IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 November 2024, accepted 1 December 2024, date of publication 11 December 2024,
date of current version 27 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3514801

APPLIED RESEARCH

Using Logs to Reduce the Impact of Process Variability and Dependence on Practitioners in Requirements Engineering for Traditional Business Process Automation Software

THIAGO MEDEIROS DE MENEZES[1,2] **AND ANA CAROLINA SALGADO**[2], **(Member, IEEE)**
[1]Sidia, Manaus, Amazonas 69055-035, Brazil
[2]CESAR School, Recife, Pernambuco 50030-390, Brazil

Corresponding author: Thiago Medeiros de Menezes (thiago.menezes@sidia.com)

ABSTRACT Context: Business Process Automation (BPA) is adopted by organizations to improve efficiency, reduce costs, and increase overall business performance. Traditional Business Process Automation (TBPA) is one of the main approaches employed to develop a BPA. TBPA entails the development of BPA in a programming language to integrate relevant applications in the digital ecosystem to execute a given process. Process variability and practitioner unavailability are issues that encumber the requirements engineering for TBPA software. Objective: This work proposes Requirements with Logs (RWL), a log-based approach for TBPA software to reduce the impact of these issues, by providing a higher alignment among business process requirements and software architecture, and employing process mining to semi-automatically discover the business process during requirements elicitation. Method: The research conducted a case study in a technology institute to assess RWL and report its results in practice. Results: The results revealed significant improvements in adaptability to business process changes, in time spent with practitioners, and in software development. RWL also presented limitations, including human intervention to accurately obtain the business process, complexity to trace the process into the architecture, data privacy concerns, and risk of network traffic overload. Conclusion: This research demonstrated the effectiveness of RWL to minimize the impact of process variability and the dependence on practitioners.

INDEX TERMS Process variability, practitioner unavailability, business process automation, requirements engineering, software architecture, software design, software development, software engineering.

I. INTRODUCTION

Business Process Automation (BPA) is widely adopted by organizations to improve efficiency, reduce costs, and increase overall business performance [4], [9], [12], [23], [25], [32], [40], [47], [48]. BPA software has improved business processes in auditing firms, banks, outsourcing providers, public entities, software industry, and telecom companies [48].

BPA software performs the digital labor of human beings to automate business processes in a particular digital ecosystem [32], [48]. It must consider a variety of information systems and applications with different ages, features, compatibilities, interfaces, and data [32]. Whereas general software is an arrangement of operations designed to aid users in performing tasks, BPA software entails integrating several applications to execute related tasks for one or more processes [48].

There are three main approaches to developing BPA: (1) Traditional Business Process Automation, (2) Robotic Process Automation, and (3) Hyperautomation [48]. Traditional Business Process Automation (TBPA) is the traditional approach, which entails developing BPA software in a programming language for integrating relevant applications in the digital ecosystem to execute a given process [16], [48]. This study considers Business Process Management

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

System (BPMS) a type of TBPA, once TBPA has a broader definition. Robotic Process Automation (RPA) uses software robots (also called agents, bots, or workers) to emulate human-computer interaction for executing a combination of processes, activities, transactions, and tasks in one or more unrelated software systems [4], [9], [32], [48]. This work does not address Robotic Desktop Automation (RDA) specifically as an approach, once it is closely related to RPA. The former focuses on automating tasks in an unattended manner, streamlining back-office processes without human intervention. In contrast, the latter is designed to assist humans by automating tasks in a more attentive manner, particularly in front-office functions. Despite their different applications, both RPA and RDA are grounded in similar concepts and objectives [32]. Hyperautomation (HA), Intelligent Automation (IA), Intelligent Process Automation (IPA), Integrated Automation Platform (IAP), and Cognitive Automation (CA) are the given different names for the technology that combines Business Process Automation, Artificial Intelligence (AI), and Machine Learning (ML) to discover, validate, and execute organizational processes automatically with no or minimal human intervention [16], [48].

Fig. 1 presents the applicability assessment for each approach on the basis of the frequency of process tasks and the process variability.

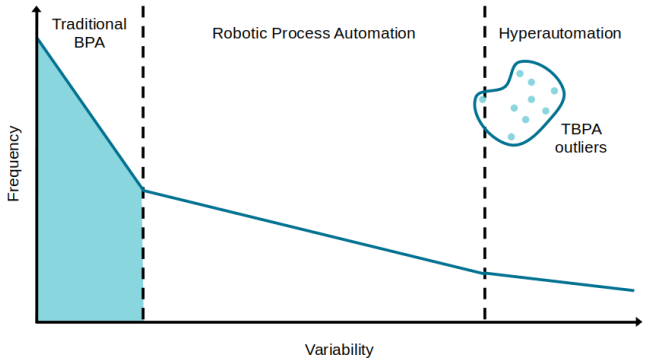

FIGURE 1. BPA approaches applicability assessment. Adapted from Aalst et al. [4].

Although TBPA is not suitable for automating processes with high variability, several TBPA outliers commonly emerge in practice for two main reasons: (1) processes are dynamic, and (2) HA is not universally applicable. Business processes are inherently dynamic, which implies the need for TBPA software to support and accommodate changes [48]. Hyperautomation is still a vanguard approach [4], [16], [24], [34], [62] and holds the potential to effectively address the challenges posed by business process changes. However, it may not be suitable for every business due to complexity, implementation costs, process suitability, resource constraints, and cultural readiness [39], [58].

To implement TBPA software, the requirements must be specified to ensure that the software meets the needs of the organization and achieves its desired goals [48]. Researchers have provided evidence that software becomes more flexible and capable of adapting to changes when requirements and architecture are well aligned [28], [41], [49], [52], [56], [71], [79], [80]. Nonetheless, Menezes [48] analyzed 46 elicitation methods and identified limitations to overcome in eliciting requirements for TBPA software, such as (1) process variability, (2) deprecated documents, (3) lack of knowledge about the process, (4) unfamiliarity with the vocabulary used by practitioners within the organization, and (5) lack of engagement from practitioners involved. Researchers [17], [26], [27], [30], [44], [48], [66] suggested employing event logs and process mining to enhance requirements elicitation. Menezes [48] also considered both as a way to deal with the mentioned challenges.

This work conducted a case study in a technology institute to investigate the following questions:

- **RQ$_1$**: How to make TBPA software more adaptable to process changes?
- **RQ$_2$**: How to reduce the dependence on practitioners to elicit requirements for TBPA software?

The research questions arose from the need to improve TBPA software development in the institute, which faces issues related to process variability and practitioner unavailability for requirement elicitation. To address such issues, this research proposes an approach considering learned lessons from previous TBPA software development projects and studies conducted by Menezes et al. [12], [47], [48], [70]. The approach employs log analysis and process mining techniques to reduce the dependence on practitioners during requirements elicitation, as well as ensures high traceability between process requirements and software architecture to make TBPA software more adaptable to process changes.

The remaining parts of the paper are organized as follows: Sections II and III address the background and related works respectively. Section IV describes the proposed approach. Section V presents the case study applied to evaluate the approach and the obtained results. Section VI discusses the results and relates them to other studies. Finally, Section VII gives the conclusion.

II. BACKGROUND

This section introduces the main concepts related to TBPA software, including its definition, challenges such as process variability and dependence on practitioners, and the role of process mining and loggers in improving and understanding the business process. Finally, the section provides an overview about Web Scraping techniques.

A. TRADITIONAL BUSINESS PROCESS AUTOMATION SOFTWARE

According to Menezes [48], Traditional Business Process Automation (TBPA) software consists of software that automates business processes within a particular digital ecosystem by using programming languages (Python), tools (Selenium), and techniques (Web Scraping). It suits to

automate repetitive standardized tasks [4], [32], [48], [74], which leads to significant time and cost savings for organizations, as well as greater accuracy and consistency in performing these tasks [7], [9], [29], [32], [33], [40], [42], [43], [48], [54], [57], [72], [73], [74], [76], [77]. Despite these benefits, Menezes [48] emphasized the challenges in TBPA software development. This research addresses only the ones related to process variability and dependence on practitioners for requirement elicitation.

B. PROCESS VARIABILITY

Developing TBPA software requires significant effort and resources to deal with process variability. Several factors impact the business process, including changes in the workflow, digital ecosystem, privacy policies, and legal regulations [48]. For example, TBPA software must be modified whenever: (1) a step is introduced, rearranged, altered, or eliminated in the process workflow; (2) software is incorporated, modified, replaced, or removed from the digital ecosystem; (3) policies alter the usage of certain data or internal systems; or (4) there are updates to comply with new laws mandating businesses to collect and store customer data.

If process changes, the TBPA software will not work as expected, and it must be updated to support them. To deal with these changes, studies [28], [41], [49], [52], [56], [71] proposed a high alignment between requirements and architecture. This ensures that when a requirement undergoes any changes, developers can promptly trace the related component and implement the required modifications in software efficiently.

Several researchers have introduced different approaches to identify and manage process changes. These approaches were based on integration of business process modeling with requirements engineering [18], [45], [55], [61], [68], automatic techniques to enhance software development [6], [10], [66], [67], and integration of requirements engineering and software architecture [71].

