Enriching Cloud-native Applications with Sustainability Features

Monica Vitali
DEIB
Politecnico di Milano
Milan, Italy
monica.vitali@polimi.it

Paul Schmiedmayer
Department of Informatics
Technical University of Munich
Munich, Germany
paul.schmiedmayer@tum.de

Valentin Bootz
Department of Informatics
Technical University of Munich
Munich, Germany
v.bootz@tum.de

***Abstract*—Due to the ever-growing demand for computational resources, the environmental impact of data centers is continuously increasing. Recently, a great effort has been made to mitigate this impact, while the demand for computational resources has continued to grow. Current mitigation strategies focus on the infrastructure perspective, while the application perspective has been neglected. This paper aims to engage application designers and developers on the path to greener application design. Following the Sustainable Application Design Process (SADP) methodology, we introduce a Sustainable Application Design Architecture (SADA) for enriching cloud-native application components with sustainability features that can be exploited to adapt the application workflow to the environmental context. The architecture enables synergies from design to deployment between all stakeholders involved in the application management. The paper focuses on enriching the application with sustainability features in the design and development phases. We also present and discuss a prototype that can translate design-level sustainability features into development features.**

***Index Terms*—cloud-native applications, green IS, application design, adaptive workflow, microservices**

I. Introduction

The energy demand for data centers has been continuously growing in the last decade: Even though it is difficult to assess their actual environmental impact, it is estimated that data centers are responsible for 3% of the global electricity supply and 2% of total greenhouse gas emissions[1]. The increased sensitiveness towards global warming and environmental sustainability, and the recent national and international regulations for reducing CO_2 emissions, have pushed leading cloud providers and IT companies, such as Facebook, Google, and Apple, towards environmental sustainability[2]. These stakeholders are advertising to have reached or are near carbon neutrality, even though these statements are sometimes controversial and inaccurate [1]. From a research perspective, two complementary approaches have been proposed for reducing the environmental impact of data centers: (i) design efficient facilities [2] and (ii) improve server utilization [3].

Even if organizations have implemented both approaches, the demand for cloud services is increasing[3]: the global demand for compute instances for training AI models has increased 300'000 times over the past ten years [4][5] while the energy demand of data centers has doubled [6]. This trend shows that merely focusing on the infrastructural perspective will not be a long-term solution for reaching carbon neutrality in IT. It will become essential that applications are also designed and executed based on their environmental footprint. At the same time, an evolution in the architectural style of the applications has been observed, moving from monolithic to cloud-native applications, designed to take advantage of the characteristics of cloud computing [7] [8] and becoming the de facto standard in industry [9]. In this architectural style, applications are designed by combining tens of microservices that are simple, single-function, and loosely coupled components. This architectural style hides the cloud provider the internal logic and the interaction among the components.

We aim to increase the sustainability awareness of the stakeholders involved in cloud-native application management, from design to deployment, and to support the development of sustainable applications by enriching them with metadata to be exploited for more efficient deployment. Starting from the general Sustainable Application Design Process (SADP) [10], we present an architecture supporting the methodology and define the components needed for its enactment. Additionally, we implemented and tested a prototype on two use cases.

The main contributions of this paper are:

- the definition of an approach to Sustainable Application Design enabling the synergy between infrastructure and application providers;
- a Sustainable Application Design Architecture (SADA) defining the components needed to enrich the application design with sustainability features defined by the SADP and their interconnections;
- a working prototype to support the design and development of sustainable applications.

The paper is organized as follows. In Section II, we discuss existing approaches toward green application design and management. In Section III, we introduce the SADP. Section IV describes the components of the SADA and how they support the implementation of SADP. Finally, the prototype of an interactive interface for designing applications enriched with

[1] https://datacentremagazine.com/articles/efficiency-to-loom-large-for-data-centre-industry-in-2023

[2] https://cloudscene.com/news/2016/12/going-green/

[3] https://www.iea.org/reports/data-centres-and-data-transmission-networks

the features identified in this paper is presented in Section V and validated on two use cases. Section VI draws conclusions and future improvements.

II. State of the Art

Sustainability is becoming a relevant topic in all fields, including IT. Sustainability includes four aspects: technical, economical, social, and environmental[11]. Reference architectures, standards, and metrics are needed to enable practitioners to analyze, design, evaluate, and maintain software systems in collaboration with other stakeholders. In this work, we focus on the environmental aspect.

The attempts to reduce the environmental impact of Information and Communication Technologies (ICT) have been ongoing for several years. Most of them have been driven by the improvement of the Power Usage Efficiency (PUE) metric to measure data centers' energy efficiency and have successfully focused on the cooling efficiency [12]. However, the focus on PUE has shown its limitations, as it does not account for the type of energy used (brown or renewable) [13] and the efficiency of IT operations [14]. A more comprehensive approach to sustainability in cloud computing has been presented by Gill and Buyya [15], through a set of taxonomies for several aspects of Cloud Data Centers sustainability taking the cloud providers' perspective. The considered aspects include application design, sustainability metrics, capacity planning, energy management, and virtualization. Existing techniques are described and classified, but no specific approach is suggested to enact them comprehensively. The cloud provider perspective is also adopted by exploiting the intermittent availability of renewable energy by Thi, Pierson, and Da Costa, Hu, Li, and Sun [16], [17], involving operations such as server consolidation [3], [18] and shutdown policies to ensure a trade-off between energy efficiency and performance [19]. Over the years, best practices have been suggested to promote data center efficiency improvements, such as the EU Code of Conduct [20] and the Data Center Maturity Model [21].

Energy reduction should be a shared responsibility between the cloud provider and the application provider (or cloud consumer) [22]. However, application providers usually lack information to improve their environmental impact [23] while, on the other hand, cloud providers see the applications they host as black boxes [11]. Limited attempts have been made to include sustainability in the application design of Green Information Systems (Green IS) by taking the application provider's perspective. An empirical approach towards energy efficiency of applications in the cloud from the application provider perspective, focusing on architectural aspects, is discussed by Vos, Lago, *et al.* [23]. Vos, Lago, *et al.* classify tactics into three categories: resource monitoring (workload energy assessment), resource allocation (scaling, scheduling, brokering), and resource adaptation (data reduction, granular scaling, batch execution, edge deployment). Current research mainly focuses on dynamic resource allocation and scheduling according to energy efficiency optimization [24], [25]. Google uses an approach to resource allocation based on time and space considerations [26], [27]. Single microservices are considered black boxes in this decision process, and diversity and interactions between microservices are not exploited. Other attempts have proposed general principles and lack practical solutions [28]–[31].

