IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 October 2023, accepted 7 December 2023, date of publication 14 December 2023,
date of current version 21 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3342911

RESEARCH ARTICLE

Esfinge Virtual Lab–A Virtual Laboratory Platform With a Metadata-Based API and Based on Dynamic Component

FERNANDO PEREIRA[1,2], DAVID FRANÇA[3], VINICIUS PASCHOAL[4], MARCO NARDES[2], REINALDO R. ROSA[2], AND EDUARDO GUERRA[5]

[1]National Centre for Monitoring and Early Warning of Natural Disasters (CEMADEN), Sã José dos Campos, São Paulo 12247-016, Brazil
[2]National Institute for Space Research (INPE), São José dos Campos, São Paulo 12227-010, Brazil
[3]São José dos Campos Aeronautics Computing Center (CCASJ), Brazilian Air Force (FAB), São José dos Campos 12228615, Brazil
[4]Airspace Control Institute (ICEA), Brazilian Air Force (FAB), São José dos Campos 12228615, Brazil
[5]Faculty of Engineering, Free University of Bozen-Bolzano (UNIBZ), Bozen, 39100 Bolzano, Italy

Corresponding author: Fernando Pereira (fernando.pereira@cemaden.gov.br)

The work of Reinaldo R. Rosa was supported by the São Paulo Research Foundation (FAPESP) under Grant 2021/15114-8.

This work involved human subjects or animals in its research. The authors confirm that all human/animal subject research procedures and protocols are exempt from review board approval.

ABSTRACT The aim of this study is to introduce a virtual laboratory platform with metadata-based API for the creation of dynamic software components. Also, the study seeks to assess the platform's validity and technological acceptance among the community of system developers. The platform is referred to as Esfinge Virtual Lab, and it has been developed utilizing the Java programming language. The platform has a general purpose and is demonstrated in this work in the context of system prototyping. Using declarative programming techniques and a metadata-based APIs, a software developer may focus on business principles and processes rather than expend energy on the retrieval of information or the programming of the user interface. The Esfinge Virtual Lab platform enables developers to efficiently produce diverse forms of data visualization with minimal coding effort. This includes the rapid generation of tables, graphs, maps, and personalized outputs. The components are packaged as Java Archive (JAR) files and uploaded onto the platform. The components that have been loaded can be accessed for execution and can be either injected or dynamically invoked for a new component, thereby facilitating code reuse. This study provides a comprehensive explanation of the platform, including its internal architecture and features. The deployment of the Esfinge Virtual Lab was demonstrated through two practical applications in Brazilian institutions, namely INPE and CEMADEN. A survey was performed to determine the validity of the platform, consisting of open-ended and closed-ended questions. The research used the Technology Acceptance Model (TAM) as a quantitative analytical instrument and thematic analysis as a qualitative analytical approach.

INDEX TERMS VirtualLab, fast prototyping, metadata-based API.

I. INTRODUCTION

During the software development process, it is a prevalent practice for development teams to allocate a significant amount of time towards the implementation of query and data visualization mechanisms within systems with the purpose of serving their users [1], [2]. Occasionally, users who require the system seek to manipulate the data to present it in multiple formats without regard for the final product. This is typically done in the context of prototyping and simulation. In scientific and technological institutions, it is not uncommon for researchers to prioritize the exploration of data presentation through tables, graphs, or maps for analytical purposes rather than initially focusing on defining an operational product. This has been noted in previous literature [3], [4].

In this particular context, the focus is on minimizing the programming effort required for data retrieval and

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Liu.

presentation. Instead, the emphasis is placed on the algorithmic code that defines the business logic of the desired analysis. Furthermore, there is a desire to disseminate these codes among fellow researchers or developers for the purpose of reviewing the outcomes or repurposing them as constituent elements. Nonetheless, the systems that are created are often confined to the researcher's personal computer and are susceptible to being lost, a common occurrence. For example, the absence of amicable methods for code reuse and component sharing can result in redundant efforts and decreased efficacy in the process of prototyping and developing new systems [5]. Typical issues encountered in research establishments such as INPE and CEMADEN.

In light of these challenges, the Esfinge Virtual Lab (EVL) platform was established, as referenced in the present paper. The platform is written in the Java programming language and is part of the Esfinge Framework Project [6]. It is compiled and installed with a built-in application server and published through a web application accessible via HTTP. It allows users to upload dynamic software components created with a small amount of code that contain classes of service or data access classes. Furthermore, it allows reusing components already loaded on the platform. In addition, it allows the presentation of data quickly by providing visualization mechanisms such as graphs, tables, and maps, as well as several front-end features.

The EVL employs declarative programming methodologies that rely on the metadata API [7]. This approach enables users to focus on data analysis and processing while minimizing the need for peripheral tasks such as information retrieval and visualization. In addition, the development of this software utilized innovative techniques in adaptive software architecture that facilitate effortless integration and modification of software components, even during runtime, in an easy and quick way [8], [9], [10].

The platform is general-purpose and contains several hotspots for the extension. It is useful in both development and homologation environments. An evident use in the development environment, for example, is to function as a virtual laboratory ecosystem of programming, allowing prototyping of approaches to be tested for research software conception [11]. By definition, a virtual laboratory ecosystem functions as a "sandbox" where it is possible to carry out tests in an isolated and controlled mode without affecting systems in production [12].

The EVL offers benefits for applications that prioritize interoperability, data sharing, and software component exchange. The application types that can benefit from using EVL are those in situations where there are multiple data sources and several people performing analyses on these data. These analyses can be combined or not. These analyses, processing, and visualizations may be of interest to other people and should be available. Both the data and analysis can be subject to updates, and this revised representation may also be of interest to users.

Examples of instances that encompass the previously mentioned benefits can be observed within the realm of virtual observatories (VOs), as exemplified by the projects DAL [13], JVOSky [14], and TOPCAT [15]. The DAL project outlines criteria for remote data access, allowing customer data analysis software to effectively utilize these services. JVOSky provides a user-friendly interface for astronomical data visualization and manipulation. TOPCAT is a software application for astronomers to evaluate and manipulate tabular data, offering an interactive graphical tool for enhancing source catalog analysis and modification. In contrast to the instances documented in the literature, the concept of EVL is rooted in the utilization of metadata-based APIs.

The EVL software is freely available as open-source code on the online platform known as GitHub, thereby facilitating collaborative efforts among members of the developer community. Product prototypes were rapidly tested in a virtual laboratory ecosystem, as the researchers wanted, during the testing phase at the National Institute for Space Research (INPE) and the National Centre for Monitoring and Early Warning of Natural Disasters (CEMADEN) [16].

A survey was conducted to assess the acceptance and overall impressions of software developers regarding a platform for a comprehensive evaluation. The survey was designed using both closed-ended and open-ended questions. The closed-ended questions were formulated and analyzed using the Technology Acceptance Model (TAM) methodology [17], [18]. The utilization of TAM was chosen by the authors due to its empirical and theoretical foundation, which has been demonstrated in studies where it was used to assess the acceptance of technologies based on user perception [19], [20]. The open-ended questions underwent qualitative analysis using the thematic analysis methodology [21], [22].

This paper offers objectively a comprehensive exposition of the EVL platform, describing the methodologies employed in its creation, its characteristics, and its features. Furthermore, the paper presents empirical case studies and survey findings pertaining to the acceptance and perceptions of the platform, which are subsequently analyzed.

The paper is structured in the following manner: Following this introductory section, Section II provides an overview of the pertinent literature. Section III outlines the platform in detail, while Section IV presents case studies of its implementation at INPE and CEMADEN. In Section V, the design of the survey and the methodology approach for the analysis are explained. In Section VI, the results of the survey are presented, encompassing both quantitative and qualitative analysis. Finally, Section VII offers concluding remarks.

II. BACKGROUND

A. VIRTUAL OBSERVATORIES

The notion of virtual observatories (VOs) was originally conceived for the field of astronomy [23], [24], [25]. The concept entails a collection of instruments and computational

structures that systematize and investigate large quantities of data that are both scattered and subject to frequent changes [26]. VOs facilitate the collaborative access, contribution, and sharing of data and catalogs among multiple users. VOs enable the process of data discovery and are commonly accessed via the Internet.

The utilization of VOs has been observed in multiple fields of science as well as in the realm of information technology. In the field of software engineering research, the integration of VOs with middleware has been explored as a means of processing software components developed by proficient users [27]. The aforementioned components are loaded dynamically and possess the capability to be utilized by multiple users. In addition to conventional resources, VOs are now serving as virtual laboratories, enabling users to develop prototypes within a sandbox-like environment.

B. METADATA-BASED APIS

Components that utilize runtime class metadata to execute their logic can be categorized as components based on the metadata API [28]. The establishment of the aforementioned can be achieved through explicit configuration in annotations or implicitly through adherence to the prescribed naming convention [29], [30]. It is feasible to configure them via external files, databases, or programmatically. The utilization of contemporary software architecture methodology is evident in various frameworks, including but not limited to JUnit, EJB, and JPA [31]. Utilizing a metadata-oriented application programming interface (API) enhances the potential for code reuse and reduces the quantity of code required to encode the intended functionality within the application.