Software architecture defines the high-level structure of a software system, focusing on major components, their relationships, and overarching goals such as scalability, security, and performance [13]. In contrast, software design addresses the finer details, outlining the internal operations within each component. It involves decisions about algorithms, data structures, and class interactions, aiming to implement the architectural vision with precise and executable solutions [22]. While architecture provides a strategic blueprint for the overall structure of the system, design translates this blueprint into the functional elements that comprise the code [60]. From now on, the term *architecture* encompasses both architecture and design.

This work proposes an approach to align business process requirements closely with the software architecture to make TBPA software more adaptable to process changes. Embedding traceability from the process workflow and involved components within the digital ecosystem into the architecture, the approach enables more efficient identification and implementation of necessary updates.

C. DEPENDENCE ON PRACTITIONERS

Practitioners, the stakeholders responsible for executing the business process, play a crucial role in providing insights, clarifications, and context during the requirements elicitation. Practitioner unavailability encumbers TBPA software development [48] and introduces risks into TBPA software development such as: (1) inadequate understanding of business processes; (2) misalignment with organization needs; (3) increased rework and delays; (4) limited validation opportunities; (5) difficulty in handling exceptions and edge cases; and (6) reduced stakeholder buy-in.

Researchers [17], [26], [27], [30], [44], [48], [66] suggested employing logs and process mining to elicit more precise and reliable requirements since the technique automatically models and documents the business process and its variants, reduces the risks arising from deprecated documentation, and reduces dependence on the engagement of stakeholders who master the process.

This research also proposes the use of log analysis and process mining with three primary goals: (1) obtaining a workflow that closely mirrors the actual business process; (2) identifying the technologies of the digital ecosystem utilized during process execution; and (3) capturing the URLs and payloads of HTTP requests exchanged among these technologies. These findings have the potential to enhance requirements elicitation for TBPA software and reduce dependence on practitioners.

D. PROCESS MINING

Process mining is a data-driven technique employed to discover, check, and improve real-world processes from event logs available in the digital ecosystem within an organization. Process mining leverages these logs, typically recorded in information systems during the execution of business process, to extract the sequence of actions, timestamps, and relevant attributes. Based on these data, it performs data mining algorithms to find patterns and transitions that are used to provide process discovery, conformance checking, and enhancement [14], [53].

The technique also helps in understanding how processes unfold in practice, identifies bottlenecks, as well as supports conformance checking and continuous monitoring. Its iterative nature enables organizations to adapt and refine their understanding of processes over time, making it a valuable tool for enhancing operational efficiency and ensuring compliance within organizational workflows [1], [3], [5]. Despite this, the technique relies solely on the existence and availability of logs, potentially neglecting tasks that lack corresponding records [64].

Saito [66] utilized process mining over event logs from existing systems to automatically model the business process and elicit more accurate and reliable requirements.

The approach was able to identify the industry entities and how they performed tasks during process execution. Hernandez et al. [30] proposed a framework to use records from relational databases to discover the patient journey in a hospital.

Although several studies have suggested mining the business process from the logs of the organizational systems [4], [17], [26], [27], [30], [44], [66], this work proposes generating the logs to mine the process. This is particularly interesting in organizations where the logs are unavailable to mine or to overcome invisible tasks [64].

E. LOGGER

Loggers, or logging systems, are used in a wide range of computer systems, such as operating systems, web servers, application servers, database management systems, security systems, etc. They are prevalent in nearly all computing environments to aid in troubleshooting, monitoring, security, and performance analysis [11]. The specific log formats, locations, and management tools can vary widely depending on the system and technology stack in use.

In this study, a logging tool was designed and implemented to capture event data from practitioners' devices, serving as an alternative solution in scenarios where system-generated event logs are unavailable.

F. WEB SCRAPING

Browsers and websites are built over two main technologies: Hypertext Transfer Protocol (HTTP) and HyperText Markup Language (HTML). Web scraping refers to a set of techniques to interact with HTTP and parse HTML pages, mimicking human navigation of websites [36], [50]. Each technique is suitable for different types of web content and objectives.

The most common involves downloading web pages and parsing the HTML to extract data. Other techniques include replicating HTTP requests, using Application Programming Interfaces (APIs), in Representational State Transfer (REST) or Simple Object Access Protocol (SOAP), provided by websites for data access, or employing browser automation tools to mimic user interactions for dynamic data extraction.

This study proposes employing web scraping techniques and microservice architecture to implement interfaces with the functionalities of each business system when APIs are unavailable.

III. RELATED WORKS

This section examines the related works that address the challenges of capturing, analyzing, and aligning business process requirements with software functionalities. Highlighted studies explored different approaches to enhance the elicitation, documentation, and maintenance of system requirements, ensuring they are comprehensive, accurate, and adaptable to evolving business environments. The related works were grouped into three categories considering common characteristics such as the methods, benefits, and limitations.

A. BUSINESS PROCESS MODELING AND REQUIREMENTS ENGINEERING INTEGRATION

Studies [18], [45], [55], [61], [68] explored the integration of business process modeling with requirements engineering to enhance the development of BPA systems. According to the authors, traditional methods might not capture the complexities of business processes and their contexts comprehensively. For this reason, these researchers proposed systematic approaches to bridge this gap by employing process models into requirements elicitation.

Cardoso et al. [18] proposed to use business process models as a foundation for deriving system requirements. By formally documenting business processes, the approach provides a clear and structured representation of the activities and their contexts within an organization. This modeling allows analysts to identify various levels of automation for different activities, leading to multiple sets of requirements based on these variations. The case study, conducted in the Human Resources (HR) department of a large energy organization, demonstrated the effectiveness of the approach. When compared to traditional techniques, the approach resulted in more complete and correct requirements. The proposed approach also identified and corrected inaccuracies. Overall, the approach provided a more holistic view, leading to requirements that better reflected the organizational context and workflows.

Przybylek [61] proposed the Business-Oriented Requirements Elicitation (BORE). The approach is organized into three main stages: (1) business process modeling, (2) business process improvement, and (3) functional requirements elicitation. The first one involves documenting current business processes to establish a baseline. The second one focuses on improving these processes through a cost-benefit analysis and prioritization of automation opportunities. The final stage involves deriving system requirements from the improved business processes, ensuring that each requirement is traceable to a specific business need. BORE was employed in a case study conducted at a private university in Poland. BORE was able to model processes, identify areas for improvement, and derive the necessary system requirements. The case study involved active participation from stakeholders through workshops and interviews, which helped validate the process models and ensure that the derived requirements met real business needs. The results showed that BORE facilitated better understanding and communication among stakeholders, leading to a more effective alignment of the business processes and the automation.

Enterprise Resource Planning (ERP) systems promise significant benefits to organizations. However, ERP projects frequently struggle due to inappropriate specifications and misunderstanding of requirements. Panayiotou et al. [55] integrated business process modeling methods to formulate functional specifications and manage requirements effectively for ERP systems. The research conducted a case study involving a Greek manufacturing company and demonstrated that the approach facilitated the transition to the new ERP

system. It also showed improved alignment between the ERP functionalities and business requirements, leading to a successful implementation.

Silva et al. [68] proposed Box Analogy Technique (BoAT) to address the challenges of capturing business process models during the requirements elicitation in agile software development. BoAT is an agile-based approach that integrates cognitive-visual analogies to enhance the efficiency and effectiveness of capturing business processes during requirements elicitation. The technique was applied in three different organizations in Brazil, demonstrating its practical utility and offering insights into its benefits compared to traditional interview methods. The three case studies demonstrated its effectiveness in improving the quality and efficiency of business process capture. Compared to traditional interview techniques, BoAT produced more accurate and detailed business process models, with participants reporting higher levels of engagement and focus during the interviews. The case studies showed that BoAT could reduce the time required to capture and validate business processes while producing artifacts that were more aligned with the actual needs of the organization. Feedback from participants in all three case studies was overwhelmingly positive, with many expressing a preference for using BoAT in future projects.

Mateus et al. [45] proposed a systematic approach to derive user stories and scenarios directly from procedd models. This approach aimed to ensure traceability and consistency between business processes and system requirements, thereby facilitating better communication between stakeholders. The approach utilized transformation patterns that translate process elements into user stories and scenarios. Process models were analyzed, and patterns were applied to extract relevant information. User stories provided a high-level view of system functionality from the perspective of users, while scenarios detailed the different paths a process can take, including exceptions.

Despite their advantages, the proposed approaches have limitations related to high business process variability and practitioner unavailability. High variability in business processes can complicate the modeling and subsequent requirements derivation, as the business process modelling must account for numerous exceptions and variations. It can lead to a continuous need for updates and adjustments in both the business process models and the derived system requirements. In addition, the success of these approaches rely on the availability and cooperation of stakeholders who possess in-depth knowledge of the business processes. If these stakeholders are unavailable or unwilling to participate, the quality and completeness of the business process models and derived requirements can suffer.

This dynamic nature of business processes requires a flexible and iterative approach, which the proposed ones partially addressed. The dependence on stakeholders poses a significant risk. Without active and consistent participation, the models and requirements may not fully capture the business needs, leading to potential misalignments and project failures. Therefore, more robust mechanisms for handling rapid changes and practitioner unavailability are needed.