The energy impact of individual workloads has to be monitored to improve the sustainability of applications. Its measurement is not trivial. Google has recently presented a Carbon Footprint Reporting Methodology to provide a report of the emissions produced by a user using its services[4]. Microsoft Azure [32], Google Cloud Platform [33], and Amazon Web Service [34] are now providing a dashboard to report the energy consumption and CO_2 emissions at the cloud service level. All these tools lack details and do not provide timely information. Third parties tools have been developed to timely estimate applications' energy usage and carbon emissions. The CodeCarbon initiative [35] estimates CO_2 emissions of Machine Learning tasks in the geographical location and the energy mix of the country in which the application is deployed. Similarly, The Cloud Carbon Footprint Tool [36] estimates the energy consumption of cloud instances. Power is also one of the metrics considered by Brondolin and Santambrogio, providing closed-box monitoring for multi-component applications [37]. Some proposals aimed to estimate the energy efficiency of specific applications in embedded systems [38] but cannot be applied in complex cloud infrastructures. A virtual energy system exploited to optimize the carbon efficiency of applications is proposed by Souza, Bashir, *et al.* [39].

All the introduced approaches attempt to enable energy awareness in cloud applications, but the workflow enhancement perspective is missing. Some solutions have been proposed for specific types of applications as by Schneider, Basalla, and Seidel, where guidelines are provided for designing Data Mining tasks [40]. Aligning with current trends in cloud application design, we decide to focus on cloud-native applications. Cloud-native applications empower the design and execution of scalable, loosely coupled, resilient, manageable, and observable applications [8]. These applications generate complex workflows as a consequence of the microservice architectural style. Specific characteristics of these applications can be exploited to improve energy efficiency. Elasticity can be used to balance energy and performance [41]. Energy-driven considerations in Mobile Cloud Computing can lead to microservices' offloading [42]. An attempt to consider the workflow and the interdependencies between microservices of the same applications is presented by Saboor, Mahmood, *et al.* [43], where a ranking approach allocates microservices to containers according to their interactions. All these approaches keep the infrastructural perspective in the optimization of resource usage. The design of adaptive applications, able to change their workflow according to the current workload and energy mix, is necessary to empower green applications from an application provider perspective. The power-aware brownout strategy proposed Papadopoulos, Krzywda, *et al.*

[4]https://cloud.google.com/carbon-footprint/docs/methodology

presented an approach to adapt the workflow execution by producing a response containing minimal components (mandatory part) while disabling additional components in case of limitations in power availability [44]. Gerostathopoulos, Raibulet, and Lago suggest the usage of decision maps to make sustainability-driven decisions [45]. Decision maps associate tactics and sustainability goals. For example, they use service scaling and something similar to brown out to adapt the application workflow from an economical and technical sustainability perspective. A way to represent the workflow is needed to enable the application's adaptivity. A standard way to model the interaction between microservices is missing, even though this information is crucial to enhance their management. We opted to represent microservice choreography through Business Process Model and Notation (BPMN) fragments, exploiting a well-known process modeling notation to represent microservices interactions [46]. However, in their approach, the adaptive aspect of the interaction cannot be mapped.

This work focuses on enriching the design of cloud-native applications and their workflow to boost the opportunity to reach sustainable applications by involving all stakeholders (application designers, developers, and infrastructure providers) in this process.

III. Sustainability Approach

The path towards carbon-neutral IT must pass not only through an improvement of the infrastructural level and its management but also through a more conscious and sustainable design of the applications using these infrastructures.

The cloud-native approach has recently become more and more popular. According to the cloud-native paradigm, applications are implemented as a set of composite microservices and serverless functions, uncoupled and independent, interacting with each other through synchronous and asynchronous messages. Typically, an application comprises tens to hundreds of independent microservices cooperating to reach the goal set by the application provider.

As described in [10], an application can be enriched with a set of sustainability features at design time that can be utilized by adapting the execution workflow according to the context. Even though some of these characteristics are intrinsic in the application design, they are currently impossible to be used as they are embedded in the implementation details.

The design of greener applications can affect the overall emissions of IT only if the synergy between different actors involved is enabled. The proposed approach considers three actors: (i) the **application designer** is made aware of the current sustainability of the application through a set of metrics and indicators and is in charge of setting sustainability targets and of re-designing the application adding sustainability features; (ii) the **application developer** is in charge of implementing the sustainability features included by the application designer according to the specific framework used for the application development and management; (iii) the **infrastructure provider** adapts the application deployment and scheduling according to the rules and characteristics added in the design and development phases and provides energy and carbon footprint data about the current application execution.

The SADP approach described in [10] identifies a set of features and makes them explicitly defined during the design process. However, these features need to be visible to the infrastructure provider. Thus, the features added at design time must be reflected in the implementation of the application components. This paper introduces a Sustainable Application Design Architecture (SADA) and focuses on the synergy between the design and the development phases.

In this section, we summarize the Sustainable Application Design Process (SADP), introduced in [10], describing a set of features used to enrich the sustainability of applications at design time.

a) Sustainability Awareness (SA): Each microservice can be enriched with functional and non-functional properties that describe its requirements and capabilities. These properties include the cloud instance type (functional) and Quality of Service (QoS) and power consumption requirements (non-functional). **Applicability:** Information about each component's functional and non-functional requirements enables the assignment of the correct amount of computational resources and the best deployment location.

b) Microservice Classification (MC): A microservice can be labeled optional or mandatory for the application execution. Mandatory microservices are necessary to obtain the desired result, while optional microservices increase the Quality of Experience (QoE) of the final users or the gain of the application provider. **Applicability:** Annotating each component of the application with its relevance in the workflow makes it possible to adapt the application workflow according to the current execution context and to reduce the environmental impact of applications in case of green energy sources shortage.

c) Microservice Enrichment (ME): A microservice can have multiple implementations with different functional and non-functional properties. The same functionality might be carried out in several ways according to the context of execution. An example is the fail-over mechanism, in which the application adapts to QoS issues by executing alternative, less demanding tasks. **Applicability:** Different implementations for a microservice enable an adaptive workflow in which a variant can be selected according to the current context.

SADP suggests an iterative approach through which these features are added to the application design. The three steps are incremental refinements: the designer can focus on a single component at each iteration, enriching the model step by step.

