern allows the separation of the user interface(s) from the actual business logic. In addition, based on the source code packages (*i.e.,* Repository and DB), it was possible to uncover the Data Access Object (DAO) pattern. Together with an Object Relation Mapping (ORM), this enables the abstraction and isolation of the database and the use of object-oriented abstraction at the data layer.

We also uncovered an inconsistency between the architecture given on GitHub and the reconstructed architecture. Instead of having Eureka service registration and discovery as a component only on the Zahori server, Eureka is used as a cross-cutting component to register both the server and process automation. Therefore, in our reconstructed architecture, we have separated Eureka from the server and presented it as a self-contained element.

After inspecting the source code, we located the Java class responsible for generating the Evidences. As part of the Zahori Framework component, the class itself uses components from *e.g.,* Selenium, Apache [15] and Simple

[4]https://github.com/zahori-io/zahori-doc. Accessed: 05-12-2022.

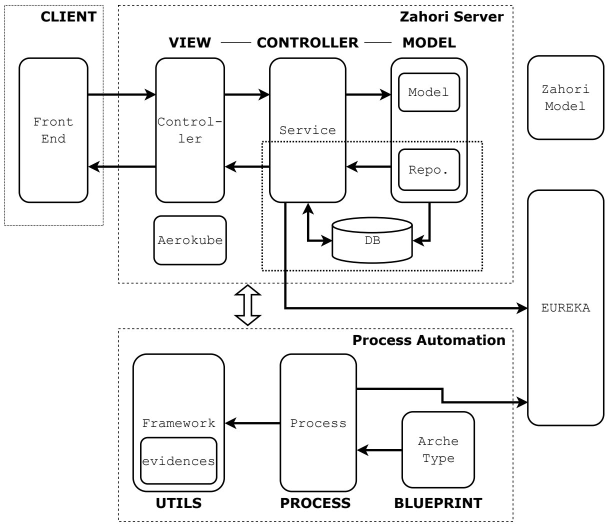


Fig. 4: Reconstructed Zahori architecture

Logging Facade for Java (SLF4J [16]) to generate the various evidences of the executed tests (logs, screenshots, documents, videos, HAR files) [17].

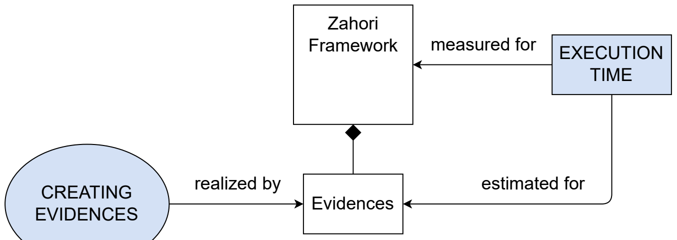
In summary, the reconstruction of the architecture confirmed as well as extended the existing architecture description. Hence, for the rest of our study, we can use the reconstructed architecture to map the uncovered elements to the related sustainability concerns and features.

C. *Variability Feature Mini-views*

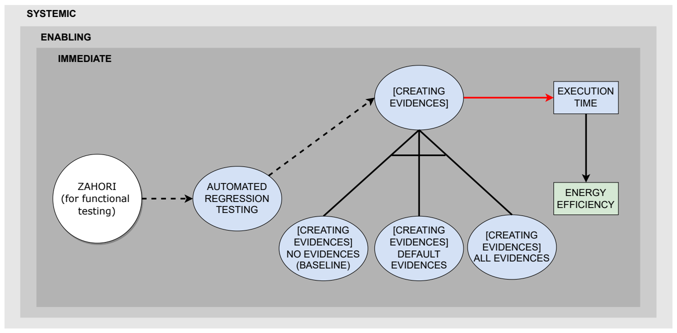
Considering the two results from the previous steps, we selected a specific variability feature, *i.e.,* Evidences, to demonstrate the identification of architecture elements (given by the reconstructed architecture) that affect the sustainability concerns (given by the DM). This feature selection is based on the observation that the DM established during the working session had not addressed environmental concerns. However, as Zahori's current customers have a strict sustainability strategy (due to the utilization in the aviation sector), environmental concerns should indeed be part of the overall architecture. After consultation and analysis of the reconstructed architecture together with the developers, it turned out that the Evidences feature could have an impact on the execution time of Zahori itself and thus on energy efficiency. Such "reassessment" is part of the DM thinking process to uncover missing concerns [7]. To establish empirically evidence on this hypothesis, *i.e.,* the impact of Evidences on environmental sustainability, we created a measurement plan to monitor this impact. Below we illustrate and examine how variability feature mini-views help to visualise (i) the mapping between architecture elements and concerns, and (ii) the variability feature itself.