Esfinge Metadata was utilized at EVL to facilitate the creation of metadata-driven APIs [7]. The extensible meta-framework is designed to facilitate the development of metadata-based frameworks by simplifying the process of reading and validating metadata [32]. Its primary objective is to provide assistance in this regard. Furthermore, it has the potential to be utilized in applications that rely on personalized annotations. The retrieval of annotations and collaborative efforts can prove beneficial in guiding developers towards stable code evolution, avoiding increasing complexity and decreasing coupling.

C. TECHNOLOGY ACCEPTANCE MODEL

Many theoretical models aim to forecast the effects of emerging technology on human behavior. The Technology Acceptance Model (TAM) is particularly noteworthy in this context. The primary aim of the TAM is to establish a foundation for monitoring the influence of extrinsic factors on individual subjective beliefs, attitudes, and intentions [33], [34]. The TAM has been extensively employed in various information technology contexts, including but not limited to evaluating cloud computing, blockchain technology, and serious games [35], [36], [37], [38]. The present model offers methodological and statistical assistance for the creation

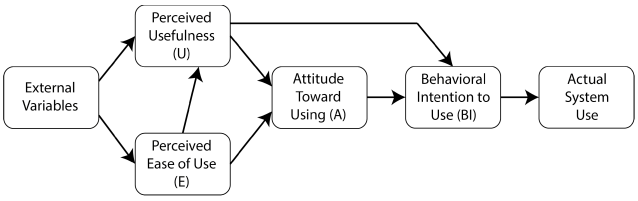


FIGURE 1. Technology Acceptance Model (TAM). Adapted from [33].

of questionnaires aimed at evaluating the acceptance of technology. The application of this approach is versatile, as it can be utilized in various contexts, such as soliciting feedback from developers and end-users regarding the implementation of an original software platform [17], [18]. See Figure 1.

External variables pertain to the characteristics of the technology under scrutiny in the context of acceptance, including but not limited to icons, interfaces, and functionalities. The remaining components within the TAM structure are referred to as constructs, where:

- Perceived Ease of Use (E) - is defined as "the degree to which an individual believes that using a particular system would be free of physical and mental effort." [17, p. 26];
- Perceived Usefulness (U) - is defined as "the degree to which an individual believes that using a particular system would enhance his or her job performance" [17, p. 26];
- Attitude Toward Using (A) - " refers to an individual's degree of evaluative affect toward the target behavior" [39, p. 216];
- Behavioral Intention to Use (BI) - "defined as an individual's subjective probability that he or she will perform a specified behavior" [39, p. 288];
- Actual System Use - is a function of evaluating the frequency of current use or the intention of future use of a system.

D. THEMATIC ANALYSIS

Thematic analysis is a suitable methodology for conducting qualitative analysis on gathered data. According to Braum [21], the approach refers to a systematic technique that involves the identification, analysis, and reporting of patterns or themes within the data. As an example, it is suggested to acquire information through interviews and unstructured responses gathered from survey respondents.

The primary concept involves the process of encoding textual excerpts into codes, subsequently grouping them, and elevating the level of abstraction by defining themes, ultimately culminating in the construction of a model. The process of conducting a comprehensive and systematic analysis of qualitative data to identify and develop themes is commonly referred to as thematic synthesis. See Figure 2.

Cruzes [22] conducted a systematic review within the domain of software engineering and delineated three distinct approaches for conducting the coding process. Initially,

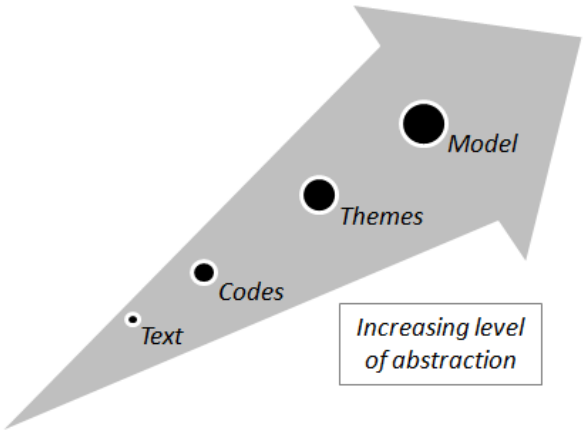


FIGURE 2. Levels of interpretation in thematic synthesis. Adapted from [22].

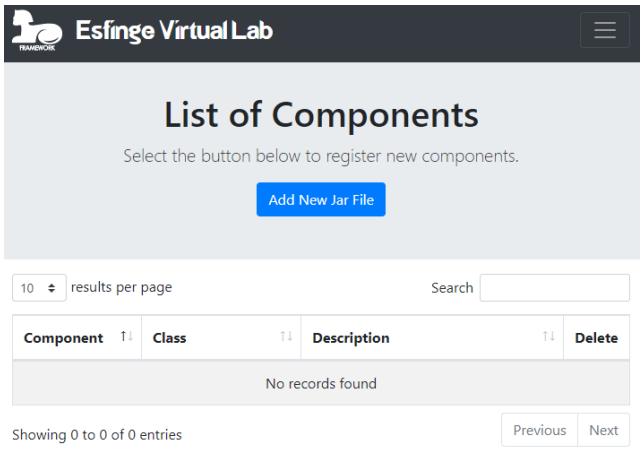


FIGURE 3. EVL initial page.

a deductive approach is employed, whereby textual passages are encoded into pre-established codes contained within a designated list. Secondly, in an inductive manner, researchers do not possess a predetermined list of codes; instead, the codes arise from the code passages. The third form of thematic synthesis adopts an integrated approach that combines elements of the previous two forms. It employs an overall structure to deal with certain aspects while also constructing themes in a deductive manner that progresses from lower to higher levels. This approach has gained prevalence in the literature.

Qualitative analysis themes, in the context of an interview or questionnaire, serve to identify the **strengths** and **weaknesses** that are derived from the responses provided by the respondents. In the context of a new software framework, it is possible to categorize the codes denoting ''Ease of Use'' and ''Hard of Use'' and amalgamate them with other relevant codes under the overarching theme of ''Developer Experience''. Within the domain of software engineering, a rigorous approach is employed to accurately identify codes. This approach involves identifying conceptual codes that pertain to the object being addressed, as well as establishing the relationship between these codes and higher-level codes. Additionally, codes that serve to identify perceptions and specific characteristics are also taken into account [40].

III. THE PLATAFORM

The EVL works as a web-based application that is available for download and installation, catering to the needs of developers, researchers, and end-users alike. The initial page of the platform is depicted in Figure 3. The EVL system facilitates the direct uploading of components by its users, which can subsequently be dynamically loaded during runtime. The aforementioned components possess the capability to establish a connection with pre-existing databases for the purpose of retrieving data and specifying the appropriate formatting. In addition, a component that is loaded becomes accessible to all users.

The utilization of a declarative API based on metadata and annotations has resulted in a reduction of the necessary code for the development of a new component [29], [41]. In addition, within the scope of the Esfinge Framework initiative, there exist other frameworks that can be employed on EVL. One such framework is the Esfinge QueryBuilder [6, p. 12], which allows easy database accessibility through the declaration of an interface featuring methods that adhere to a specific naming convention.

The configuration of the front-end view is determined by annotations that specify the output type produced by each service method or component that has been developed. The utilization of visualization and data recovery techniques enables developers and researchers to focus on the business logic and analytical algorithms.

The EVL platform has a high degree of modularity and consists of a plugin-based architecture. The process of developing and implementing new connectors for diverse database categories and alternative output configurations is an easy one. Upon deployment within the architecture, these entities become readily accessible for utilization by the various components. It is possible to inject or invoke components developed by other developers and utilize them in the creation of a new component.

A. ARCHITECTURE AND INTERNAL ORGANIZATION

The platform's approach is designed to minimize dependencies and promote independent evolution of its components. The customization of new output formats can be achieved through the creation of new annotations, *processors*, and *validators* in the context of pre-existing hotspots. The process of creating new annotations serves to enhance the platform's API by introducing additional capabilities. The *processors* execute the visualization return logic that is designated to annotations, while the *validators* authenticate the source code uploads by utilizing the API to ensure conformity with the framework.

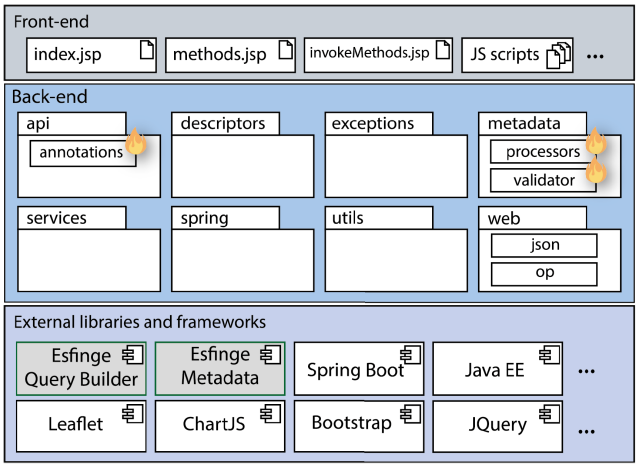


FIGURE 4. Architecture's EVL.

```
//ommited code
@ServiceClass(label = "TableDemo",description =
    "Demonstration")
public class TableDemo {
    private static final List<Task> tasks = new
     ArrayList<>();
    //ommited code
    @ServiceMethod(
        label = "List uncompleted tasks",
        description = "@TableReturn setting the
     headers.")
    @TableReturn(
        fields = {"name", "priority", "completed"},
        headerLabels = {"Name", "Priority", "Done"})
    public List<Task> getOpenTasks() {
        return Utils.filterFromCollection(tasks, t ->
     !t.isCompleted());
    }
    //ommited code
}
```

LISTING 1: Basic example of *@ServiceClass*.