IV. A Sustainable Application Design Architecture (SADA)

In this section, we introduce the Sustainable Application Design Architecture (SADA). The architecture supports the design of sustainable applications enabling the synergy between the stakeholders involved in the application management

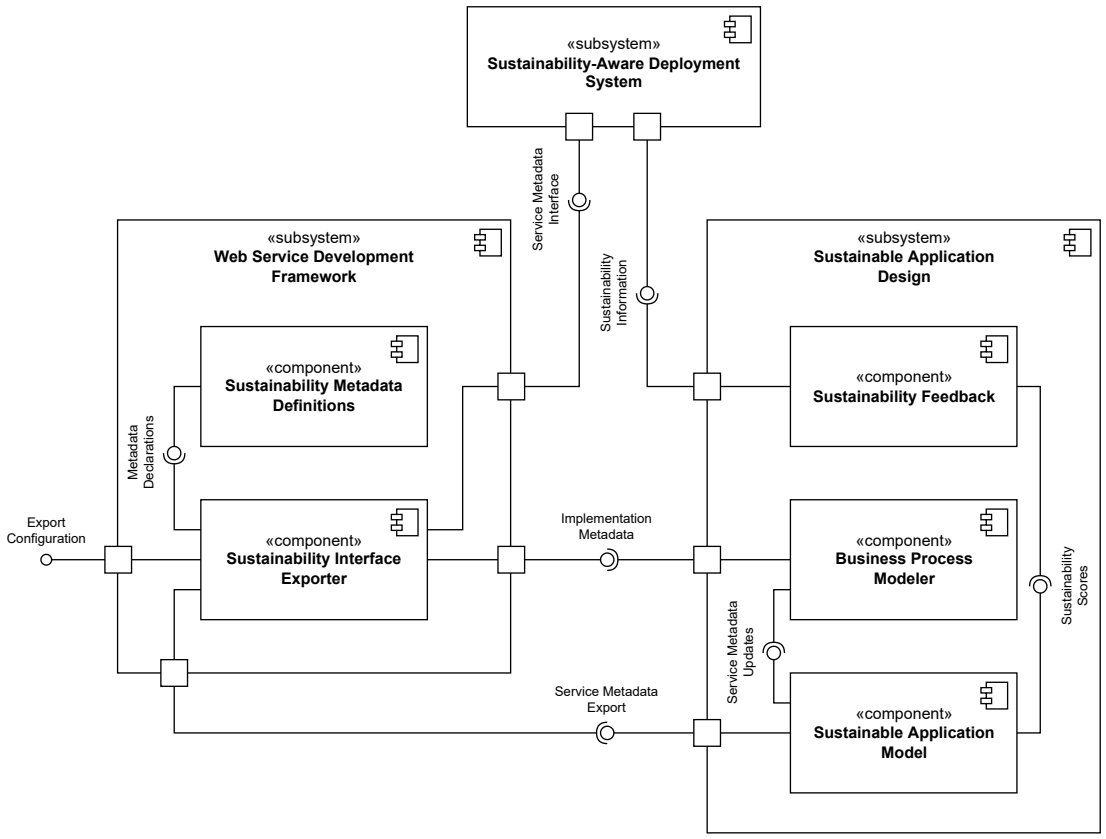

Fig. 1. UML component diagram detailing the Sustainable Application Design Architecture (SADA) including the three subsystems: i) Web Service Development Framework to enable developers to annotate sustainability information to web services, ii) Sustainability Application Design to integrate multiple components and refine annotations, and iii) Sustainability-Aware Deployment System to close the feedback loop and inform cloud service providers.

phases. The overall architecture is shown in Figure 1 using a Unified Modeling Language (UML) component diagram. The architecture is characterized by three main subsystems, each one responsible for one of the phases of the application management and addressed to one of the actors defined in Section III:

- *Sustainable Application Design*, addressed to the application designer;
- *Web Service Development Framework*, addressed to the application developer;
- *Sustainability-Aware Deployment System*, addressed to the infrastructure provider.

While all three actors and subsystems provide and fulfill distinct functionalities in an integrated software engineering process, stakeholders can assume multiple roles simultaneously. Sometimes, a tighter integration might be beneficial or necessary, e.g., when working in smaller teams or cross-functional organizations. Therefore, the generalized concepts defined in our process model can be seen as a blueprint adaptable to the needs and requirements of each sustainability-focused development team or infrastructure provider. We explicitly encourage cross-disciplinary collaboration to tackle the numerous sustainability challenges for cloud-native applications.

Figure 1 defines the software architecture, including subsystems, components, and interfaces [47]. We intend to make this architecture as general as possible since the web service developers can rely on various technologies, programming languages, and development stacks. We do not define specific dependencies to these tools and focus on larger architectural patterns and mechanisms.

The individual components all aim to support and automate the stakeholders' goals to collaborate on enhancing cloud-native applications with sustainability features. We briefly describe all the subsystems, focusing mainly on the design and development phases and on how to support their alignment and interaction. We highlight the benefits these components bring to the day-to-day development of cloud-native applications and how they instantiate the SADP.

A. Sustainable Application Design

This first subsystem supports the application designer in enriching the application with sustainability features at design time. The *Sustainable Application Design* subsystem has to: (i) enable the (re)design of the application, including the features of the SADP; (ii) provide feedback on the current sustainability level of the application; (iii) enable to export the application model in a machine-readable format to ease the implementation of the defined characteristics.

The three components depicted in Figure 1 inside the *Sustainable Application Design* subsystem cover these tasks. More specifically, the subsystem is provided with two front-end components (i.e., *Business Process Modeler* and *Sustainability Feedback*) interacting with the application designer and a back-end component (*Sustainable Application Model*).

1) Business Process Modeler: As discussed in Section II, there is a lack of standard models to represent the sustainability-related context associated with components of an application and its workflow. We propose to embed the sustainability context embedded in the BPMN notation [48], a standard to model an organization's internal processes and collaborations and orchestrations between different organizations. Following the approach adopted in [46], the application can be represented as a set of microservices orchestrated by a general process, defining their order of execution (workflow). This integration enables us to build on proven and tested solutions and extend them with metrics and information relevant to all involved stakeholders.

The *Business Process Modeler* component implements the front-end for the application design using BPMN. It enables the application designer to load and edit an existing model or to create a new one. It also allows the enrichment of this model with the sustainability features introduced in Section III. Each task in the BPMN model represents a microservice of the application and is linked to its implementation by specifying the service identifier. This enables the alignment between the application representation and its actual implementation.