Figure 5a shows the mapping of the Evidences feature to its architecture element and the sustainability concern. We can see that the Zahori feature is implemented by the *Evidences* Java class, which is part of the Zahori framework component. We can locate this component in its context in Figure 4, *i.e.,* in the

Zahori process automation component. As shown in the DM in Figure 2, we can also locate the sustainability concern *Execution Time*. Since it is only feasible to monitor the execution time of a particular test case for the entire Zahori application, we need to estimate the execution time for the particular Evidences component (this is further discussed in Section III-D).



(a) Architecture element “Zahori Evidences” mapped on the Zahori feature and its sustainability concerns


SYSTEMIC
ENABLING
IMMEDIATE
ZAHORI (for functional testing)
AUTOMATED REGRESSION TESTING
[CREATING EVIDENCES]
EXECUTION TIME
ENERGY EFFICIENCY
[CREATING EVIDENCES] NO EVIDENCES (BASELINE)
[CREATING EVIDENCES] DEFAULT EVIDENCES
[CREATING EVIDENCES] ALL EVIDENCES

(b) Decision map illustrating the notion of *variability feature*, *i.e.,* the Zahori Evidences feature; its *variants*, *i.e.,* (i) no evidences, (ii) default evidences, (iii) all evidences; and the impact on *sustainability*, *i.e.,* execution time in the technical dimension and energy efficiency in the environmental dimension

Fig. 5: Variability Feature Mini-views

Figure 5b shows the variability feature in its context by zooming in the DM from Figure 2. This mini-view illustrates Evidences as a sub-feature of the main-feature Automated Regression Testing. Invoking an additional feature, *i.e.,* creating Evidences, obviously negatively impacts the overall Execution Time of a given test case (red arrow). However, the execution time depends on the evidence variant. In this research, we consider three alternative variants: (i) default evidence (including logs and screenshots), (ii) all evidence (logs, screenshots, documents, videos), and (iii) no evidence. To illustrate the different variants, we extended the notation of DMs by adopting the notation of “alternative features” from Lago *et al.* [9]. As empirical measurements are not yet incorporated, the impact of execution time on the overall energy efficiency is defined as undecided (black arrow). In the next section, however, we report on the results of the controlled experiment to provide the empirical evidence.

D. Variability Feature Analysis

In this final step, we created a measurement plan for our variability feature to implement the metrics related to the two sustainability concerns, *i.e.,* execution time and energy efficiency, and to monitor their impact on sustainability. To do so, we used the sustainability-quality (SQ) model from the Sustainability Assessment Framework (SAF) Toolkit [18]. Table I lists the two sustainability concerns Execution Time (ET) and Energy Efficiency (EE) as quality attributes (QAs). However, as mentioned in the previous section, the metrics for ET and EE are variant specific. To estimate ET and EE, we first measured the impact of the entire variant and then subtract the baseline measurement, as indicated in Equation (1) and (2) in the SQ model in Table I, where:

$$\begin{aligned}
ET &= \text{Execution Time}\\
EE &= \text{Energy Efficency}\\
feature &= \text{Var. Feature; } CreatingEvidences\\
variant_X &= \text{Variant; } [no|default|all]\text{-}Evidences
\end{aligned}$$

From an additional study [17], where the measurement plan has been implemented in a controlled experiment, we can derive the following conclusion:

> “*Default Evidences* (logs and screenshot) have negligible impact on energy, while *All Evidences* (logs, screenshot, video and document) lead to increased energy consumption.” [17]

The result of this study shows the substantial impact of the *all evidences* variant on energy consumption. Therefore, we can support Zahori developers in their decision making by allowing them to better understand the impact of the variability function on the targeted sustainability concerns. We suggest using *default evidences* as the standard variant in a production environment, so that valuable insights are gained in every case, and using *all evidences* only when absolutely necessary.

IV. Discussion and Lessons Learned

In the following, we summarize our key observations emerged from this industrial collaboration and report on the lessons learned. The section is organized according to the phases and outcomes defined in the study design (Section II).