The architecture of the framework is presented in Figure 4, which highlights the primary external libraries and frameworks employed, with a particular emphasis on the Esfinge Query Builder and Esfinge Metadata. Additionally, the diagram showcases the framework modules and their respective components, as well as the front-end pages that constitute the system. The fire icon in the figure indicates the packages that contain hotspots for the extension.

B. COMPONENT LOADING

The EVL platform facilitates the generation of service classes or data access classes. The annotations denoted by *@ServiceClass* and *@ServiceDAO* are at one's disposal for this purpose. The annotation *@ServiceMethod* is utilized to notify the platform of the availability of a method. Finally, the annotated method should be output in an informed format on the front end. The utilization of annotations such as *@TableReturn* can facilitate the presentation of output in a tabular layout, while *@BarChartReturn* and *@LineChartReturn* can be employed to generate charts. Additionally, *@MapReturn* can be utilized to produce maps, and a basic or customized JSON can be defined using *@CustomReturn*. The code snippet provided in Listing 1 pertains to the creation of a class intended for the *TableDemo.java* example.

In the present illustrative example, consider a class called *Task.java*, which includes the properties of *id*, *name*, *priority*, and *completed*, along with their corresponding getters and setters. The provided code snippet refers to the *TableDemo.java* file, which contains a randomized task list utilized for testing. The file contains four distinct methods, namely *getTasksByPriority()*, *getClosedTasks()*, *getAllTasks()*, and *getOpenTasks()*. The Java class named *TableDemo.java* has been annotated with the *@ServiceClass* annotation and has set the values for the "label" and "description" annotation attributes. The method denoted as *getOpenTasks()* in the provided code snippet, along with other methods, has been marked with the *@ServiceMethod* annotation, which includes the specified annotation attributes of "label" and "description". The method was ultimately annotated with the *@TableReturn* annotation and specified the "fields" and "headerLabels" annotation attributes. Having reached this point, the subsequent stage involves making the necessary arrangements for the integration of this particular component onto the platform. The process involves the bundling of the *Task.class* and *TableDemo.class* into a JAR file. This task is simple and can be executed utilizing the Java development Integrated Development Environment (IDE) of choice by the developer.

Upon selecting the "Add New Jar File" button on the platform homepage, the JAR file undergoes a series of validation procedures that encompass various aspects ranging from the file format to the successful loading of individual internal classes. Subsequently, the classes are loaded and their dependencies are verified through the utilization of Java reflection techniques via the Class Loader, with the aim of identifying any pre-existing dependencies on the platform. Subsequently, the metadata pertaining to the class is scrutinized and deciphered, thereby categorizing them into service classes, DAO classes, and classes that possess JPA *@Entity* annotations. The Esfinge Metadata Framework is utilized in this stage to retrieve the annotations of the methods. The Esfinge QueryBuilder framework is utilized for the interpretation of interfaces that adhere to its method name convention. Upon completion of the requisite procedures, the component undergoes dynamic loading of its services onto the platform and subsequently integrates into the service pool. The diagram described in Figure 5 serves to illustrate the aforementioned steps.

The front-end of the platform is depicted in Figure 6, subsequent to the uploading of the JAR file that was generated from the TableDemo component's service methods, as listed on the subpage. The attributes denoted as "label" and "description" within the annotations of "@ServiceClass" and "@ServiceMethod" are utilized for the purpose of recognizing components and their respective services on the front-end.

The procedure for loading data access components remains identical, albeit with the utilization of the *@ServiceDAO*

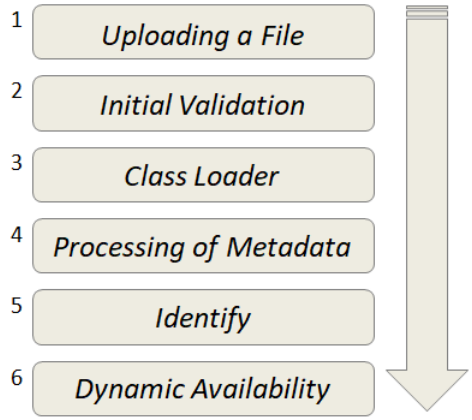


FIGURE 5. Steps for dynamic component loading.

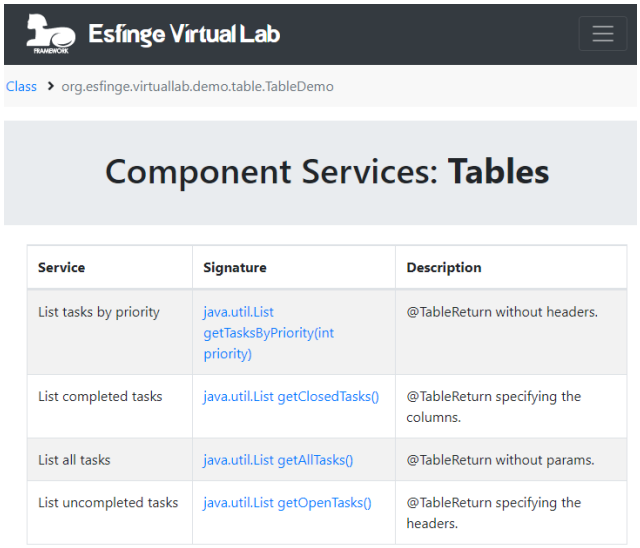


FIGURE 6. Front-end of the platform listing services for a loaded component.

annotation instead of *@ServiceClass*. It is recommended that an interface extending the Esfinge QueryBuilder framework's *Repository* interface be annotated with *@ServiceDAO* in this scenario. As demonstrated in Listing 2, it is possible to generate methods that utilize a naming convention to retrieve data from databases.

C. DATA OUTPUT

The EVL platform enables the visualization of the outcomes obtained from the implementation of service methods. The platform offers automated features for visual representation of data through tables, graphs, maps, and JSON formats. In addition, it is possible to specify different significant parameters for each visualization type in a declarative way using annotation attributes, in order to personalize the output.

The utilization of the *@TableReturn* annotation allows the formatting of the response generated by the service method execution into a table format. This formatting enables the

```
//ommited code
@ServiceDAO(
        label = "DAO",
        description = "Demonstration of @ServiceDAO.",
        url =
    "jdbc:postgresql://localhost:5432/postgres",
        user = "postgres",
        password = "postgres",
        dialect =
    "org.hibernate.dialect.PostgreSQLDialect")
public interface DaoDemo extends Repository<Task> {
    @ServiceMethod(
            label = "List uncompleted tasks",
            description = "Returns by method's name
     convetion")
    @TableReturn
    public List<Task> getTaskByCompleted(boolean
     completed);
    //ommited code
}
```

LISTING 2: Basic example of *@ServiceDAO*.

automatic activation of certain features in the front-end, including but not limited to field ordering, pagination, and search. It is permissible to utilize it in conjunction with data structures such as collections, namely List, Set, and Queue.

The use of *@BarChartReturn* and *@LineChartReturn* annotations enables the visualization of service methods in a format of bar and line graphs, correspondingly. The EVL employs ChartJS, a JavaScript library, for this purpose [42]. The utilization of List<Number>, List<Object>, Map<String, Number> or Map<String, Object> is possible. The chart's properties, including but not limited to colors, category names, axes, labels, and font size, can be customized. The *@LineChartReturn* annotation enables the configuration of multiple datasets and the specification of whether a given set represents a time series.

The *@MapReturn* annotation is utilized to present the result of a service method in the format of a map. The EVL employs Leaflet [43] when combined with OpenStreetMaps [44] in EPSG:4326 projection at the web client for this purpose. It is allowed to be utilized with the collections that represent the markers on the map. The utilization of expression language is a viable option for accessing properties in a practical and declarative way, directly within the annotation attributes.

The *@CustomReturn* function allows the presentation of outcomes in JSON structure, allowing for the specification of desired fields and their corresponding labels. It can be utilized with collections. The aforementioned tables, namely Listing 3, 4, 5, and 6, serve as examples of the utilization of annotations. The aforementioned Figures 7 show examples of utilizing annotations and their corresponding impacts on the front-end execution.

```
@TableReturn(showHeader = false)
public List<Task> getTasksByPriority(int priority)
    //ommited code
```

LISTING 3: Code snippet for the *@TableReturn* annotation.