As allowed by BPMN, the notation is enriched to include a *Cloud Native Sustainability Extension*. More specifically, a *Cloud Native Sustainability Task* type has been introduced that enables to: i) classify the task as optional or mandatory according to the MC feature in SADP through a boolean attribute in the task definition; ii) define multiple variants for each task according to the ME feature in SADP, specifying the implementation endpoint for each of them; iii) specify a set of requirements for each of the variants, each represented in a machine-readable format as "$[requirement] : [value]$".

2) Sustainability Feedback: This front-end component focuses on the sustainability awareness of the application designer. It shows the environmental impact of the current implementation by exploiting the monitoring data collected by the infrastructure provider and accessed through the *Sustainability Information* interface. Additionally, the component provides information about the current sustainability maturity level of the application design, providing hints on further improvements. A set of metrics for this purpose have been defined in [10] and implemented in this component. A summary is shown in Table I.

3) Sustainable Application Model: The Sustainable Application Model translates the BPMN and all the metadata obtained through the extended version into a structured XML document and provides a *Service Metadata Export* interface. The module extracts the relevant metadata about the sustainability features of the application design into a structured document in a specific format (e.g., JSON or XML). This document can be used as input for the implementation, listing the features to be included by the application developer in the microservices.

The component also keeps the synergy between the application model and its implementation. It allows the user to upload both the BPMN model and the metadata extracted from the actual implementation of the microservices and automatically enriches the application model with them.

Finally, the *Sustainable Application Model* component is in charge of the assessment of the metrics described in Table I and shown by the *Sustainability Feedback* component, obtained from the analysis of the expressed metadata.

B. Web Service Development Framework

The *Web Service Development Framework* subsystem defines how to enrich the microservices composing the application with the metadata expressed in the design. It also enables the application designer and developer to exchange sustainability information. We deliberately empower the application developer as the expert in the detailed implementation decisions to provide default sustainability information for each microservice. The two components are defined regardless of the specific tools and languages used to implement and orchestrate.

1) Sustainability Metadata Definitions: It enables the application developers to specify sustainability metadata within the microservice implementation. The metadata are associated with endpoints or specific functionalities of a web service. In this way, the implementation of a microservice is enriched with functional and non-functional properties expressed in the SA features, can be labeled as optional, or can define several endpoints as different versions of the same functionality. The implementation of this component is technology-dependent, as demonstrated in Section V. The component provides a *Metadata Declarations* interface for making the defined metadata accessible.

2) Sustainability Interface Exporter: It manages the exchange of information with the *Sustainable Application Design* subsystem. It collects the metadata from the *Sustainability Metadata Definitions* component through the *Metadata Declarations* interface. It translates them into a standard format that can be processed by the *Business Process Modeler* component (e.g., JSON). The application design can be fed with the implemented sustainability features. This module is also responsible for including the updates in the application design, exported through the *Service Metadata Export* interface: the component compares the metadata of the current implementation with the ones extracted from the updated design. This information is provided to the application that can implement the new required features.

C. Sustainability-Aware Deployment System

SADP provides a methodology for improving and evaluating the sustainability in the design of a cloud-native application, enriching each component with additional details and metadata. This information can impact the sustainability

TABLE I
APPLICATION SUSTAINABILITY LEVEL ASSESSMENT

Score	Description	Metric
SA_score	Sustainability Awareness Score – Assess the completeness of metadata on requirements, QoS, and energy usage	Coverage (0–100 %)
MC_score	Microservice Classification Score – Assess the availability of annotations about relevance for each component	Coverage (0–100 %)
ME_score	Microservice Enrichment Score – Assess the definition of alternative modalities of execution for each component	Coverage (0–100 %)

of the application only if it is used at run-time properly, driving the decisions on how to deploy and manage the application. The *Carbon-Aware Deployment System* is the connection point between the Infrastructure Provider, who is in charge of deploying the application, and the Application Designer and Developer, who designs the adaptive workflows. This subsystem is out of the scope of this paper. However, according to the SADA, it will have to provide two interfaces: (i) a *Sustainability Information* interface, through which the application designer can collect the information about the actual environmental impact of the current implementation of the application at run-time; (ii) a *Service Metadata Interface* providing the functionalities through which the application can be deployed. The sustainability metadata with which the microservices have been enriched in the previous phases can be exploited by this subsystem. For instance, the type of instance expressed in the SA feature will be used to select the type of VM to be used, as well as the non-functional requirements that can be used to monitor the correct behavior of the application. A sustainability-aware scheduler can take advantage of the ME and MC features by adapting the set of active microservices and their version according to the context of execution and the availability of renewable resources.

V. VALIDATION

This section validates the proposed methodology and architecture, focusing on the *Sustainable Application Design* and the *Web Service Development Framework* subsystems. The goal of the validation is not to provide a complete and general implementation of the architecture components but to prove the feasibility of the approach and the existence of tools in the state of the art to support the SADA.

Section V-A describes an instantiation of the SADA building on top of the standards and elements noted in Section IV. We apply the proposed approach to two different use cases: a Flight Booking application (Section V-B) and the Online Boutique application (Section V-C). All the source code used for implementing the architecture components and the experiments presented in this section is publicly available[5].

A. Instantiation

The *Web Service Development Framework* subsystem enables the specification of sustainability metadata in the web service implementation. Annotations are integrated into the design and implementation of endpoints and components. They express the level of detail of the information defined by the application developer and designer. Code annotations or Domain-Specific Languages (DSLs), that are *"computer programming language of limited expressiveness focused on a particular domain"* [49], are typical mechanisms to annotate domain-knowledge information to implementations. We instantiated the web service development framework integration components into a DSL-based Apodini[6] web service application framework. Apodini is a declarative Swift framework to express the interface, requirements, and functionalities of web services in a single internal DSL independent of any specific middleware, protocol, or Interface Description Language (IDL) [50], [51]. This integration demonstrates the capabilities of metadata definitions and annotations in a declarative software development environment and builds on existing integrations and extension points in the research project. The provided integration uses the Apodini metadata system to enable cloud-native developers to annotate microservices with sustainability-related information [51]. Metadata definitions in Apodini specify the scope and structure of metadata that application developers can use to annotate the application components. Domain-specific sustainability metadata definitions enable developers to annotate Apodini web services with the sustainability-related information added in SADP. The description of the Flight Booking application with Apodini is shown in Listing 1 while Listing 2 shows the Apodini specification for one of the versions of the *Flight Search* component. The metadata includes description, execution modality, and functional (type of instance required) and non-functional (maximum response time) requirements.