B. AUTOMATED TECHNIQUES TO ENHANCE SOFTWARE DEVELOPMENT

Traditionally, requirements elicitation is time-consuming and prone to inconsistencies. Researchers [6], [10], [66], [67] used automated techniques to enhance software development and reduce the impact of these issues.

Aysolmaz et al. [10] proposed a semi-automated approach to deriving natural language requirements documents directly from business process models. It integrates process model analysis with natural language generation to produce consistent and maintainable requirements documents. The method ensures that the generated requirements are both consistent with the process models and easy to maintain. The approach was evaluated through a multiple case study involving three organizations. The findings demonstrated that the generated requirements were well-readable, complete, and maintainable.

Saito [66] addressed the challenge in discovering business processes that involve multiple stakeholders, especially when these processes are not well-documented. Traditional methods for identifying stakeholders require extensive manual effort and are prone to errors and omissions. To solve this issue, Saito [66] utilized process mining techniques to analyze event logs and user information. The results of applying it in an industrial case study demonstrated that the approach successfully provided a comprehensive view of the business process enabled the identification of all relevant stakeholders and provided a clear mapping of activities to organizational entities. This level of detail supports more accurate and complete requirements elicitation.

The study conducted by Abbas et al. [6] investigated the relationship between requirements similarity and software similarity in industrial applications to enhance software reuse. It explored various Natural Language Processing (NLP) techniques to determine their effectiveness in correlating these similarities, using real-world requirements from two railway domain projects. The research addressed the lack of empirical evidence supporting the assumption that similar requirements can be proxies for retrieving similar software. The study compared several state-of-the-art NLP approaches. Its findings suggest that requirements similarity can serve as a proxy for software similarity, although further improvement in NLP techniques for requirements-based code retrieval is needed.

Unlike Aysolmaz et al. [10], Sholiq et al. [67] addressed the conversion of natural language requirements into process diagrams, which can be used to better understanding business process. The approach was tested on 10 textual requirements of an enterprise application, demonstrating its ability to handle various sentence structures, showing a potential improvement for efficiency and accuracy of software project planning and execution.

Despite the positive results, some common limitations were identified for the cited approaches. First, they relied on existence and quality of data. In cases where processes vary significantly or stakeholders are not available to provide insights, they can generate inaccurate results. The approaches also may not fully capture the dynamic nature of business processes that change frequently or involve ad-hoc activities. In addition, AI techniques implementation are computationally intensive and requires high initial investment in resources such as expensive and skilled labor, time, and digital infrastructure.

C. REQUIREMENTS ENGINEERING AND SOFTWARE ARCHITECTURE INTEGRATION

Spijkman et al. [71] proposed Requirements Engineering for Software Architecture (RE4SA) to address the failure of software projects due to poor requirements management and the disconnect between requirements engineering and software architecture. Enterprise software, such as ERP systems, often requires extensive customization, leading to additional maintenance challenges and complexities in managing these customizations. The lack of adequate documentation and alignment between requirements and architecture exacerbates these issues, resulting in project delays, increased costs, and suboptimal software solutions that do not fully meet user needs.

RE4SA offers a structured approach to link requirements and architecture through specific artifacts and trace links. The approach integrates user stories with software modules and features, creating a functional architecture that evolves concurrently with the requirements.

The research [71] conducted four case studies involving a software product. RE4SA improved the goal orientation of requirements, provided a clear overview of customer-specific environments, and helped identify the location of features and modules in new software versions. The model also contributed to concise yet detailed documentation, which enhanced communication with customers and stakeholders.

Despite its advantages, RE4SA has some gaps. One is its reliance on manual updates to the architecture diagrams, which can become outdated if not maintained regularly. Additionally, its complexity can be overwhelming, particularly for large-scale enterprise software with numerous features and modules. There is also a need for better tool support to automate traceability and navigation within the architecture diagrams.

IV. PROPOSED APPROACH

This paper proposes *Requirements with Logs* (RWL). The approach initially emerged as lessons learned from previous TBPA software development projects and studies conducted by Menezes et al. [12], [47], [48], [70], and it was refined over time. RWL uses log analysis and process mining techniques to specify requirements for TBPA software with high alignment between business process requirements and software architecture. With this, the approach aims to improve TBPA software development by making it more adaptable to changes in process, as well as reducing the dependence on practitioners during requirements elicitation.

RWL brings two contributions to requirements engineering: (1) business process discovery and (2) software architecture generation. RWL utilizes a logger to register events, a process miner to discover the business process, and an HTTP request analyzer to examine requests from the browser. These tools allow RWL to gain insights into how the business process operates and interacts with the digital ecosystem, instead of eliciting them directly from practitioners. These insights are used to refine requirements and trace them into the software architecture.

The following subsections introduce an overview of the RWL application during requirements engineering, the employed tools, the business process discovery, as well as the software architecture generation and its artifacts. Moreover, Figs. 2, 5, and 6 also overview the main differences between the versions developed during the case study described in Section V, in which blue items represent $TBPAS_1$, and blue and orange items $TBPAS_2$.

A. APPROACH OVERVIEW

Fig. 2 shows how the approach can be applied in requirements engineering. Any Software Development Cycle (SDC) has activities related to elicit, express, document, and validate requirements for software. RWL adds activities to discover the process and generate the architecture.

1) ELICIT REQUIREMENTS

RWL starts by eliciting requirements. This activity provides a shared understanding among the stakeholders (developers, testers, designers, practitioners, leaders, managers, etc.) about the domain, business process, and needs, producing the requirements for the TBPA software.

2) DISCOVER BUSINESS PROCESS

After that, RWL plays a role in discovering the real business process. It employs a strategy based on event logs and process mining to help in understanding how the business process operates in reality [1], [3], [8]. This activity outputs the transitions, tasks, systems, Uniform Resource Locators (URLs), payloads, and requirements.

3) WRITE REQUIREMENTS

The next activity involves expressing and documenting requirements to facilitate the management of activities, communication, and collaboration among all stakeholders. This activity produces the documented requirements.

4) VALIDATE REQUIREMENTS

Following the writing requirements, the documented requirements are shared with the stakeholders for validation. If the stakeholders approve the requirements, the artifacts can be generated. However, if there are disagreements or changes

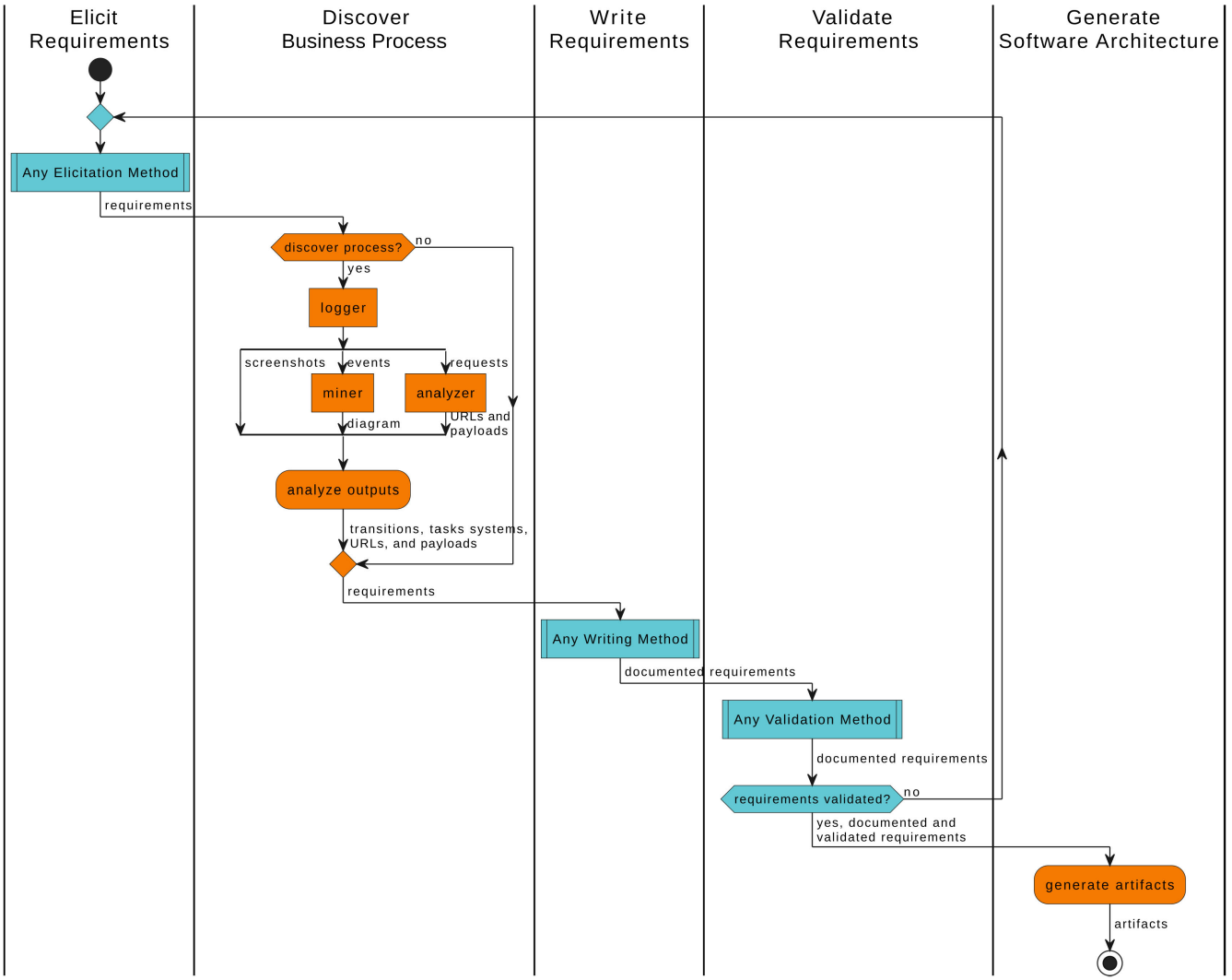

FIGURE 2. UML activity diagram showing the application of RWL in requirements engineering.

requested by any stakeholder, it may be necessary to restart the specification phase to address their concerns and ensure alignment among them. This activity produces the validated requirements.