D. COMPONENT INJECTION

Two annotation options are available for creating a new component that utilizes an already loaded component on the

```
@BarChartReturn(
    dataValuesField = "total",
    dataLabelsField = "region",
    dataColorsField = "color",
    legend = "Number of votes",
    title = "Elections - Votes",
    titleFontSize = 40,
    xAxisLabel = "Votes",
    yAxisLabel = "Region",
    axisFontSize = 30,
    xAxisShowGridlines = true,
    yAxisShowGridlines = false,
    horizontal = true)
public List<Votes> createChartWithAllParameters()
     //ommited code
```

LISTING 4. Code snippet for the *@BarChartReturn* annotation.

```
@LineChartReturn(typeOfChart = "line",
    dataLabels = "eventDateTime",
    temporalSeries = true,
    multipleDataset = true,
    xAxisShowGridlines = true,
    title = "Graph",
    yAxisLabel = "Values", yAxis = {"value"},
    xAxisLabel = "Time", xAxis = {"eventDateTime"})
public List<?> listData(Long header) //ommited code
```

LISTING 5. Code snippet for the *@LineChartReturn* annotation.

```
@MapReturn(
    markerTitle = "${obj.city}",
    markerText = "Min: ${obj.minimum}°C / Max:
     ${obj.maximum}°C")
public List<Temperature>
     listTemperatureOfTheMonth(String month) //ommited
     code
```

LISTING 6: Code snippet for the *@MapReturn* annotation.

```
//ommited code
@ServiceClass(label = "Proxy", description = "@Invoker
    and @Inject.")
public class ProxyDemo {
    @Invoker
    private InvokerProxy invoker;
    @Inject
    private DaoDemo temperatureDAO;

    @ServiceMethod(label = "List all.", description =
     "Using @Invoker.")
    @TableReturn
    public List<Temperature> getTemperatureByInvoker() {
        // Using invoker
        return
     this.invoker.invoke("org.esfinge.virtuallab.demo.dao.
     DaoDemo", "getTemperature", List.class);
        return result;
    }

    ServiceMethod(label = "Temperature chart.",
     description = "Using @Inject.")
    @BarChartReturn
    public Map<String, Number>
     getAverageTemperatureByInject(String month) {
        //Using inject
        List<Temperature> result =
     this.temperatureDAO.getTemperatureByMonth(month);
        Map<String, Number> temp = new HashMap<>();
        result.forEach(t -> temp.put(t.getCity(),
     (t.getMaximum() + t.getMinimum()) / 2));
        return temp;
    }
    //ommited code
}
```

LISTING 7: Code snippet for the *@Inject* annotation.

platform: the *@Inject* and *@Invoker* annotations. The utilization of the *@Inject* annotation is employed in scenarios where the source code of the loaded component is readily accessible, thereby establishing it as a compilation dependency. The *@Invoker* annotation is utilized in situations where the source code of the loaded component is not accessible, necessitating the invocation of service methods at runtime via a dynamic proxy. The aforementioned possibilities ensure the reuse of code, foster a collaborative environment, and diminish the amount of code required for an new product. See Listing 7.

E. ORIGINALITY

The EVL allows the decrease of imperative code required for data processing during the development of a new component by the programmer. The utilization of metadata-based APIs and declarative programming techniques enhances the comprehensibility of the source code. Moreover, the existence of multiple hotspots renders EVL naturally dynamic. Furthermore, the software developer can focus on the operational principles to be integrated into the service method while also reusing preloaded components. The aforementioned attributes are notably prominent in the EVL methodology as compared to alternative approaches such as scientific computing languages (e.g., Python, R Studio) or UI creation frameworks (e.g., Spring Framework).

IV. DEPLOYMENT INSTANTIATIONS

The platform is for general use and has been implemented in case studies within the context of prototyping. Research units such as INPE and CEMADEN have uninterruptedly inserted data into numerous environmental databases. The aforementioned data constitute space-time series that necessitate analysis for the purpose of generating knowledge and facilitating decision-making. The framework demonstrated its efficacy in promoting collaboration in both institutions, enabling the retrieval and presentation of environmental data from relational databases through the creation of graphs, maps, and tables with minimal programming requirements.

The databases utilized for deployment at INPE were sourced from the EMBRACE program, which is responsible for the Brazilian Study and Monitoring of Space Weather. The EMBRACE databases contain substantial amounts of data and continuously receive environmental data from diverse scientific instruments that monitor solar, interplanetary, and high-atmosphere phenomena [45], [46], [47]. As illustrated in Figure 8, a line graph was generated utilizing the EVL technique to scrutinize x-ray data obtained from solar phenomena that were observed.

The deployment at CEMADEN involved the utilization of databases containing data from pluviometric and hydrological stations. Continuous data is acquired from numerous instruments installed throughout Brazil. Furthermore, the platform allows collaboration among operators who possess expertise in the fields of meteorology, hydrology, and geology but lack proficiency in programming. This is achieved by enabling them to create prototypes and develop products with minimal coding through the utilization of the declarative API offered by the framework, which can be customized by institution developers. An instance showcasing the utilization

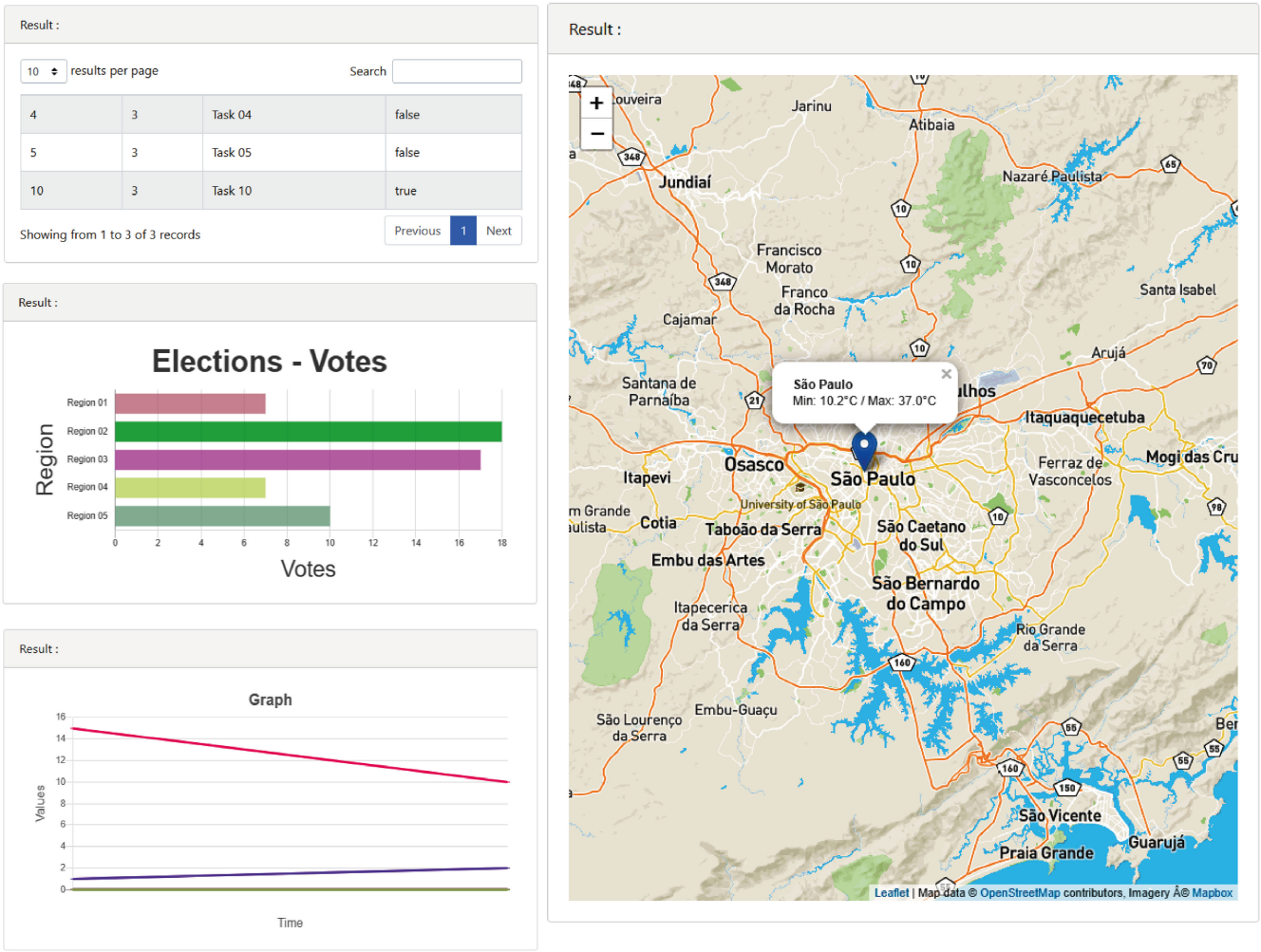


FIGURE 7. Example of the *@TableReturn*, *@BarChartReturn*, *@LineChartReturn*, and *@MapReturn* annotations.

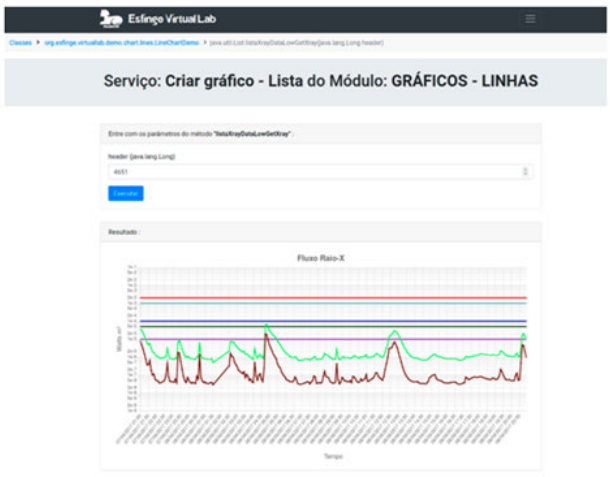


FIGURE 8. EVL platform test at INPE.