We instantiated several reusable and extensible tools to facilitate the SADP building on the general components defined in the SADA. The first tool instantiates the sustainable application design subsystem by providing a graphical interface and functionalities to enrich the application design with the described sustainability features. A view of the tool is shown in Section V-B. The tool is integrated with a BPMN modeler allowing the user to model and annotate the application workflow. The prototype extends the off-the-shelf BPMN.io[7] tool with SADP features. The editor is enriched to support SADP features: an additional dashed border represents optional tasks introduced by the MC feature. A marker in the bottom right represents multiple variants (ME feature) defined for a component. The custom properties panel through which the sustainability features can be edited is integrated into the editor as demonstrated in Figure 2. It is also possible to specify

[5]https://vitali.faculty.polimi.it/sadp/

[6]https://github.com/Apodini

[7]https://github.com/bpmn-io

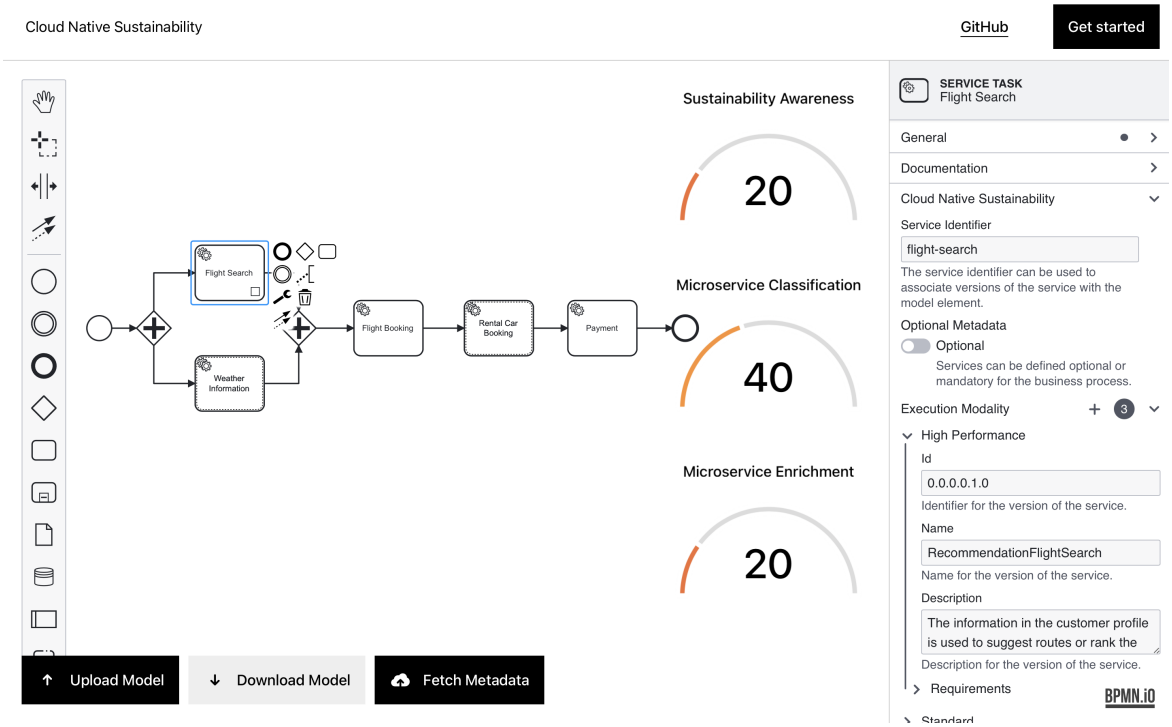

Fig. 2. User Interface of the Sustainable Application Design tool for the Flight Booking application. The user interface is displayed in a browser with the BPMN modeler and the sustainability information components being integrated into the right detail inspector.

the service identifier for each displayed component, linking it to its implementation. A component classification boolean property can be expressed (optional or not), and different execution modalities can be specified. These classifications can be linked with their implementation endpoints and requirements. The application designer can start the annotation process from scratch or upload an existing BPMN model in XML with the *Upload Model* button. The *Service Identifier* property links the task represented in the model with its implementation. This association allows importing existing metadata directly from the microservice implementation using the *Fetch Metadata* button.

The defined service metadata is stored using custom attributes and elements from the BPMN extension in the XML representation of the application. The *Download Model* button enables the export of the enriched design. The dashboard also provides feedback on the sustainability level of the application with three indicators, one for each dimension. Sustainability scores are computed according to the metrics defined in Table I. This information can be used to decide how to improve the sustainability level of the application and to detect which features require more attention.

B. Flight Booking Use Case

We demonstrate the implementation of the architectural components using a Flight Booking application shown in Figure 3). The application consists of five microservices. The *Weather Information* and the *Rental Car Booking* components are optional (their execution can be skipped if needed), while

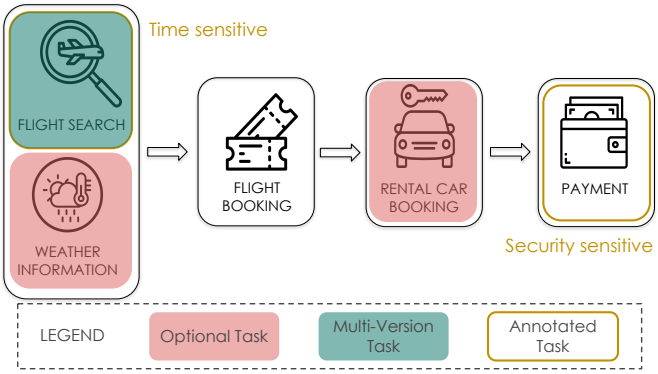

Fig. 3. Microservices of the Flight Booking application

for the *Flight Search* component, three versions have been defined: i) a normal version, collecting updated data from flight companies at each new request; ii) a low-power version using cached and possibly outdated information in the preliminary search phases to reduce energy waste; iii) a high-performance version, using a recommender system to improve the results presented to the customer and their order.