5) GENERATE SOFTWARE ARCHITECTURE

Finally, the validated requirements are analyzed to produce the RWL artifacts following the guidelines described in Section IV-F.

B. LOGGER

RWL logger is an application that runs on the computers of practitioners. It monitors mouse clicks and keyboard key presses. When such events occur, it takes a screenshot and identifies the active application name. It also works as a sniffer and records the HTTP requests. The logger encrypts and stores data in a database with the attributes in Table 1.

The logger was developed in Python and uses the `os`, `win32api`, and `win32gui` modules to interact with the operating system to capture the user identifier and application name. The `pyautogui` module was used to capture the screenshot and the events from the mouse and keyboard. In addition, it employs the `pyshark` module to capture network traffic from a given interface.

TABLE 1. Logger data example.

Activity	UserID	DeviceID
Browser: New Tab	145	226
ScreenshotID	**PCAPID**	**Timestamp**
264877	289784	20230405T150132

C. PROCESS MINER

There are more than 30 process miners available on the market [2]. The most cited miners are provided by Celonis (available at https://www.celonis.com) and UiPath (available at https://www.uipath.com/platform/discover/

process-mining). Due to the costs of obtaining the tool and the limitations of the trial versions, it was decided to implement the miner following the insights provided by several studies [2], [14], [30].

The miner was developed in Python using the module `pm4py` (available at https://pm4py.fit.fraunhofer.de). This module provides algorithms to convert tabulated data into the eXtensible Event Stream (XES), data mining algorithms to discover the business process, and others to generate the process diagram. Despite the fact that `pm4py` offers several algorithms to mine processes, this miner uses Directly-Follows Graph (DFG) due to its status as the standard algorithm in commercial miners [2].

First, the miner retrieves the event logs, in tabulated format, from the database. These logs comprise the execution of activities by each practitioner on each device within the business process. The miner loads attributes such as Activity, UserID, DeviceID, and Timestamp from each log and converts them into XES format. After loading XES data, the miner runs DFG to perform the process discovery. DFG represents activities as nodes and the transitions among them as directed edges. To determine the transitions between two nodes, the miner considers the time inter-lapsed between the two events. For this reason, it employs the performance decorator instead of the frequency one. Finally, the miner outputs the mined process as a diagram as shown in Fig. 3. The diagram is a DOT file, which can be manipulated and visualized by Graphviz (available at https://graphviz.org), a widely used open-source software for graph visualization.

Fig. 3 illustrates a process for data extraction, preparation, and dissemination, incorporating repetition and conditional decision-making. The green circle represents the beginning of the process, while the orange one its conclusion. It initiates with practitioner opening a spreadsheet file labeled `agency-data`. A decision gateway subsequently divides the workflow into two potential paths. In the first path, the practitioner opens a new browser tab, navigates to the website `IT Dashboard`, and accesses the page `Agencies`. This pathway appears to incorporate a loop, indicating repeated visits to the page, likely to iteratively gather or validate information for accuracy and completeness. The alternative path involves data formatting within the spreadsheet. Here, the practitioner applies formatting to data cells, possibly to standardize the data presentation and ensure clarity for subsequent users. Following data preparation, the workflow progresses to the email client, in which the practitioner composes a new message, attaches the formatted spreadsheet file, and initiates the sending task. The sequence culminates with the successful transmission of the email, signifying the completion of the workflow.

D. HTTP REQUEST ANALYZER

RWL logger monitors the network interfaces of computers to capture HTTP messages transmitted during business process execution. The logger stores these messages in PCAP format, a format widely used by most network traffic analyzers.

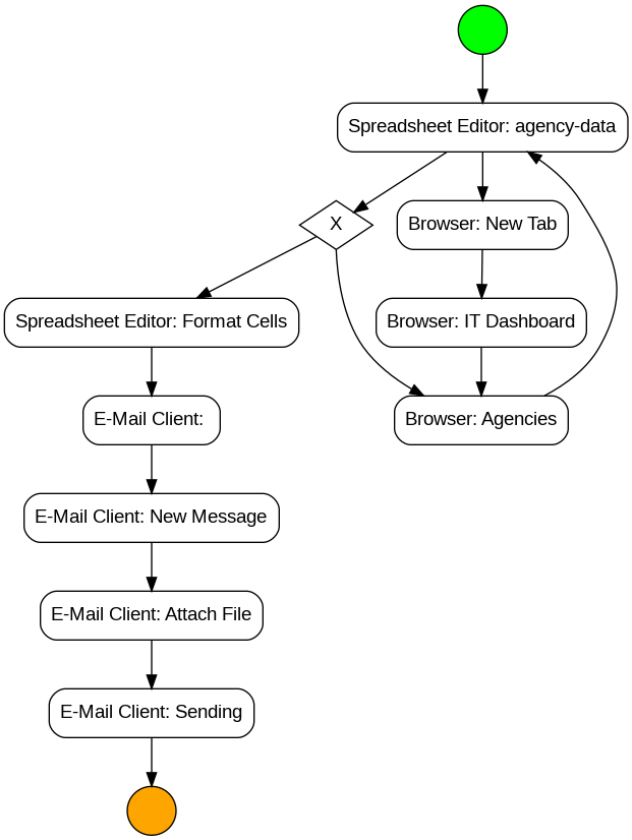

FIGURE 3. Example of process diagram generated by the miner.

Wireshark (available at https://www.wireshark.org/) was used to analyze the recorded messages from HTTP requests. Wireshark is a free and open-source application that captures and analyzes network traffic, supporting a wide variety of protocols. It provides features to quickly and easily dissect network packets to identify URLs and payloads used in HTTP messages.

Overall, Wireshark is a powerful and flexible packet analyzer that allows elicitors to analyze data transmitted by a system or web API, which facilitates the identification of digital ecosystem components and deepens knowledge about the process, its steps, and the tasks carried out.

E. BUSINESS PROCESS DISCOVERY

The business process discovery begins with the practitioners, the stakeholders responsible for executing the business process. They run the logger on their computers to capture computer events, screenshots, and browser requests. The logger stores them in a database for later analysis.

After that, the process miner reads the logs from the database and runs data mining algorithms over them to create a comprehensive diagram of the real business process. Additionally, the HTTP request analyzer assists in the analysis of these requests to identify the involved systems, transmitted data, URLs, and payloads during process execution.

By analyzing these outputs, the elicitors obtain a more precise understanding of the business process and can generate the software architecture. For example, elicitors use the diagram and the screenshots to outline the transitions, tasks, and systems involved in the process. The analyzer gives the URLs and payloads used by the systems during the process execution, which are utilized by developers to implement microservices for each system of the process.

F. SOFTWARE ARCHITECTURE GENERATION

Fig. 4 shows the relationships between the outputs produced by RWL tools and the software artifacts. Such outputs are instrumental in maintaining traceability between the business process requirements and the software architecture.

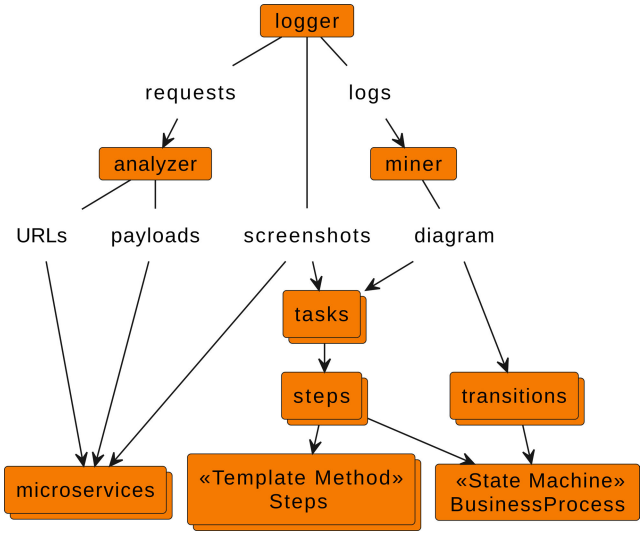

FIGURE 4. Relationships between outputs and software artifacts.