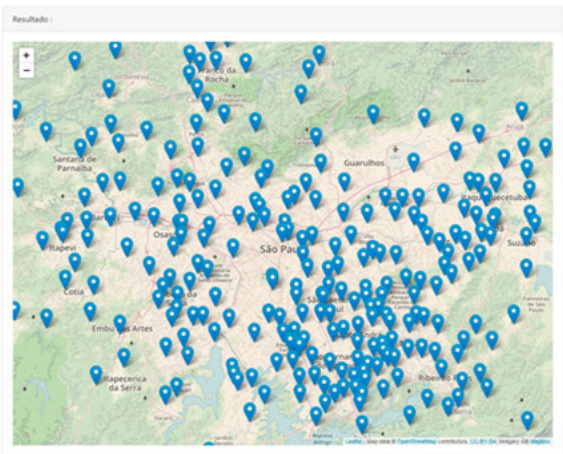


FIGURE 9. EVL platform test at CEMADEN.

of the platform to exhibit a map featuring operational stations from the CEMADEN network is presented in Figure 9.

V. ACCEPTANCE EVALUATION

A. OBJECTIVE

The evaluation of the EVL was conducted with the objective of ascertaining its acceptance among developers and

researchers. The study sought to gather information regarding their perceptions of the platform. Furthermore, the objective was to elicit feedback from participants in order to determine the level of usability and utility of EVL. This data has the potential to provide valuable guidance for informing the development of the platform in subsequent initiatives. This survey specifically utilized the platform with its general applicability for rapid prototyping software components.

B. SURVEY DESIGN

A survey containing open and closed questions in the format of a questionnaire was devised to evaluate the level of acceptance of the platform. The survey was conducted in three modules. The initial module requires the participant to provide confirmation of their willingness to participate in the research for academic purposes and to complete their socio-demographic and technical profiles. The second module introduces the platform to the participant, providing an overview of its overall objectives. The present module is organized into four distinct stages that provide a comprehensive overview of the platform's functionalities. These stages respectively cover the following topics: the API for creating, loading, and executing new components; the API for accessing databases; the API for generating maps and graphics; and the API for injecting and utilizing previously loaded components. The overview is derived from code snippets featuring illustrations with trial data and the utilization of the available annotations. Subsequently, the third module requires the respondent to provide responses to closed questions regarding the platform.

The second module has open-ended questions that seek to elicit overall impressions regarding each presented functionality at every stage. The participant in this particular case possesses the liberty to express their preferred form of feedback, including recommendations, commendations, evaluations, and observations. Likewise, the third module features an open-ended inquiry that prompts the participant to expound upon their comprehensive evaluation of the platform.

The third module of the survey was designed in accordance with the TAM methodology, wherein closed-ended questions were specifically formulated. The closed-ended questions are formulated as declarative statements and employ Linkert's unidimensional scale for the response options, prompting the participant to indicate their level of agreement on a continuum ranging from ''completely disagree'' (denoted as 1) to ''fully agree'' (denoted as 5). A set of six statements (E_x) has been formulated for the *Perceived Ease of Use (E)*, while another set of six statements (U_x) has been developed for *Perceived Usefulness (U)*. Additionally, two statements (A_x) have been created for *Attitude Toward Using (A)*, along with two statements (BI_X) for *Behavioral Intention to Use (BI)*. The *Actual System Use (SU)* framework, designed to assess the frequency of technology use, faces a limitation in measuring the current usage of a platform due to its unavailability to all. In this manner, participants were queried regarding comparable methodologies currently employed. The aforementioned components of the research instrument are elaborated upon in tables 1 and 2.

C. ETHICAL IMPLICATIONS

It is imperative for researchers to contemplate the ethical implications of any procedure implemented in a research endeavor that involves human subjects or sensitive personal information. In cases where risks beyond the minimum threshold are detected, it is imperative to conduct a thorough and appropriate independent ethical evaluation before initiating any research activities. The study's researchers took into account the guidelines of the institutes and universities involved and determined that the potential risks are negligible. Consequently, it did not undergo evaluation by any Institutional Review Board. The necessary steps were implemented to guarantee the preservation of anonymity, the safeguarding of data, and the maintenance of confidentiality. The study ensured that all participants provided informed consent, and their participation was voluntary and anonymous. Furthermore, the study did not pose any potential harm to the participants.

D. DISSEMINATION

The survey was executed and graphically represented through the utilization of Google Forms. The survey hyperlink was disseminated to individuals with professional or academic backgrounds with an invitation to participate in the questionnaire and a request to share it within their respective academic and professional networks. The survey hyperlink was distributed among cohorts of software development experts in both private companies and academic institutions such as CEMADEN and INPE. Particularly, the optional activity was extended to undergraduate and graduate students at UNIBZ, allowing for anonymous participation and without any grading of the responses provided.

E. EVALUATION METHOD

1) QUANTITATIVE EVALUATION

The TAM methodology is utilized to conduct a quantitative evaluation of the survey results of EVL. Through the implementation of this methodology within the context of EVL, yet considering the research instrument specified in the survey design as outlined in section V-B, it is possible to establish six hypotheses. See the Figure 10.

- **H1** - The (E) of the EVL platform is expected to exert a substantial positive influence on the (U).
- **H2** - The (E) of the EVL platform is expected to exert a substantial positive influence on the (A).
- **H3** - The (U) of the EVL platform is expected to exert a substantial positive influence on the (A).
- **H4** - The (U) of the EVL platform is expected to exert a substantial positive influence on the (BI).
- **H5** - The (A) of the EVL platform is expected to exert a substantial positive influence on the (BI).

TABLE 1. Research instrument for EVL according to TAM.

Perceived Ease of Use (E)
$\mathbf{E}_1$ - Learning about the EVL would be easy for me.
$\mathbf{E}_2$ - I would easily know how to use the EVL to create new components.
$\mathbf{E}_3$ - The annotations in the EVL seem easy to use.
$\mathbf{E}_4$ - I could use the EVL in whatever way was most convenient for me.
$\mathbf{E}_5$ - I would gain practice quickly if I could use the EVL to prototype components.
$\mathbf{E}_6$ - I consider EVL annotations easy to use, and navigate the front end to view the list of components and service methods is also easy.
Perceived Usefulness (U)
$\mathbf{U}_1$ - If the EVL platform were available to me, the prototyping activities that I was going to do would be quicker to complete.
$\mathbf{U}_2$ - If I had the platform available to use, my prototyping work would have more performance.
$\mathbf{U}_3$ - If the platform were available to me, my productivity would increase in prototyping work.
$\mathbf{U}_4$ - I think my prototyping work would be more efficient and effective if I used the platform to create new components.
$\mathbf{U}_5$ - I think my prototyping work would be easier if I used the annotations on the platform to develop and share components.
$\mathbf{U}_6$ - Developing software components for prototyping purposes with EVL would be useful for me.
Attitude Toward Using (A)
$\mathbf{A}_1$ - Developing prototypes in a virtual lab environment like EVL is a good idea.
$\mathbf{A}_2$ - I desire to develop prototypes with EVL if it is made available to me.
$\mathbf{A}_3$ - I like the idea of using the EVL in my day-to-day work if it is made available for that.
Behavioral Intention to Use (BI)
$\mathbf{BI}_1$ - If the platform were made available to me, I would try to use it in prototyping activities whenever possible.
$\mathbf{BI}_2$ - If the platform were available to me, I would use its resources to do some of the prototyping activities I needed to do.

TABLE 2. Research instrument for EVL according to TAM: Actual System Use.

Actual System Use (SU)
$\mathbf{SU}_1$ - If the platform were available to me, I would adopt it in the future.
$\mathbf{SU}_2$ - Which of the approaches below do you use, or would you use, to perform prototyping work on a software component?
( ) - Develop each type of visualization and share the code with someone else who wanted to reuse it.
( ) - Send the database to the person who wants to access it or provide remote access to the database.
( ) - Use another framework to create prototypes in a virtual laboratory environment.
( ) - Others.

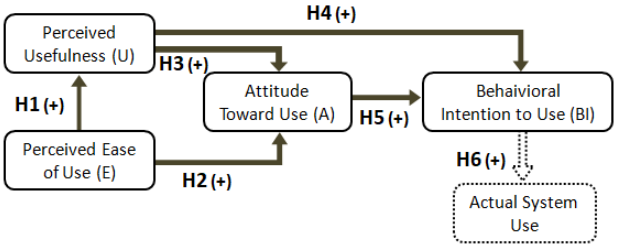


FIGURE 10. Hypotheses about EVL using TAM methodology. Adapted from [33].

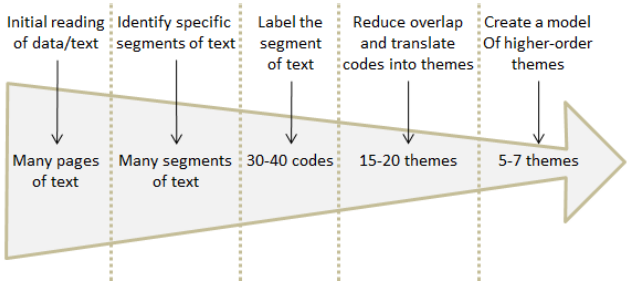


FIGURE 11. Coding process for qualitative analysis. Adapted from [50].

- **H6** - The (BI) of the EVL platform is expected to exert a substantial positive influence on the Actual Use of the platform.

The H6 hypothesis in this study is deemed inapplicable to project EVL, despite its is the open-source and availability for download, it has not yet undergone extensive distribution and dissemination. Consequently, there is a lack of available data to quantify its present usage.