The screenshots and code snippets demonstrate the prototype functionalities and the ability of the solution to ensure the synergy between the application design and development phases through the set of tools described in Section V-A. Figure 2 demonstrates the user interface of the sustainable application design dashboard while Listing 1 and Listing 2

```
struct FlightBooking: WebService {
    @OptionGroup
    var options: SustainabilityDocumentExportOptions

    var configuration: Configuration {
        REST()
        Sustainability(options)
    }

    var content: some Component {
        FlightSearchService()
            .serviceIdentifier("flight-search")
        WeatherInformationService()
            .serviceIdentifier("weather-info")
        FlightBookingService()
            .serviceIdentifier("flight-booking")
        RentalCarBookingService()
            .serviceIdentifier("rental-car-booking")
        PaymentService()
            .serviceIdentifier("payment")
    }
}
```

Listing 1. Flight Booking application configuration using the Apodini framework. The configuration details the different components of the microservice architecture and the `Sustainability` interface exporter in the `configuration` and related service identifiers that can be referenced in the dashboard component.

```
struct FlightSearchComponent: Component {
  struct RecommendationFlightSearch: Handler {
    var metadata: Metadata {
      ExecutionModality(.highPerformance)
      ResponseTime(value: 100, unit: 'ms')
      InstanceType(.large)
    }
  }

  var content: some Component {
    Group("flight-search") {
      Group("recommendation") {
        RecommendationFlightSearch()
          .description("Information ...")}
    }
  }
}
```

Listing 2. Annotation of the high-performance version of the Flight Search component in the corresponding Apodini component. The `metadata` annotations allow application developers to annotate all implementation-related information in the code to provide a single source of truth for the sustainability-related information.

show the Apodin-based metadata in the declarative web service annotations.

The *Sustainability Interface Exporter* in Apodini exports the metadata from the web service to a REST API at runtime. The sustainability service metadata structure is shown in Listing 3 in JSON. The set of tools described enables an iterative annotation process. Each time the model and its metadata can be loaded in the tool, new annotations can be added and exported to be included in the microservices implementation. Every time new features are added, the sustainability scores are automatically updated.

```
{
  "services": [
    {
      "id": "flight-search",
      "name": "Flight Search",
      "optional": false,
      "modalities": {
        "highPerformance": {
          "id": "0.0.0.0.1.0",
          "name": "RecommendationFlightSearch",
          "description": "Information ...",
          "responseTime": {
            "value": 100,
            "unit": "ms"
          },
          "instanceType": "large"
        }
      }
    }
  ]
}
```

Listing 3. Elements of the sustainability metadata of the Flight Search service implementation in the JSON format exported by the Apodini sustainability interface exporter.

C. Online Boutique Use Case

In this section, we used the prototype to model the Online Boutique application[8], a microservice-based application used for demo and testing on the Google Cloud platform. This use case shows the applicability of the proposed approach to more complex and realistic applications. The application consists of eight microservices implementing the functionalities of an e-commerce platform. Based on the workflow described in the documentation, the application has been represented in BPMN as shown in Figure 4. The presented tools have been subsequentially used to enrich the application with sustainability features in the *Sustainable Application Design* tool. Two tasks have been labeled as optional (i.e., *Advertisement* and *Product Recommendation*). Additionally, two tasks have been complemented with alternative versions: a low-power version *Product Catalog* has been configured to show a limited set of products during the navigation, the *Product Recommendation* component recommends a reduced set of products using a simpler algorithm. Each component is linked with its implementation through the service identifier, and metadata on the functional and non-functional requirements can be edited in the custom properties panel.

This use case has demonstrated that the approach can be successfully applied to existing applications. Without any need to re-implement existing microservices, these can be enriched and extended using the proposed methodology. The prototype makes it easier for the application designer to understand the current sustainability features implemented and suggests new features to be added.

D. Results Discussion

The instantiation in Section V-A and the subsequent use cases in Section V-B and Section V-C have demonstrated the

[8]https://github.com/GoogleCloudPlatform/microservices-demo

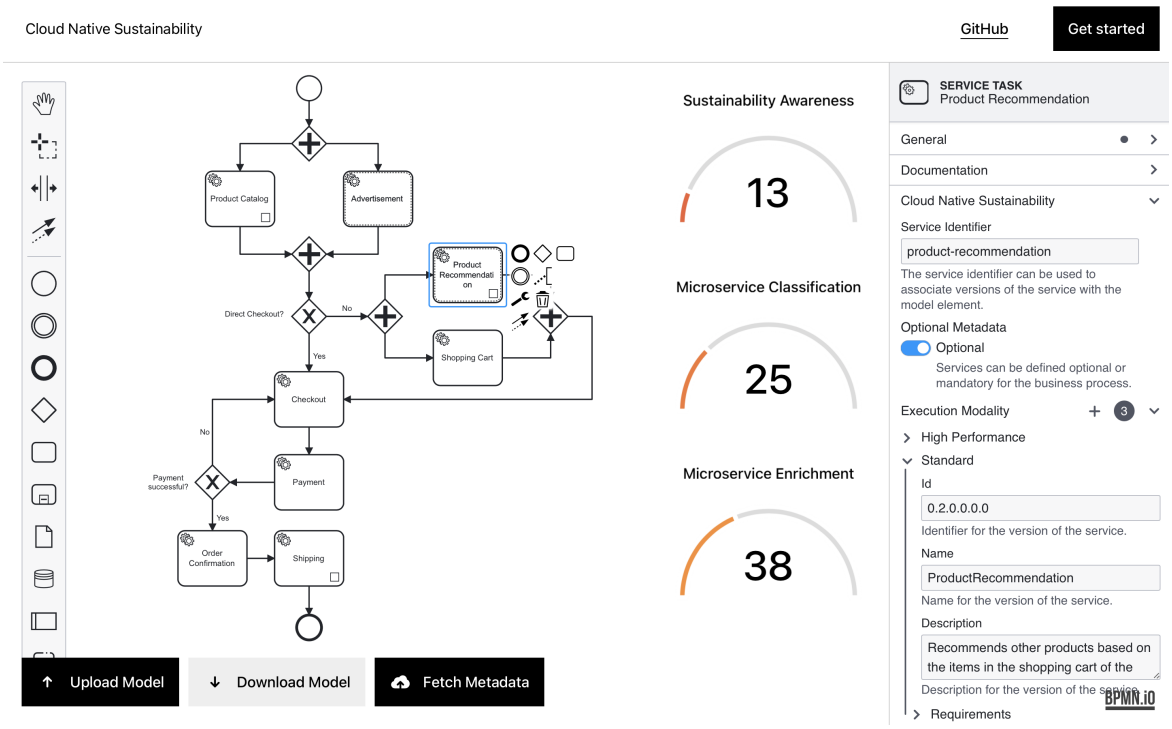

Fig. 4. User Interface of the Sustainable Application Design tool for the Online Boutique application.

feasibility of using existing tools and extending them to support the defined sustainability features. All the code, including the use cases implementation, is available and open source to ensure the reproducibility of the results. We encourage the research community to use and enrich the architectural components.