Working session: As described in our study design, the methodology used in this research to model a DM differs from previous work. By defining the researcher as the DM specialist and modeling entity, and the industry partner as the knowledge specialist and validation entity, **we observed a considerable increase in time efficiency**. While previous work spent several days teaching the methodology for DMs and the SAF toolkit itself [8], our working session took only about three hours to produce the initial DM. In addition, **new perspectives on Zahori and its overall sustainability goal emerged** from the inquiries of the researchers. For example, the social feature of ‘building a community’ was before only implicit in the minds of the developers (*i.e.,* tacit knowledge [19]).

However, involving the researcher in the DM modeling process could bias the results and pose a potential threat to construct validity. The “experimenter expectancies” [20] may have influenced the way the DM was created by incorporating the researcher’s own expectations and their experience in modeling DMs. To mitigate this risk, non-DM experts also

TABLE I: Sustainability Quality (SQ) Model
REF = Reference; Dimensions: **TEC** = Technical; **ENV** = Environmental; **ECO** = Economic; **SOC** = Social

QUALITY ATTRIBUTE	DEFINITION	REF	TEC	ENV	ECO	SOC	METRIC/KPI
Execution Time (ET)	the time it takes for Zahori to execute the predefined set of tests; Time (in seconds)	[17]	X				$ET_{feature}(variant_X) = ET_{system}(variant_X) - ET_{system}(baseline)$ (1)
Energy Efficiency (EE)	the total energy consumed by Zahori to execute a predefined set of tests; Energy (in Joule)	[17]		X			$EE_{feature}(variant_X) = EE_{system}(variant_X) - EE_{system}(baseline)$ (2)

participated in the working session, and the resulting DM was re-presented to the industry partner as a final validation step.

Reconstructed architecture: A thoroughly documented software architecture is essential to map the architecture elements to quality concerns and features at a fine granularity. Having only high-level components available hampers in-depth investigation and examination of the architecture elements that impact sustainability. Therefore, we reconstructed the architecture of Zahori. **By comparing the given architecture with the reconstructed one, we were able to (i) confirm the given view, (ii) uncover inconsistencies between the given architecture and the source code, and (iii) identify common software architecture patterns.**

Variability Feature: By introducing variability features, we were able to identify a particular application feature and identify three alternative variants, each with a different impact on the overall sustainability of the Zahori application. We incorporated the "alternative feature" notation of Lago *et al.* [9] into our DM to visualise variability features and variants. With this extension, it is now possible to attribute sustainability concerns and corresponding metrics to different instances, *i.e.,* variants of the application and a given feature. Moreover, **introducing the notion of variability feature supports the evaluation of different usage scenarios, hence enabling data-driven sustainability choices**.

Variability Feature Mini-views: A variability feature mini-view allows us to map architecture elements (*e.g.,* components, classes, instances) to both system features and sustainability concerns. While the mapping of elements to features identifies the associated implementation units; the mapping of elements to concerns identifies the implementation units that can be used to measure. **Such mini-views are a first attempt to integrate architecture elements in a thinking framework and to visualise their mapping with quality concerns.**

Strategies and Suggestions: Based on our results and the introduction of the notion of variability features, **we were able to provide our industry partner with strategies and suggestions to improve the sustainability of their software product**. First, along with the reconstructed architecture, developers are now able to map additional architecture elements to the available DM and their quality concerns in order to create a concrete measurement plan and monitor their impact. Second, by using the concept of variability features, we provide developers the ability to analytically investigate further Zahori features and their impact on sustainability, measure them, and make design choices based on sustainability concerns. Finally, **to decrease execution time and thus be more energy efficient, we can provide empirically-derived guidance** (*cf.* Section III-D), *i.e.,* suggest the variant *No Evidences* in a development and test environment, *Default Evidences* in a production environment, and *All Evidences* only if absolutely necessary, *e.g.,* for legal matters. Adjusting the level of evidences in favor of energy efficiency could, of course, impact other sustainability concerns, such as *Trust* in the social dimension. Therefore, we generally suggest a thorough trade-off analysis to ensure that when one sustainability aspect is taken into account, the others are not affected to such an extent that the measure becomes unacceptable.

V. Conclusion and Future Work

In this paper, we reported on the experience of applying the thinking framework decision maps [7] to an industrial software product. We introduced for the first time the notion of variability feature applied to sustainability. Through our case study, variability features resulted to be beneficial in supporting the architecture process towards more sustainable decisions.