The process of quantitative analysis involves two fundamental stages, namely **the validation of the research instrument** and the **testing of hypotheses**. The validation of the instrument is conducted through the computation of Cronbach's alpha coefficient [48] to demonstrate the internal coherence of the survey responses. The linear regression technique is employed to conduct the hypothesis test in a subsequent approach. Based on the hypotheses test, inferences can be made concerning the constructs.

2) QUALITATIVE EVALUATION

This study employs coding methodology to perform a qualitative evaluation while considering the open-ended, free-response items of the survey. The methodology involves the process of systematically assigning codes to responses by sequentially analyzing each line of the data [49]. In the absence of predetermined codes and themes, the researchers undertake an inductive approach to the process. Through this methodology, the codes emerge through the responses See the Figure 11. In the realm of software engineering, it is advisable to undertake the following procedures: **extract data**, **code data**, and **translate code into themes**.

As delineated in section V-B, this study presents a set of five open-ended questions for the purpose of data collection. The second module of the survey included four open-ended questions that sought to gather participants' overall impressions and rationales pertaining to each EVL API and its respective functionalities. The third module included an open-ended question that requested participants to provide a comprehensive evaluation and a rationale for their assessment of the platform in its entirety. The process of codification is executed on this set of five questions, with the supposition of being capable of deducing comprehensive and particular themes for the platform.

F. THREATS TO VALIDITY

With regard to the design of the survey, closed questions are mandatory for all participants, whose answers are the subject of TAM analysis. However, open questions are not mandatory. This is a possible threat that may, in the thesis, decrease the obtaining of answers for the thematic analysis. The decision was made not to make the questions compulsory, allowing participants to freely contribute with constructive criticism, if they so choose. However, the questionnaire, while indicating non-mandatory, emphasizes in the statement that everyone is strongly advised to answer the open questions, always indicating their impressions of each item demonstrated on the platform.

The TAM model methodology provided a theoretical and empirical background for designing and executing the survey. A threat to the validity of the method is the lack of reliability of the questionnaire elaborated following the model of constructs. This threat can be verified by the Cronbach alpha calculation, which can attest to the consistency of the questionnaire from the answers obtained. Values above 0.8 for each construction indicate good internal consistency. Correlation analysis using statistical methods can also eliminate threats to the validity of the method in the context of the EVL platform.

Regarding thematic analysis, the process of coding is a crucial and thorough undertaking. In order to address potential biases in the data, a collaborative coding approach is employed whereby two researchers work together and a third researcher conducts a review of the coding. This methodology serves to alleviate instances of overlapping and misinterpretation, which can be rectified through iterative efforts by researchers.

VI. SURVEY RESULTS

The survey was distributed through email by providing a hyperlink to the questionnaire and was accessible for a period of one month to collect responses. Upon the conclusion of the designated period, a total of 91 individuals had provided responses to the survey. Four participants responded twice out of the total. The only response taken into account was the most recent one. This way, a total of 87 distinct respondents were considered. The following is an in-depth analysis of the participants' profiles.

A. PARTICIPANTS PROFILE

With respect to the demographic composition of the participants, it can be observed that 84.6% of the respondents are male, 13.2% are female, and 2.2% have refrained from providing their gender information. With respect to nationality, 39.6% of the sample population identify as Brazilian, 54.9% as Italian, and the remaining 5.5% as belonging to other nationalities. In terms of age distribution, nearly half of the sample (49.5%) falls within the 18-24 age range, while a quarter (25.3%) falls within the 25-34 age range. The remaining respondents are distributed as follows: 20.9% are between 35 and 44 years old, and 4.3% are between 45 and 54 years old. With respect to the primary vocation, the majority of respondents, specifically 68.1%, identified themselves as students. A smaller proportion of participants, comprising 15.4%, reported being professional developers, while 13.2% indicated that they were academic researchers. The remaining 3.3% of respondents identified their occupation as something other than the aforementioned categories. In terms of educational attainment, 35.2% of the population possess a high school diploma, while 34.1% have completed an undergraduate degree. A further 24.2% hold a master's degree, with 5.5% having attained a Ph.D. Additionally, 1% of the population have received technical training.

Regarding the technical profile of the participants, it was found that Java is the language in which the majority (72.5%) exhibit proficiency, followed by Web technologies with 64.8%. With respect to programming experience, 28.6% of the respondents reported possessing over three years of experience, while 24.2% of the overall sample reported having 10 to 20 years of experience. In relation to the proficiency level in Java, 13.2% of the participants were classified as Novice, 41.8% as Intermediate, 36.3% as Advanced, and 8.8% as Expert.

The prevalent demographic observed in the collected responses consists of male students aged between 18 and 24 years, possessing moderate proficiency in Java.

B. TAM ANALYSIS

1) VALIDATION OF THE RESEARCH INSTRUMENT

Descriptive statistics and reliability analysis were computed for each construct as defined in Table 1. The obtained Cronbach's alpha values ranged from 0.82 to 0.88, which is deemed satisfactory and appropriate according to Sekaran's [33]. The statistical summary of the responses (ranging from 1 to 5) gathered for the statements in the comprehensive evaluation of EVL is presented in Table 3. The table includes the mean, standard deviation, and alpha coefficient, calculated by SPSS software [51]. The statistical data is presented by construct.

The results of the descriptive statistical analysis indicate that there was a significant perception of ease of use, as evidenced by a mean score of 4.26 in relation to the variable of *Perceived Ease of Use*. The obtained score indicates

TABLE 3. Mean, standard deviation and Cronbach's alpha.

Constructs	$(\bar{x})$	(σ)	(α)
Perceived Ease of Use (E)	4,26	0,56	0,82
Perceived Usefulness (U)	3,84	0,67	0,88
Attitude Toward Using (A)	3,89	0,72	0,83
Behavioral Intention to Use (BI)	3,74	0,81	0,84

TABLE 4. Correlation analysis for E, U, A and BI.

		E	U	A	BI
E	Pearson Correlation	1	,644**	,536**	,459**
U	Pearson Correlation	,644**	1	,700**	,610**
A	Pearson Correlation	,536**	,700**	1	,646**
BI	Pearson Correlation	,459**	,610**	,646**	1
The correlation is significant at the 0.01 level (two-tailed).					

that the participants perceived the platform as user-friendly. Regarding the constructs of *Perceived Usefulness*, *Attitude Toward Using*, and *Behavioral Intention to Use*, it can be observed that when the score is equal to or exceeds 3.74, there is an inclination towards agreement that the platform is beneficial and there exists a desire to utilize it.

Regarding the employment of the *Actual System Use*, 81.1% of the participants indicated that in order to carry out software component prototyping tasks, they create various forms of visual representations and manually distribute the code to others who may wish to utilize it.

2) TESTING OF HYPOTHESES

Added to the reliability analysis, the correlation analysis between the constructs is carried out. The correlation was significant and positive according to Table 4, calculated via SPSS software. Thus it is possible to proceed with the hypothesis test as expected by the TAM methodology according to section V-E1.

Following the assessment method illustrated in the Figure 10, we proceed with the test of the hypothesis H1, H2, H3, H4 and H5 through linear regression analysis. See Table 5.

Based on the results, it is evident that all tested hypotheses receive strong support. There is a significant positive influence on all respective hypotheses, as expected by the TAM methodology. The value of Sig. is <0,01 for the statistical model in all hypotheses. It is possible to exclude the null hypothesis. In detail, we concluded that E explains 41.5% of U, E explains 28.7% of A, U explains 49,0% of A, and A explains 41.7% of BI.

3) DISCUSSION

In summary of the quantitative analysis it is possible to conclude that EVL was considered easy to use supported by

TABLE 5. Summary of hypothesis testing through linear regression analysis.

Hypothesis	Results			
	R^2	Beta	F(1,85)	Sig.
H1	,415	,644	60,349	<0,01
H2	,287	,536	34,288	<0,01
H3	,490	,700	81,698	<0,01
H4	,372	,610	50,293	<0,01
H5	,417	,646	60,844	<0,01

Perceived Ease of Use mean. *Perceived Usefulness*, *Attitude Towards Using* and *Behavioral Intention to Use* had an mean very close to 4, which can be considered as positive points for EVL, but suggesting that improvements can be made. However, considering the platform as a technology still under development, the trends can be considered encouraging for the continued development of the platform.

C. THEMATIC ANALYSIS

1) DATA EXTRACTION

The initial phase of the thematic analysis involved the execution of the data extraction procedure. The survey received responses from 87 different individuals, who were all anonymously counted from P1 to P87. 9 people (10.3%) completed none of the 5 open questions. 55 people (63.2%) answering all of them, 8 people (9.1%) answering between one and two questions, and 15 people (17.2%) answered between three and four questions. At least one of the open questions was answered by 78 people (89,6%).

2) CODING DATA

All answers were placed on a spreadsheet, with columns for Q1 to Q5 questions according to the order of appearance of the questions opened in the survey and lines for participants from P1 to P87. From this point on, the data coding process has begun. The coding process was conducted question by question for all participants by identifying one or more codes for each of the 87 answers.

To illustrate the process, see some remarkable examples of response text: from participant P43 to Q5 - *"I really like it! Looks promising, easy to use and practical, easy to use and practical."*, In this fragment, we can directly infer the code **"Easy to Use"**. In P68ś response to Q3 – *"The idea to have a visualization of data in such an easy way is something that would benefit a lot of programmers. So, it looks very nice to me."* , adds the code **"Very helpful"**. For the participant P69 to Q2 - *"In the IDE you don't have type checking for method names. This can make it difficult to find errors."* adds the code **"Hard to debugging"**.