At the current stage, it is impossible to quantify the benefits of the proposed approach from an environmental sustainability perspective. The actual reduction in energy and emissions depend on several factors, including the involvement of all the stakeholders. However, some preliminary tests have been executed to compare the emissions generated by the applications presented as examples under different execution modalities. In a best-effort approach given the described constraints, we compared the online Boutique application introduced in Section V-C execution of the out-of-the-box application with the execution of the same application with the optional and low-power version of its tasks simulating a workload of 100 users. The results show a reduction of 27% in computational resource demand.

Adopting the presented methodology and architecture can improve IT sustainability with a focus on the application perspective. It can reduce the environmental impact of applications (in terms of energy consumption and emissions) while guaranteeing the required QoS and QoE levels. The features identified to enrich the application design can be complemented with additional aspects that might be identified in the future. However, the presented sustainable design can only affect the environmental impact of cloud-native applications if the synergy between all the stakeholders is enacted and appropriate actions are taken accordingly in their management at runtime.

VI. Conclusion

This paper introduced a Sustainable Application Design Architecture (SADA) to support sustainable applications design, development, and deployment. The approach is based on the SADP, which proposes an incremental process, defining levels of sustainability for the design of cloud-native applications and three features that can be exploited to improve the application energy efficiency. The paper defines the architectural components needed to support the methodology. A prototype for its implementation is provided, showing the feasibility of the approach, and it is tested on two use cases. The SADP methodology is the first step towards energy-aware application management and aims at engaging application owners in the path towards sustainable IT. Thanks to the SADA presented in this paper, the synergy between all the stakeholders involved in the application management is ensured.

In future work, we will extend the current implementation by defining an adaptive and context-aware scheduler to support the sustainable workflow design introduced in this paper. Additionally, we aim to explore the trade-off between carbon footprint reduction, QoS, and QoE to integrate this information in the deployment and execution decision system. We will also consider an additional challenge: the microservices deployment optimization in a heterogeneous fog environment to minimize the energy footprint of applications.

Acknowledgment

This research was partly funded by the European Union (TEADAL, 101070186). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. This research was partly funded by the Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 - Next Generation EU (NGEU).

References

[1] G. Swan, "How green is my cloud?" *CIO*, 2011.

[2] J.-M. Pierson, G. Baudic, *et al.*, "Datazero: Datacenter with zero emission and robust management using renewable energy," *IEEE Access*, vol. 7, pp. 103 209–103 230, 2019.

[3] M. Pedram, "Energy-efficient datacenters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 10, pp. 1465–1484, 2012.

[4] W. Knight, *AI Can Do Great Things — if It Doesn't Burn the Planet*, https://www.wired.com/story/ai-great-things-burn-planet, 2020.

[5] F. Lucivero, "Big data, big waste? a reflection on the environmental sustainability of big data initiatives," *Science and engineering ethics*, vol. 26, no. 2, pp. 1009–1030, 2020.

[6] E. Masanet and A. e. a. Shehabi, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020.

[7] N. Kratzke and P.-C. Quint, "Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study," *Journal of Systems and Software*, vol. 126, pp. 1–16, 2017.

[8] D. Gannon, R. Barga, and N. Sundaresan, "Cloud-native applications," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 16–21, 2017.

[9] CNCF, "Cloud Native Computing Foundation Annual Survey 2021," CNCF, Tech. Rep., 2022. [Online]. Available: https://www.cncf.io/wp-content/uploads/2022/02/CNCF-%20AR%5C_FINAL-edits-15.2.21.pdf.

[10] M. Vitali, "Towards greener applications: Enabling sustainable-aware cloud native applications design," in *International Conference on Advanced Information Systems Engineering*, Springer, 2022, pp. 93–108.

[11] V. Andrikopoulos and P. Lago, "Software Sustainability in the Age of Everything as a Service," in *Next-Gen Digital Services*, ser. Lecture Notes in Computer Science, Springer International Publishing, 2021, pp. 35–47, ISBN: 978-3-030-73203-5. DOI: 10.1007/978-3-030-73203-5_3. [Online]. Available: https://doi.org/10.1007/978-3-030-73203-5_3 (visited on 03/22/2023).

[12] Z. Liu, Y. Chen, *et al.*, "Renewable and cooling aware workload management for sustainable data centers," in *Proc. of the 12th International Conference on Measurement and Modeling of Computer Systems*, 2012, pp. 175–186.

[13] Í. Goiri, W. Katsak, *et al.*, "Parasol and greenswitch: Managing datacenters powered by renewable energy," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 51–64, 2013.

[14] H. Garrett-Peltier, "Green versus brown: Comparing the employment impacts of energy efficiency, renewable energy, and fossil fuels using an input-output model," *Economic Modelling*, vol. 61, pp. 439–447, 2017.

[15] S. S. Gill and R. Buyya, "A Taxonomy and Future Directions for Sustainable Cloud Computing: 360 Degree View," *ACM Computing Surveys*, vol. 51, no. 5, 104:1–104:33, Dec. 18, 2018, ISSN: 0360-0300. DOI: 10.1145/3241038. [Online]. Available: https://doi.org/10.1145/3241038 (visited on 03/22/2023).

[16] M.-T. Thi, J.-M. Pierson, and G. Da Costa, "Game-based negotiation between power demand and supply in green datacenters," in *2020 IEEE International Conference on Big Data and Cloud Computing (BdCloud)*, IEEE, 2020, pp. 690–697.

[17] X. Hu, P. Li, and Y. Sun, "Minimizing energy cost for green data center by exploring heterogeneous energy resource," *Journal of Modern Power Systems and Clean Energy*, vol. 9, no. 1, pp. 148–159, 2021.

[18] N. Gholipour, E. Arianyan, and R. Buyya, "A novel energy-aware resource management technique using joint vm and container consolidation approach for green computing in cloud data centers," *Simulation Modelling Practice and Theory*, vol. 104, p. 102 127, 2020.

[19] A. Benoit, L. Lefevre, *et al.*, "Reducing the energy consumption of large-scale computing systems through combined shutdown policies with multiple constraints," *Int. Journal of High Performance Computing Applications*, vol. 32, no. 1, pp. 176–188, 2018.