The analysis of these relationships leads to an intuitive conception of artifacts. To facilitate the conception process, RWL suggests adopting the following guidelines:

1) Group related tasks into a step to reduce transitions;
2) Implement each step utilizing the template method pattern to create a class `Step` with a method for running its respective tasks;
3) Model steps and transitions using a state machine to create the class `BusinessProcess`;
4) Emulate system functionalities in specific microservices; in general, such emulation is implemented using Web Scraping techniques [20].

Once conceived, these artifacts must be accommodated into RWL architecture, which has a hybrid architecture that combines several aspects of other architectures and design patterns such as orchestration [46], client-server, microservices, domain-driven design, component-based development [63], factory, state machine, and template method [21].

Fig. 5 illustrates the RWL architecture to implement TPBA software for a generic business process. Additionally, Fig. 6 details the main artifacts of the RWL backend for TBPA software.

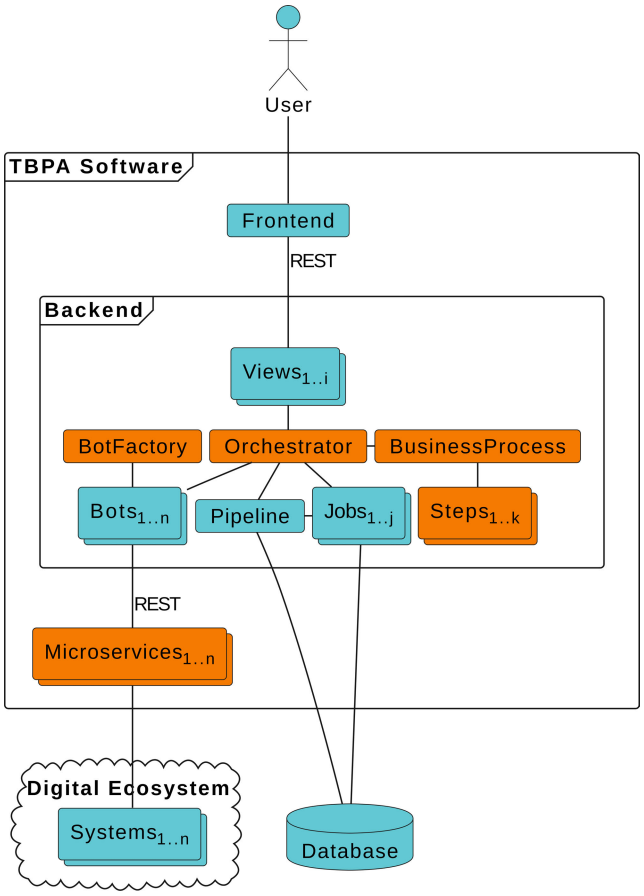

FIGURE 5. RWL architecture.

RWL backend defines some artifacts needed to develop TBPA software. Each artifact is developed as a reusable component that represents or considers the core business concepts of the TBPA software. These artifacts are described below:

- `Orchestrator` centralizes and orchestrates the business process execution;
- `BusinessProcess` refers to a state machine that models the business process transitions;
- `Step` implements a template method to execute a set of related tasks that are associated with a specific process step;
- `Pipeline` stores general process data or information that is shared across all `jobs`;
- `Job` stores specific data about a particular process execution;
- `BotFactory` provides an interface for creating `bots`;
- `Bot` integrates the `Orchestrator` into a `microservice`;
- `Microservice` implements the necessary functionalities of a particular system.
- `View` associates a specific URL route to a method from the `Orchestrator`.

RWL utilizes the state machine `BusinessProcess` to represent the business workflow in a structured way,

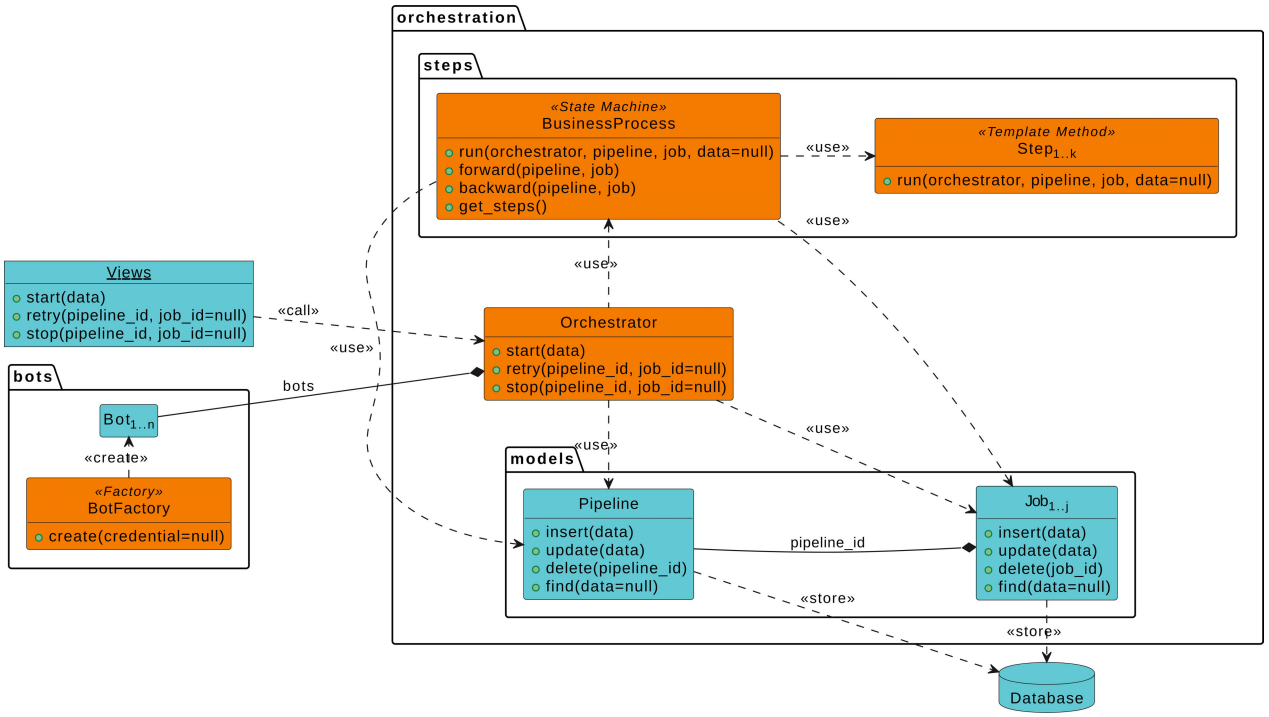

FIGURE 6. RWL backend design.

making it easier to understand and manage, ensuring that the process is always followed correctly [21]. Despite running the process, the class knows the sequence of steps to be followed and walks through it by going forward or backward.

`Steps` are used by `BusinessProcess` to execute a set of related tasks for the process. Each `Step` implements a template method `run` that allows developers to create different variations of steps and easy reuse of code [21]. A process with `k` steps will potentially generate `k Steps`.

`Microservices` implement the interfaces between the backend and the systems. Such implementation allows each interface to be developed, deployed, reused, and scaled independently of others [63]. This way, a process with `n` systems will possibly have `n Microservices`, as well as a process with `n Microservices` will have, at least, `n Bots` in the backend to interact.

The `Orchestrator` provides centralized control over the entire business process. It allows TBPA software to have a high level of control by starting, restarting, or stopping the process. In addition, orchestration improves the efficiency and scalability of TBPA software [46]. The `Orchestrator` performs the business process through the `BusinessProcess` and interacts with the digital ecosystem by using the interfaces provided by `Bots` and `Microservices`. The `Orchestrator` also utilizes the factory `BotFactory` to abstract the creation of `Bots` [21].

Finally, `Views` are responsible for handling REST requests from the `Frontend`, calling the methods from the `Orchestrator`, and returning the respective responses [63]. `Frontend` is the bridge between `Backend` and `User`.

RWL architecture provides centralized control of the business process, reuse of code and functionalities, and a loosely coupled architecture, which improves the efficiency, maintainability, and scalability of TBPA software.

G. PROOF OF CONCEPT

To provide an initial example of the proposed approach, this work developed a Proof of Concept (POC), which is available on GitHub at https://github.com/joao-nascimento-it/rwl-poc. This implementation serves as a practical demonstration of how RWL can be utilized to standardize and simplify TBPA development.

V. CASE STUDY

In this research, a case study was conducted to assess RWL in TBPA software development. The protocol was based on the guidelines described in works [38], [65], which consisted of the following steps: (1) define objectives and hypotheses; (2) select the case; (3) define the metrics; (4) develop a sister project; (5) collect data; and (6) analyze results.

A. OBJECTIVES AND HYPOTHESES

Considering the research questions presented in Section I, the objective of this case study was to evaluate RWL in improving TBPA software development, specifically in terms of making software more adaptable to process changes and reducing the dependence on practitioners to elicit requirements.

RWL accurates requirement elicitation by using logs and process mining to discover the business process and trace

it to software components, making requirements closer to architecture.