3) TRANSLATE CODES INTO THEMES

The translate procedure involved categorizing all identified codes as either **strengths** or **weaknesses** of the EVL platform

TABLE 6. Themes obtained from the thematic analysis.

Theme: Developer Experience		
	Codes	Total
Strenghts	Good Code Reusability	24
	Faster Development	34
	Facilitated Extensibiblity	3
	Clear Code	2
	Easy to Use	59
	Increase Produtivity	19
	Very useful	29
Weakness	Hard to Use	8
	Not Useful	8
Theme: Documentation		
	Codes	Total
Strenghts	Good Documentation	2
Weakness	Poor Documentation	6
Theme: Goals		
	Codes	Total
Strenghts	For Data-driven	2
	For Requirements Gathering	1
	For Proof of Concepts	1
	For Complex Tasks	1
Weakness	*Not applicable*	
Theme: Learning Curve		
	Codes	Total
Strenghts	Ease to Learn	26
Weakness	Hard to Learn	5
Theme: Limitations		
	Codes	Total
Strenghts	*Not applicable*	
Weakness	General Limitations	29
	Limited Customization	16
Theme: Metadata		
	Codes	Total
Strenghts	Good Conventions	6
	Clean Annotations	17
Weakness	Bad Conventions	20
	No Clean Annotations	9
Theme: User Experience		
	Codes	Total
Strenghts	Good Upload Components Process	10
	Intuitive UI	1
Weakness	Bad Upload Components Process	27
	Limited Front-end Functions	12

and subsequently grouping them under relevant top-level themes. The ensuing Table 6 presents the outcome of the aforementioned translation and reduction of overlap.

4) DISCUSSION

The feedback provided by the respondents was favorable towards the theme of **Developer Experience**. Evidently, a significant proportion of the respondents assert that the platform is easy to use and very useful. There was limited discussion regarding the subjects of **Documentation** and **Goals**. Regarding the **Learning Curve** theme, it has been observed that around 30% of participants acknowledged the platform as being easy to learn. Regarding the theme of **Limitations**, it was observed that 33% of the participants hold the view that the platform still exhibits both general and specific limitations. The **Metadata** theme highlights that the platform's annotations were considered clear, but bad conventions were often cited. In conclusion, with respect to the **User Experience** theme, it was deemed necessary to enhance the process of uploading components.

The responses provided by the participants reveal that the platform constraints are primarily related to their preference for personalized annotation options, the ability to link new attributes, and the customization of the user interface. The aforementioned constraints are frequently referenced in relation to the API for graphs and maps on the EVL. The API for creating database access methods is often associated with the most frequently cited bad conventions, primarily due to the participants' perception that declarative programming alone may not suffice for constructing complex methods. Regarding the procedure for uploading new components, several contributors propose exploring alternative methods for executing this task. One potential approach involves integrating the functionality directly into the user interface via a live editor, rather than relying on a JAR file.

VII. CONCLUSION

The EVL platform was explained in this study, which detailed its architecture and internal organization, component loading, data output annotations, and component injection. The platform was applied to real databases with large volumes of data through case studies conducted by INPE and CEMADEN. Furthermore, the authors conducted a study involving 87 individuals to validate the acceptance of the platform. The theoretical and statistical support for the development and analysis of the research included the use of the Technology Acceptance Model (TAM) for quantitative evaluation and thematic analysis for qualitative evaluation.

The platform was deemed user-friendly, easy to learn, and highly useful by survey respondents, as evidenced by both quantitative and qualitative evaluations. Thematic analysis was able to demonstrate the limitations of the platform and multiple possibilities for potential improvement in order to enhance both the user and developer experience. Given that the platform is still evolving and maturing, the results have demonstrated promise.

The future work aims to advance the platform to retrieve and visualize data from various types and storage paradigms. Furthermore, it is expected that transparent persistence can be achieved for the developer even when the data originates from other sources such as NoSQL, ShapeFiles, TIFF images, and related or complementary relational databases. Another interest is to conduct usability experiments with developers

to improve the APIs, making them more user-friendly and practical with good conventions and documentation. Finally, given that the components are dynamically loaded, future considerations will be given to designing resources that support code generation based on visual programming tools.

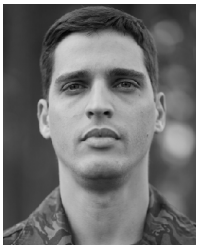
A future application of Esfinge Virtual Lab is to accelerate the development process of digital platforms in EdTech directed to the areas of health [52] and space medicine [53], [54] in the context of the ARTEMIS mission [55].