In the future, we plan to further evaluate both the notion of variability feature and the associated mini-views with software architects to assess their modeling feasibility in more complex software-intensive systems, and their practical application in other domains. This extended evaluation will also allow us to mitigate potential threats to external validity linked to the single case study adopted for this research. In addition, we intend to further design and evaluate a systematic process to map architecture elements to sustainability concerns.

For researcher, this work provides an experience report on the application and assessment of variability features in an industrial context. For practitioners, we provide a comprehensive use case on how architecture elements can be determined to evaluate their impact on sustainability concerns and hence assessing the overall sustainability of the system at hand. Based on specific measurements, software architects are supported at design time and decision makers can develop strategies towards achieving sustainability goals.

References

[1] ACM Technology Policy Council, "ACM TechBrief: Computing and Climate Change," Tech. Rep. Issue 1, Nov. 2021.

[2] M. Uddin and A. A. Rahman, "Energy efficiency and low carbon enabler green IT framework for data centers considering green metrics," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 4078–4094, Aug. 2012.

[3] K. Janssen, T. Pelle, L. de Geus, R. van der Gronden, T. Islam, and I. Malavolta, "On the Impact of the Critical CSS Technique on the Performance and Energy Consumption of Mobile Browsers," in *International Conference on Evaluation and Assessment in Software Engineering*. Gothenburg Sweden: ACM, Jun. 2022, pp. 130–139.

[4] R. Verdecchia, R. Aparicio Saez, G. Procaccianti, and P. Lago, "Empirical Evaluation of the Energy Impact of Refactoring Code Smells," in *International Conference on ICT for Sustainability (ICT4S)*, 2018, pp. 365–345.

[5] S. Vos, P. Lago, R. Verdecchia, and I. Heitlager, "Architectural Tactics to Optimize Software for Energy Efficiency in the Public Cloud," in *International Conference on ICT for Sustainability (ICT4S)*. IEEE, Jun. 2022, pp. 77–87.

[6] R. Kazman, S. Haziyev, A. Yakuba, and D. A. Tamburri, "Managing Energy Consumption as an Architectural Quality Attribute," *IEEE Software*, vol. 35, no. 5, pp. 102–107, Sep. 2018.

[7] P. Lago, "Architecture Design Decision Maps for Software Sustainability," in *41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE/ACM, May 2019, pp. 61–64.

[8] P. Lago, R. Verdecchia, N. Condori-Fernandez, E. Rahmadian, J. Sturm, T. van Nijnanten, R. Bosma, C. Debuysscher, and P. Ricardo, "Designing for Sustainability: Lessons Learned from Four Industrial Projects," in *Advances and New Trends in Environmental Informatics*. Springer, 2021, pp. 3–18.

[9] P. Lago, H. Muccini, and H. van Vliet, "A scoped approach to traceability management," *Journal of Systems and Software*, vol. 82, no. 1, pp. 168–182, 2009.

[10] K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer Science & Business Media, Aug. 2005.

[11] International Organization Of Standardization, "ISO/IEC/IEEE 42010:2011 - Systems and Software Engineering – Architecture Description," International Organization for Standardization (ISO), Tech. Rep., 2011.

[12] G. G. Parker, M. W. Van Alstyne, and S. P. Choudary, *Platform Revolution: How Networked Markets Are Transforming the Economy and How to Make Them Work for You*. Norton & Company, 2016.

[13] Spring-Cloud, "Eureka," accessed: 2022-12-05. [Online]. Available: https://github.com/spring-cloud/

[14] Software Freedom Conservancy, "Selenium," accessed: 2022-12-05. [Online]. Available: https://www.selenium.dev

[15] The Apache Software Foundation, "Apache," accessed: 2022-12-05. [Online]. Available: https://www.apache.org/

[16] SLF4J, "Simple logging facade for java," accessed: 2022-12-05. [Online]. Available: https://www.slf4j.org/

[17] M. Dînga, "An Empirical Evaluation of the Energy and Performance Overhead of Monitoring Tools on Docker-based Systems," Master's thesis, Vrije Universiteit Amsterdam, The Netherlands, 2022.

[18] P. Lago and N. Condori-Fernandez, "The Sustainability Assessment Framework (SAF) Toolkit: Instruments to help sustainability-driven software architecture design decision making," Apr. 2022. [Online]. Available: https://github.com/S2-group/SAF-Toolkit

[19] M. Ali Babar, T. Dingsøyr, P. Lago, and H. van Vliet, Eds., *Software Architecture Knowledge Management*. Springer, 2009.

[20] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.