[20] M. Acton *et al.*, "2018 Best Practice Guidelines for the EU Code of Conduct on Data Centre Energy Efficiency," *Publications Office of the European Union, Luxembourg, Tech. Report. EUR 29103 EN, 2018*, 2017.

[21] H. Singh *et al.*, "Data center maturity model," *Techn. Ber. The Green Grid*, 2011.

[22] Amazon Web Service, "AWS Well-Architected Framework," AWS, Tech. Rep., 2023. [Online]. Available: https://docs.aws.amazon.com/pdfs/wellarchitected/latest/framework/wellarchitected-framework.pdf.

[23] S. Vos, P. Lago, *et al.*, "Architectural tactics to optimize software for energy efficiency in the public cloud," in *2022 International Conference on ICT for Sustainability (ICT4S)*, Jun. 2022, pp. 77–87. DOI: 10.1109/ICT4S55073.2022.00019.

[24] O. O. Ajibola, T. E. El-Gorashi, and J. M. Elmirghani, "Energy efficient placement of workloads in composable data center networks," *Journal of Lightwave Technology*, vol. 39, no. 10, pp. 3037–3063, 2021.

[25] H. Valera, M. Dalmau, *et al.*, "DRACeo: A smart simulator to deploy energy saving methods in microservices based networks," in *IEEE Int. Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2020, pp. 94–99.

[26] A. Radovanovic, R. Koningstein, *et al.*, "Carbon-aware computing for datacenters," *arXiv preprint arXiv:2106.11750*, 2021.

[27] A. Radovanovic, B. Chen, *et al.*, "Power modeling for effective datacenter planning and compute management," *IEEE Transactions on Smart Grid*, vol. 13, no. 2, pp. 1611–1621, 2021.

[28] A. Nowak, T. Binz, *et al.*, "Pattern-driven green adaptation of process-based applications and their runtime infrastructure," *Computing*, vol. 94, no. 6, pp. 463–487, 2012.

[29] C. Cappiello, M. Fugini, *et al.*, "Business process co-design for energy-aware adaptation," in *ICCP*, IEEE, 2011, pp. 463–470.

[30] J. vom Brocke, R. T. Watson, *et al.*, "Green Information Systems: Directives for the IS discipline," *Communications of the Assoc. for Information Systems*, vol. 33, no. 1, p. 30, 2013.

[31] P. Loos, W. Nebel, *et al.*, "Green it: A matter of business and information systems engineering?" *Business & Information Systems Engineering*, vol. 3, no. 4, pp. 245–252, 2011.

[32] Microsoft Azure, *Azure Emissions Impact Dashboard*, https://www.microsoft.com/en-us/sustainability/emissions-impact-dashboard, 2023.

[33] Google Cloud Platform, *Google Carbon Footprint Console*, https://cloud.google.com/carbon-footprint, 2023.

[34] Amazon Web Service, *AWS Customer Carbon Footprint Tool*, https://aws.amazon.com/aws-cost-management/aws-customer-carbon-footprint-tool, 2023.

[35] K. Lottick, S. Susai, *et al.*, "Energy usage reports: Environmental awareness as part of algorithmic accountability," *arXiv:1911.08354*, 2019.

[36] Thoughtworks Inc., *Cloud Carbon Footprint Tool*, https://www.cloudcarbonfootprint.org, 2023.

[37] R. Brondolin and M. D. Santambrogio, "A black-box monitoring approach to measure microservices runtime performance," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 17, no. 4, pp. 1–26, 2020.

[38] T. Béziers la Fosse, M. Tisi, *et al.*, "Annotating executable dsls with energy estimation formulas," in *Proc. of the 13th ACM SIGPLAN International Conference on Software Language Engineering*, 2020, pp. 22–38.

[39] A. Souza, N. Bashir, *et al.*, "Ecovisor: A virtual energy system for carbon-efficient applications," *arXiv preprint arXiv:2210.04951*, 2022.

[40] J. Schneider, M. Basalla, and S. Seidel, "Principles of green data mining," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019, pp. 1–10.

[41] I. F. De Nardin and R. o. da Rosa Righi, "On revisiting energy and performance in microservices applications: A cloud elasticity-driven approach," *Parallel Computing*, vol. 108, p. 102 858, 2021.

[42] A. Ali and M. M. Iqbal, "A cost and energy efficient task scheduling technique to offload microservices based applications in mobile cloud computing," *IEEE Access*, vol. 10, pp. 46 633–46 651, 2022.

[43] A. Saboor, A. K. Mahmood, *et al.*, "Enabling rank-based distribution of microservices among containers for green cloud computing environment," *Peer-to-Peer Networking and Applications*, vol. 15, no. 1, pp. 77–91, 2022.

[44] A. Papadopoulos, J. Krzywda, *et al.*, "Power-aware cloud brownout: Response time and power consumption control," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 2686–2691. DOI: 10.1109/CDC.2017.8264049.

[45] I. Gerostathopoulos, C. Raibulet, and P. Lago, "Expressing the adaptation intent as a sustainability goal," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '22, New York, NY, USA: Association for Computing Machinery, Oct. 17, 2022, pp. 36–40, ISBN: 978-1-4503-9224-2. DOI: 10.1145/3510455.3512776. [Online]. Available: https://dl.acm.org/doi/10.1145/3510455.3512776 (visited on 03/22/2023).

[46] P. Valderas, V. Torres, and V. Pelechano, "A microservice composition approach based on the choreography of bpmn fragments," *Information and Software Technology*, vol. 127, p. 106 370, 2020.

[47] B. Bruegge and A. H. Dutoit, *Object Oriented Software Engineering Using UML, Patterns, and Java*, 3rd. Prentice Hall, 2010, ISBN: 9780136061250.

[48] OMG, "Business Process Model and Notation (BPMN), Version 2.0," Object Management Group, Tech. Rep., 2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0.

[49] M. Fowler, *Domain-Specific Languages* (Addison-Wesley Signature Series). Pearson Education, 2010, ISBN: 9780132107549.

[50] P. Schmiedmayer, "Apodini: An internal domain specific language to design web services," in *Proc. of the 21st International Middleware Conference Doctoral Symposium*, ser. Middleware'20 Doctoral Symposium, New York, NY, USA: ACM, 2020, pp. 47–49.

[51] P. Schmiedmayer, "Designing evolvable web services," Ph.D. dissertation, Technische Universität München, 2022.