REFERENCES

[1] M. Morisio, I. Stamelos, V. Spahos, and D. Romano, "Measuring functionality and productivity in web-based applications: A case study," in *Proc. 6th Int. Softw. Metrics Symp.*, 1999, pp. 111–118.
[2] E. Mendes, N. Mosley, and S. Counsell, "A comparison of length, complexity & functionality as size measures for predicting web design & authoring effort," in *Proc. EASE Conf.*, 2001, pp. 1–14.
[3] M. L. Bernardi, G. A. Di Lucca, and D. Distante, "A model-driven approach for the fast prototyping of web applications," in *Proc. 13th IEEE Int. Symp. Web Syst. Evol. (WSE)*, Sep. 2011, pp. 65–74.
[4] J. Arnowitz, M. Arent, and N. Berger, *Effective Prototyping for Software Makers*. Amsterdam, The Netherlands: Elsevier, 2010.
[5] V. R. Basili, L. C. Briand, and W. L. Melo, "How reuse influences productivity in object-oriented systems," *Commun. ACM*, vol. 39, no. 10, pp. 104–116, Oct. 1996.
[6] E. Guerra, "Designing a framework with test-driven development: A journey," *IEEE Softw.*, vol. 31, no. 1, pp. 9–14, Jan. 2014.
[7] E. Guerra, P. Lima, J. Choma, M. Nardes, T. Silva, M. Lanza, and P. Meirelles, "A metadata handling API for framework development: A comparative study," in *Proc. 34th Brazilian Symp. Softw. Eng.*, New York, NY, USA, 2020, pp. 499–508.
[8] I. Welch and R. J. Stroud, "KAVA: Using byte code rewriting to add behavioural reflection in Java," in *Proc. Coots*, vol. 1, 2001, pp. 119–130.
[9] Y. Li, T. Tan, and J. Xue, "Understanding and analyzing Java reflection," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, no. 2, pp. 1–50, Feb. 2019.
[10] P. Lima, E. Guerra, M. Nardes, A. Mocci, G. Bavota, and M. Lanza, "An annotation-based API for supporting runtime code annotation reading," in *Proc. 2nd ACM SIGPLAN Int. Workshop Meta-Programming Techn. Reflection*, New York, NY, USA, Oct. 2017, pp. 6–14.
[11] E. Ciepiela, D. Harezlak, J. Kocot, T. Bartynski, M. Kasztelnik, P. Nowakowski, T. Gubala, M. Malawski, and M. Bubak, "Exploratory programming in the virtual laboratory," in *Proc. Int. Multiconference Comput. Sci. Inf. Technol.*, Oct. 2010, pp. 621–628.
[12] B. C. Lucas, J. A. Bogovic, A. Carass, P.-L. Bazin, J. L. Prince, D. L. Pham, and B. A. Landman, "The Java image science toolkit (JIST) for rapid prototyping and publishing of neuroimaging software," *Neuroinformatics*, vol. 8, no. 1, pp. 5–17, Mar. 2010.
[13] *Ivoadal*, IVOA, Canberra, Australia, 2023.
[14] *Jvosky3*, JVO, Tokyo, Japan, 2023.
[15] F. Rothmaier, S. Gabriel, and M. Demleitner, "Data discovery using the virtual observatory registry," German Astrophysical Virtual Observatory (GAVO), Berlin, Germany, Tech. Rep., 2023.
[16] P. Pinho, S. Finco, and C. Pinho, "Overview of the Brazilian centre for natural disaster monitoring and alerts (CEMADEN)," in *Proc. 5th Int. Conf. Manag. Emergent Digit. EcoSystems*, Oct. 2013, pp. 302–305.
[17] F. D. Davis, "A technology acceptance model for empirically testing new end-user information systems: Theory and results," Ph.D. thesis, Massachusetts Inst. Technol., Cambridge, MA, USA, 1985.
[18] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, "User acceptance of computer technology: A comparison of two theoretical models," *Manag. Sci.*, vol. 35, pp. 982–1003, Aug. 1989.
[19] P. Silva, V. Pimentel, and J. Soares, "A utilizaçao do computador na educaçao: Aplicando o technology acceptance model (TAM)," *Biblionline*, vol. 8, pp. 263–272, Sep. 2012.
[20] G. Rigopoulos and D. Askounis, "A TAM framework to evaluate users - perception towards online electronic payments," *J. Internet Banking Commerce*, vol. 12, no. 3, pp. 1–6, 1970.
[21] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Res. Psychol.*, vol. 3, no. 2, pp. 77–101, Jan. 2006, doi: 10.1191/1478088706qp063oa.
[22] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Proc. Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2011, pp. 275–284.
[23] *Japonese Virtual Obsevatory*, JVO, Tokyo, Japan, 2023.
[24] *All-Sky Virtual Observatory*, ASVO, Canberra, Australia, 2023.
[25] *IVOA.Net*, IVOA, Canberra, Australia, 2023.
[26] S. G. Djorgovski and R. Williams, "Virtual observatory: From concept to implementation," 2005, *arXiv:ph/0504006*.
[27] R. D. C. dos Santos, L. A. R. Correa, E. M. Guerra, and N. L. Vijaykumar, "A private cloud-based architecture for the Brazilian weather and climate virtual observatory," in *Proc. Int. Conf. Comput. Sci. Appl.* Cham, Switzerland: Springer, 2013, pp. 295–306.
[28] E. Guerra, "Design patterns for annotation-based APIs," in *Proc. 11th Latin-American Conf. Pattern Lang. Program.*, 2016, pp. 1–14.
[29] E. M. Guerra, J. T. de Souza, and C. T. Fernandes, "A pattern language for metadata-based frameworks," in *Proc. 16th Conf. Pattern Lang. Programs*, Aug. 2009, pp. 1–29.
[30] E. Gomes, E. Guerra, P. Lima, and P. Meirelles, "An approach based on metadata to implement convention over configuration decoupled from framework logic," Tech. Rep., 2023. [Online]. Available: https://www.authorea.com/doi/full/10.22541/au.168067455.55373151
[31] *JSRS by Platform*, Oracle, Austin, TX, USA, 2023.
[32] E. Guerra, F. Alves, U. Kulesza, and C. Fernandes, "A reference architecture for organizing the internal structure of metadata-based frameworks," *J. Syst. Softw.*, vol. 86, no. 5, pp. 1239–1256, May 2013.
[33] Y.-W. Sek, S.-H. Lau, K.-K. Teoh, C.-Y. Law, and S. B. Parumo, "Prediction of user acceptance and adoption of smart phone for learning with technology acceptance model," *J. Appl. Sci.*, vol. 10, no. 20, pp. 2395–2402, Oct. 2010.
[34] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, "User acceptance of information technology: Toward a unified view," *MIS Quart.*, vol. 27, no. 3, pp. 425–478, 2003.
[35] H. C. Hedler, E. Ferneda, B. S. Duarte, H. A. do Prado, and C. E. C. Gutierrez, "Aplicaçao do modelo de aceitaçao de tecnologia à computaçao em nuvem," *Perspectivas em Gestão & Conhecimento*, vol. 6, no. 2, pp. 188–207, 2016.
[36] A. G. Kern, "Blockchain Technology: A technology acceptance model (TAM) analysis," Ph.D. thesis, Católica Lisbon-Bvsiness & Economics, Lisboa, Portugal, 2018.
[37] A. Yusoff, R. Crowder, and L. Gilbert, "Validation of serious games attributes using the technology acceptance model," in *Proc. 2nd Int. Conf. Games Virtual Worlds Serious Appl.*, Mar. 2010, pp. 45–51.
[38] A. M. Aburbeian, A. Y. Owda, and M. Owda, "A technology acceptance model survey of the metaverse prospects," *AI*, vol. 3, no. 2, pp. 285–302, Apr. 2022.
[39] I. Ajzen and M. Fishbein, "Attitude-behavior relations: A theoretical analysis and review of empirical research," *Psychol. Bull.*, vol. 84, no. 5, pp. 888–918, Sep. 1977.
[40] D. S. Cruzes, T. Dyba, P. Runeson, and M. Höst, "Case studies synthesis: A thematic, cross-case, and narrative synthesis worked example," *Empirical Softw. Eng.*, vol. 20, no. 6, pp. 1634–1665, Dec. 2015.
[41] R. Teixeira, E. Guerra, P. Lima, P. Meirelles, and F. Kon, "Does it make sense to have application-specific code conventions as a complementary approach to code annotations?" in *Proc. 3rd ACM SIGPLAN Int. Workshop Meta-Programming Techn. Reflection*, Nov. 2018, pp. 15–22.
[42] J. S. Chart, "Chart.js | open source html5 charts for your website," Toronto, ON, Canada, Tech. Rep. 2.8.0, 2023.
[43] Leaflet, *An Open-Source Javascript Library for Interactive Maps*, Kyiv, Ukraine, 2023.
[44] OpenStreetMap, *Openstreetmap*, London, U.K., 2023.
[45] N. Sant'Anna, E. Guerra, A. Ivo, F. Pereira, M. Moraes, V. Gomes, and L. G. Veras, "Modelo arquitetural para coleta, processamento e visualizaçao de informaçoes de clima espacial," in *Anais Principais do X Simpósio Brasileiro de Sistemas de Informação*. Porto Alegre, Brazil: Sociedade Brasileira de Computação, 2014, pp. 125–136.
[46] C. M. Denardini, S. Dasso, and J. A. Gonzalez-Esparza, "Review on space weather in Latin America. 3. Development of space weather forecasting centers," *Adv. Space Res.*, vol. 58, no. 10, pp. 1960–1967, Nov. 2016.
[47] T. B. Veronese, R. R. Rosa, M. J. A. Bolzan, F. C. R. Fernandes, H. S. Sawant, and M. Karlicky, "Fluctuation analysis of solar radio bursts associated with geoeffective X-class flares," *J. Atmos. Solar-Terrestrial Phys.*, vol. 73, nos. 11–12, pp. 1311–1316, Jul. 2011.
[48] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, Sep. 1951.
[49] J. Thomas and A. Harden, "Methods for the thematic synthesis of qualitative research in systematic reviews," *BMC Med. Res. Methodol.*, vol. 8, no. 1, pp. 1–10, Dec. 2008.

[50] J. W. Cresswell, *Educational Research : Planning, Conducting, and Evaluating Quantitative and Qualitative Research*, 4th ed. Boston, MA, USA, 2012.
[51] A. Buhl, *SPSS 22*. London, U.K.: Pearson, 2014.
[52] E. A. S. Tomazini, L. Tobase, S. V. Teodoro, H. H. C. Peres, D. M. D. Almeida, and D. C. Alavarce, "Online course on advanced life support in cardiorespiratory arrest: Innovation for continuing education," *Revista da Rede de Enfermagem do Nordeste*, vol. 19, Oct. 2018, Art. no. e32444.
[53] C. Krittanawong, N. K. Singh, R. A. Scheuring, E. Urquieta, E. M. Bershad, T. R. Macaulay, S. Kaplin, C. Dunn, S. F. Kry, T. Russomano, and M. Shepanek, "Human health during space travel: State-of-the-art review," *Cells*, vol. 12, no. 1, p. 40, Dec. 2022.
[54] A. Lara, A. Borgazzi, O. Mende Jr., R. R. Rosa, and M. O. Domingues, "Short-period fluctuations in coronal mass ejection activity during solar cycle 23," *Sol. Phys.*, vol. 248, no. 1, pp. 155–166, Mar. 2008.
[55] A. Halford, S. Savage, B. Walsh, S. Christe, A. Pulkkinen, P. O'Brien, and A. Young, "Artemis, gateway, the return to the moon and forward to Mars for heliophysics," in *Bulletin of the American Astronomical Society*. Washington, DC, USA: American Astronomical Society, 2022.

FERNANDO PEREIRA received the bachelor's degree in computer science from the Federal University of Ouro Preto (UFOP), in 2008, and the master's degree in applied computing from the National Institute of Space Research (INPE), in 2013, where he is currently pursuing the Ph.D. degree in applied computing. He is also an Information Technologist with the National Centre for Monitoring and Early Warning of Natural Disasters (CEMADEN). He has expertise in the domain of software engineering, particularly in java development, mobile device development, database management, and systems architecture.



DAVID FRANÇA received the M.Sc. degree from the National Institute for Space Research (INPE), Brazil, with a focus on web systems interoperability architecture. He is currently a Software Architect with the Brazilian Air Force and a Solutions Consultant for private companies in systems integrations and development process. His research interests include API design and automatic code generation.

VINICIUS PASCHOAL is currently a System Analyst and a Developer with the Airspace Control Institute (ICEA), Brazilian Air Force.

MARCO NARDES received the Graduate degree in information systems from Universidade Regional Integrada do Alto Uruguai e das Missões—Campus de Santo Ângelo. He is currently a former Research Fellow with the Laboratory for Computing and Applied Mathematics, National Institute for Space Research (INPE), worked with agile methods, software patterns, and framework development.

REINALDO R. ROSA is currently a Researcher with the Laboratory for Computing and Applied Mathematics, National Institute for Space Research (INPE), Brazil. He developed his Ph.D. research with the Advanced Visualization Laboratory, University of Maryland, in computational astrophysics and his postdoctoral research in computational astrophysics with Nagoya University. He also works in EdTech focused on the application of computer vision as a functionality in digital platforms. His main research interests include AI and data analytics from big data in space and environmental sciences.

EDUARDO GUERRA received the Ph.D. degree in computer engineering from the Aeronautics Institute of Technology. He is currently a Researcher with the Free University of Bozen-Bolzano. He was a Researcher with the National Institute for Space Research, Brazil, and a Teacher with the Aeronautics Institute of Technology. His research interests include agile methods, software patterns, framework development, software analytics, and dynamic architectures. He has practical experience in architecture and framework design.

Open Access funding provided by 'Libera Università di Bolzano' within the CRUI CARE Agreement