According to researchers [18], [28], [35], [37], [41], [49], [52], [56], [59], [71], [78], a high alignment between requirements and architecture makes software more adaptable to changes. Moreover, Menezes [48] argued that the usage of logs and process mining has the potential to reduce dependence on practitioners. Based on this context, the following hypotheses were formulated to answer the research questions:

- For RQ_1:
  - **H_1**: High traceability between business process requirements and software architecture improves the adaptability of TBPA software to process changes.
- For RQ_2:
  - **H_2**: Logs and process mining aid elicitors to discover the digital ecosystem technologies and the business process, minimizing assistance from practitioners;
  - **H_3**: Logs and process mining give an overview of the whole process and the digital ecosystem that assists elicitors to elicit more precise and reliable requirements, which has the potential to reduce the dependence on practitioners;

B. THE CASE

This case study was conducted in a research institute of a global technology company. The institute collaborates in multiple processes with diverse organizations worldwide to develop Android for mobile devices, utilizing Distributed Software Development (DSD) [12]. The institute has a dedicated team responsible for developing TBPA software to automate these processes.

Fig. 7 overviews the case of this research, which is a TBPA software developed to automate the process of updating device specifications for Android in Latin America. This process involves 82 customers (telecom companies), 8 teams, and 109 people across 23 countries spanning Latin America and Asia. It is executed at least 12 times daily. The process also changes according to customer demands, modifications in the systems of the company, and policies to access them. On average, there are 6 changes per month. The TBPA software performs a series of sequential and parallel tasks, interacting with various functionalities across 6 distinct systems. Among these systems, only 2 provided APIs, while interfaces with the remaining were implemented through web scraping.

Unfortunately, event logs are inaccessible, as the entire digital ecosystem falls outside the domain of the institute. Complicating matters, practitioners were frequently unavailable due to differing time zones and time constraints for meetings. They also had different levels of expertise in the process, coupled with deprecated process documentation.

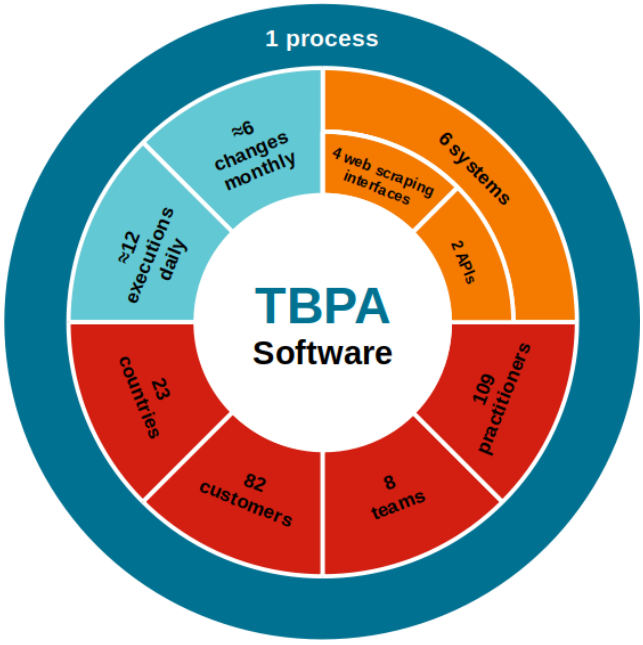

FIGURE 7. TBPA software for automating device specification updates for Android in Latin America.

The high process variability and practitioner unavailability led to a decline in quality and failure to meet deadlines for TBPA software releases.

C. DATA AND METRICS

In this research, the following data were collected to evaluate RWL and validate the hypotheses:

- Data related to H_1, adaptability to process changes:
  - *Process Adaptation Time* (PAT): time, in hours, taken to adapt the project when process changes [69];
- Data related to H_2 and H_3, dependence on practitioners:
  - *Practitioner Meeting Time* (PMT): time, in hours, spent with practitioners to clarify the development of an issue [19];
  - *Total Practitioner Meetings* (TPM): amount of issues due to clarify the development of an issue with practitioners [19].

For triangulation purposes, this study also collected data already utilized in the institute to assess TBPA software development, as suggested by Runeson and Höst [65]:

- Data related to development efficiency:
  - *Bug Resolution Time* (BRT): time, in hours, taken to resolve a bug [69];
  - *Issue Resolution Time* (IRT): time, in hours, taken to complete an issue (bug, improvement, process change, practitioner meeting, task, or other) [69];
  - *Total Bugs* (TB): amount of bugs reported within the project [19], [51];
  - *Total Issues* (TI): amount of all issues within the project (bugs, improvements, process changes, practitioner meetings, tasks, and others) [19].

- Data related to BPA performance:
  - *Completion Pipeline Time* (CPT): time, in minutes, taken by the TBPA software to complete an pipeline correctly [19];
  - *Total Completed Pipelines* (TCP): amount of pipelines executed and finished correctly by the TBPA software [19];
- Data related to code size:
  - *Number of Files* (NoF): amount of files found within the project [31];
  - *Lines of Code* (LoC): amount of code lines found within the project [15], [31], [75];

D. SISTER PROJECT

This work utilized Replicated Product Design (RPD), a method that consists of replicate the existing product with a new method or tool, and measure the metrics for both versions [38]. This way, a new version of the same TBPA software were developed employing RWL. $TBPAS_1$ denotes the previous version without RWL, while $TBPAS_2$ refers to the new one with the approach.

Figure 8 compares the distribution of roles for each TBPAS development team. The $TBPAS_1$ development team consisted of 21 members: 2 specialist, 4 senior developers, 3 mid-level developers, 5 junior developers, 3 interns, 1 product owner, 2 junior testers, and 1 designer. In contrast, the $TBPAS_2$ development team was smaller, comprising 14 members: 1 specialist, 1 senior developer, 2 mid-level developers, 3 juniors, 4 interns, 1 product owner, 1 junior tester, and 1 designer. It is important to highlight that two developers, a senior and a junior, contributed to both versions of TBPA.

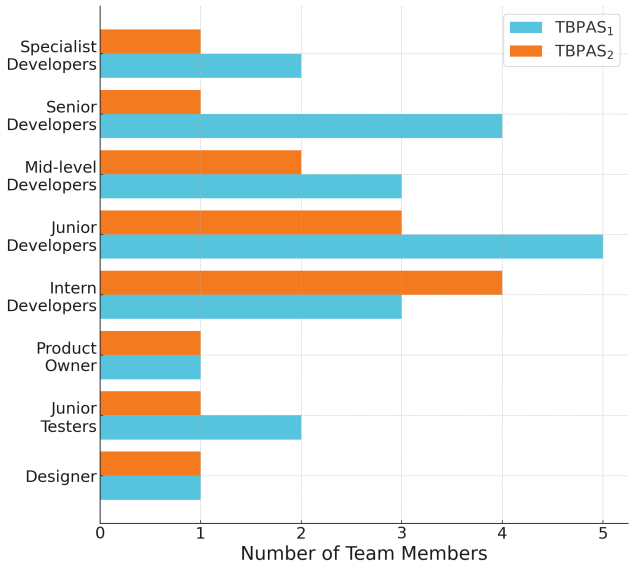

FIGURE 8. Comparison of $TBPAS_1$ and $TBPAS_2$ development teams.

$TBPAS_1$ was developed from May, 2020, to June, 2022. It took 11 months to be deployed in the production environment and accumulated 1153 issues and the 183 process changes during the 27 months of development. On the other hand, $TBPAS_2$ started in February, 2022. It underwent a 5-month development period before being deployed in the production environment. Over the course of 18 months, the project accumulated 407 issues and the process has changed 97 times.

In addition, Figures 2, 5, and 6, introduced in Section IV, also overview the main differences between the versions, in which blue items represent $TBPAS_1$, and blue and orange items $TBPAS_2$.

E. DATA COLLECTION

The institute provides a Project Management System (PMS) to manage project issues and releases, for example, ClickUp, Jira, or OpenProject. Each TBPAS is a project in PMS in which stakeholders register issues such as bugs, improvements, meetings, new features, process changes, and tasks. The tool also allows stakeholders to create flexible reports by describing queries and selecting relevant data.

Except for LoC and NoF, all data were collected from PMS. Additionally, the Linux command-line was employed to measure LoC and NoF by running the following commands:

- For LoC:
  - `find . -name "*.py" | xargs wc -l;`
  - `find . -name "*.ts" | xargs wc -l;`
  - `find . -name "*.html" | xargs wc -l;`
  - `find . -name "*.css" | xargs wc -l.`
- For NoF:
  - `find. -type f | wc -l.`

For each TBPAS, two reports were created in PMS: (1) a table with relevant data about the issues such as ID, description, type, assignee, created date, and resolved date; and (2) a created vs. resolved issues report. The data were collected on August 18, 2023. The collection considered the period between May 1, 2020, and August 17, 2023.

F. RESULTS

In this study, data analysis was conducted using quantitative methods [38]. Initially, the data distribution was examined, followed by normality test. Finally, the comparison results were calculated to evaluate the metrics, outlined in Section V-C, for both TBPAS versions.

Table 2 shows the distribution of time for each variable across $TBPAS_1$ and $TBPAS_2$.

The sample sizes for $TBPAS_1$ and $TBPAS_2$ indicate that $TBPAS_1$ had significantly more data points across all variables. Looking at the percentiles, it is clear that $TBPAS_1$ generally has higher values across the table, indicating longer durations for issues, bugs, changes, and meetings compared to $TBPAS_2$. These results suggest that $TBPAS_1$ typically requires more time to resolve issues, fix bugs, implement process changes, and have meetings with practitioners compared to $TBPAS_2$, which may reflect differences in operational efficiency.

TABLE 2. Descriptive results.

	TBPAS	IRT	BRT	PAT	PMT	CPT
Size	1	1153	495	183	112	516
	2	407	156	97	78	202
Percentiles						
25th	1	8.47	14.1	6.61	26.1	29.3
	2	0.967	8.19	4.03	2.05	28.1
50th	1	49.2	30.1	9.09	51	29.3
	2	26.1	14	5	10	28.1
75th	1	52.9	104	13.4	51.8	58.4
	2	32.6	35.9	5.92	13.2	35.9

TABLE 3. Statistical test results for independent samples.

	Shapiro-Wilk		Mann-Whitney U	
Metric	W	p	Statistic	p
IRT	0.723	<.001	131792	<.001
BRT	0.837	<.001	21421	<.001
PAT	0.941	<.001	1330	<.001
PMT	0.877	<.001	758	<.001
CPT	0.456	<.001	48663	0.084

Table 3 presents the results from Shapiro-Wilk and Mann-Whitney U tests. The first assesses the normality of the data. The second is a non-parametric test.

For all variables, the p-values are less than 0.001, and the W statistics indicate a significant deviation from normality. This means that the time distributions for all variables did not follow a normal distribution for either of the TBPA versions. This justifies the use of non-parametric tests like the Mann-Whitney U in this context, which was used to compare the distributions of the time data between both versions. All variables yielded extremely low p-values, indicating that $TBPAS_1$ and $TBPAS_2$ have distinct time distributions in all five variables.

The comparison of each metric between both TBPA versions was calculated by using Equation 1 and Equation 2.

Equation 1 calculated the result for all metrics, except for TCP. These metrics assess outcomes based on data reduction, in which a lower metric value signifies a larger gain.

$$Metric_{Result} = \frac{Metric_{TBPAS_1} - Metric_{TBPAS_2}}{Metric_{TBPAS_1}} \times 100\% \quad (1)$$

Equation 2 measured the result only for TCP, in which an increase in the percentage of completed pipelines signifies enhanced gains.

$$TCP_{Result} = \frac{TCP_{TBPAS_2} - TCP_{TBPAS_1}}{TCP_{TBPAS_1}} \times 100\% \quad (2)$$

Fig. 9 illustrates the comparison of metrics for $TBPAS_1$ and $TBPAS_2$. Each chart within the figure represents a specific performance metric, allowing for a clear visual comparison between the two versions. Moreover, Table 4 compares the metrics of both versions. The metrics are grouped by equation and arranged in descending order, highlighting the best results at the top.

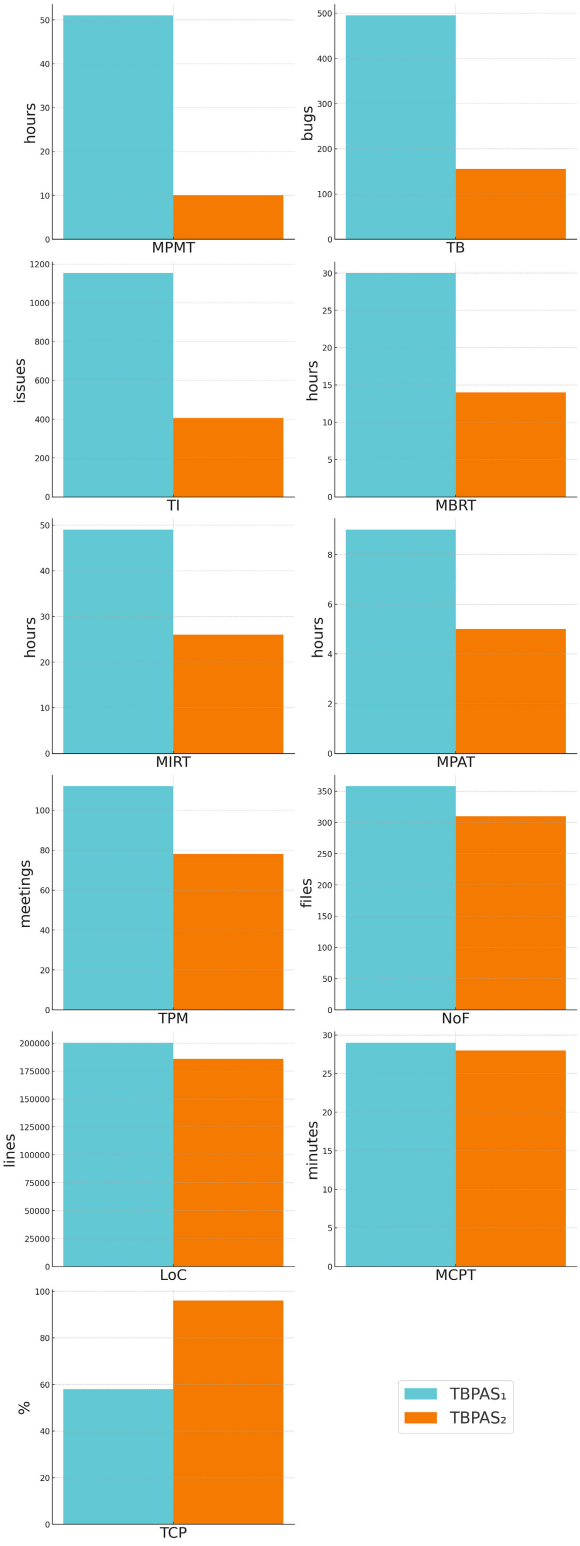

FIGURE 9. TBPA Software Development Performance Metrics.

The statistical analysis of these results shows improvements in the $TBPAS_2$ development across all TBPA software development performance metrics.

TABLE 4. TBPA software development performance metrics.

Metric	Unit	$TBPAS_1$ 01/05/2020 01/07/2022	$TBPAS_2$ 01/02/2022 17/08/2023	Result
Equation 1 (Direction: Down)				
Practitioner Meeting Time (PMT)	hours	51	10	80%
Total Bugs (TB)	bugs	495	156	68%
Total Issues (TI)	issues	1153	407	65%
Bug Resolution Time (BRT)	hours	30	14	53%
Issue Resolution Time (IRT)	hours	49	26	47%
Process Adaptation Time (PAT)	hours	9	5	44%
Total Practitioner Meetings (TPM)	meetings	112	78	30%
Number of Files (NoF)	files	358	310	13%
Lines of Code (LoC)	lines	200379	185838	7%
Completion Pipeline Time (CPT)	minutes	29	28	3%
Equation 2 (Direction: Up)				
Total Completed Pipelines (TCP)	%	58	96	65%

First, adaptability to process changes saw marked improvement. MPAT decreased by 44%, from 9 hours to 5 hours. Such reduction supports hypothesis H_1. The high traceability between business process requirements and software architecture has improved adaptability in implementing modifications, reflecting increased flexibility and responsiveness to business process changes.

There were reductions in the dependence on practitioners. The MPMT decreased by 80%, from 51 hours to 10 hours, while TPM saw a reduction of 30%, from 112 meetings to 78 meetings. These decreases support H_2 and H_3. They showcased a shift towards a more autonomous requirements elicitation, facilitated by the analysis of the logs, mined process, and HTTP requests, thereby reducing the need for meetings and the time spent with practitioners.

Development efficiency experienced improvements as well. MBRT and MIRT decreased by 53% and 47%, respectively, indicating faster resolution times. Additionally, TB and TI decreased by 68% and 65%, respectively. Such reductions, corroborating with MPAT, demonstrate the benefits of making TBPA software more adaptable to process changes. In particular, the decreases in TB and TI also indicate that the requirements have become more reliable and refined, minimizing subjective biases and errors during elicitation. These results in addressing and resolving bugs and issues lead to improved development quality and stability.

In BPA performance, metrics also showed positive trends, particularly in TCP, which increased by 65% from 58% to 96%. Although the MCPT saw only a slight decrease of 3%, from 29 minutes to 28 minutes, the increase in TCP indicates a marked improvement in pipeline efficiency and completion rates, underscoring the effectiveness of the approach.

Lastly, NoF and LoC demonstrated controlled reductions in the code size. The NoF decreased by 13%, from 358 to 310 files, and LoC decreased by 7%, from 200379 to 185838 lines. These reductions suggest efforts towards codebase optimization and refactoring, leading to a more efficient and maintainable code structure. The controlled growth or reduction in these metrics indicates efficient code management practices.

In addition, the study also compared the created versus resolved issues report for both versions. Figures 10 and 11 present the created versus resolved issues report for $TBPAS_1$ and $TBPAS_2$, respectively. Each figure consists of a visual representation of the number of issues created and resolved over a specific period. The gray vertical line represents the deployment in the production environment.

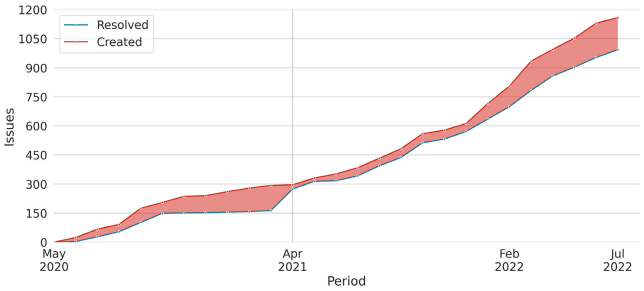

FIGURE 10. Created vs. resolved issues report for $TBPAS_1$.

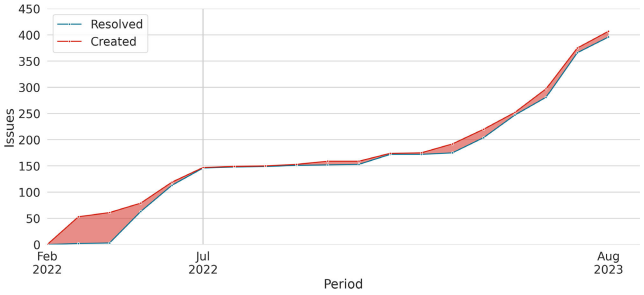

FIGURE 11. Created vs. resolved issues report for $TBPAS_2$.

Fig. 10 reveals a ratio between created and resolved issues, indicating that the $TBPAS_1$ team faced challenges in effectively delivering results. The team encountered difficulties in addressing issues and making progress, resulting

in a significant backlog and potential bottlenecks within the project. On the other hand, Fig. 11 demonstrates that $TBPAS_2$ team effectively delivered results, keeping up with the incoming workload.

In summary, the results for MPAT, MBRT, MIRT, TB, TI, and TCP supported H_1, while MPMT, TPM, TB, and TI confirmed H_2 and H_3. $TBPAS_2$ presented fewer meetings and issues. $TBPAS_2$ also reduced the time spent with bug fixes, process changes, general issues, and meetings with practitioners. These results suggest that RWL software design empowered developers to quickly identify the impacted code and efficiently make necessary modifications to accommodate new requirements. Even with a smaller team, $TBPAS_2$ outperformed $TBPAS_1$. For this reason, $TBPAS_1$ was considered deprecated, and it was replaced by $TBPAS_2$ in July 2022.

G. LESSONS LEARNED

The case study has provided some lessons and limitations that have significantly contributed to understand the dynamics and requirements for successful implementation of the proposed approach.

The first lesson learned was related to capturing HTTPS requests. The captured requests were encrypted, making it impossible to obtain useful URLs and payloads. To overcome this, it was necessary to prepare the computers of practitioners by exporting the `SSLKEYLOGFILE` environment variable into a file and using it to decrypt the requests.

Another lesson learned from using RWL was that an effective analysis of business process still requires human intervention. The logger, the miner, and the analyzer worked as semi-automated tools to provide an overview of the process and the digital ecosystem involved in the institute. Although human intervention has been necessary to align the mined process with the real one, the overview substantially accelerated the understanding of the process and enhanced the requirements elicitation with practitioners, leading to fewer meetings with them and minimizing the subjective biases and errors that can occur with traditional elicitation.

Another significant lesson concerns the complexity of software architecture development. Generating software architecture demanded human effort because it was necessary to acquire a deep understanding of the process to model each step as an entity in the architecture. Furthermore, the case study involved the creation of six microservices to integrate with the systems of the institute. Due to the unavailability of APIs, four microservices were developed using web scraping techniques, which were supported by data derived from the analyzer. This data provided sufficient insights to develop the microservices independently, without requiring practitioner assistance.

Despite this, the development of $TBPAS_2$ became more efficient and precise once developers became accustomed to tracing requirements into the standardized architecture. This mastery streamlined the development by reducing extensive rework, iterative cycles typically required to address misaligned requirements, and time spent on revisions, which improved accuracy in implementation. Additionally, the ability to reuse microservices either within or across different TBPA software greatly reduced the effort required to develop communications with systems throughout the organization. This reuse not only speds up the development cycle but also enhanced consistency and reduces errors by leveraging previously tested and proven components. Consequently, these practices not only expedited the development process but also enhanced the robustness and reliability of the software solutions provided, which was corroborated by MBRT, MIRT, TB, TI, and TCP results.

The loggers installed on the computers of practitioners supplied enough data to compensate for the inaccessibility of the event logs. However, a large volume of logs was generated and transmitted between the loggers and the database. Even in the absence of complaints from practitioners, this extensive data collection represents a risk of data and network traffic overload. Large volumes of data transmitted across the network can strain the digital ecosystem infrastructure, leading to potential congestion. This congestion can significantly impede the efficiency of network communications, affecting the speed and reliability of data exchanges within the digital ecosystem. For practitioners who rely on real-time data access and rapid communication channels to perform their activities effectively, this can result in decreased productivity and operational delays. Thus, managing the volume of data flow and implementing robust network solutions to handle increased traffic is crucial for maintaining network performance and operational efficiency in data-intensive environments.

Finally, privacy and data protection are challenging. The data collection required by RWL raises issues related to privacy and compliance with data protection policies, once the logger has the potential to collect private or confidential data. A practical solution to mitigating the risks associated with data logging involves the careful compilation of a list of applications and systems integral to the process execution. By identifying and cataloging them, organizations can strategically control and limit the scope of data collection. This targeted data logging strategy is crucial in avoiding the inadvertent capture of sensitive information, thereby safeguarding the privacy of practitioners and the confidentiality of business information. Implementing such a measure not only enhances data security but also aligns with legal and ethical standards, ensuring that only relevant data is gathered and stored. Ultimately, this approach facilitates a more secure and responsible handling of data, mitigating the risks associated with privacy breaches and information leakage.

H. THREATS TO VALIDITY

It is important to highlight that two developers have already worked on both TBPA versions. They had significant knowledge about the process as well. Together, these developers worked on 138 of 1153 issues in $TBPAS_1$ (12%) and 126 of

407 issues in $TBPAS_2$ (31%), which may introduce bias into the results.

I. CONFIDENTIALITY AND COMPLIANCE

This case study was conducted under confidentiality and compliance agreements with data protection regulations and safeguarded proprietary processes, tools, and software. For this reason, only parts of this case study were published or redacted to protect sensitive information.

VI. DISCUSSION

This research introduced RWL, a specification approach for TBPA software. By using logs and process mining, the approach traces business process changes and translates them into more precise and reliable requirements, which are accommodated into the software architecture.

The results of the case study highlighted significant improvements in dependence on practitioners, adaptability to process changes, development efficiency, and BPA performance. By using real-time logs and process mining to dynamically elicit requirements, the dependency on stakeholder input was reduced, as objective data captured in logs refined the requirements. This approach enhanced the accuracy and consistency of the requirements by minimizing subjective biases and errors common in traditional elicitation methods.

Continuous monitoring and analysis of logs allowed RWL to detect changes in business processes in real-time and trace them into a standardized architecture that reflects operational workflows, ensuring alignment with current operations. Figs. 10 and 11 clearly show how the real-time feedback loop and the standardized architecture impact software development, improving its efficiency. Consequently, TBPA software performance has increased. These results corroborated studies [26], [48], [66] and studies [28], [41], [49], [52], [56], [71]. The former provided further support for the utilization of business process models and process mining in requirements elicitation. The latter demonstrated a strong correlation between requirements and software architecture to improve software adaptability.

The case study also demonstrated some limitations of the approach. A primary concern was the human intervention required to accurately discover the real business process and accommodate it into the software architecture. Additionally, its dependence on data collection can lead to privacy issues and requires measures to prevent unauthorized access to sensitive information. The approach also faced challenges with network congestion due to the high volume of data transmission, potentially slowing down operations and affecting the overall performance of the digital ecosystem.

Table 5 provides a comparative summary of the methods, benefits, and limitations for RWL and the related works considered in this research.

When analyzing the above-cited approaches, it is important to consider the context in which they were implemented. While RWL was applied to develop TBPA software, the others were employed in different contexts, such as HR software [18], BPA software in a private university in Poland [61], ERP for furniture manufacturer [55], governmental systems [10], enterprise software customization [71], software industry [66], power propulsion control software [6], and others. Each context addresses specific challenges in which similarities and differences have emerged.

While studies [18], [45], [55], [61], [68] focuses on structuring and formalizing requirements, RWL dynamically adjusts to real-time data from process execution, making it more adaptable to changes. RWL minimizes the need for upfront modeling by leveraging logs and process mining, making it potentially less resource-intensive and more agile in rapidly changing environments.

Both automated techniques and RWL aim to enhance the efficiency and accuracy of requirements engineering. However, RWL takes advantage because its direct application to real-time logs and process mining, providing a clear and immediate connection between business processes and software requirements. On the other hand, the approaches proposed by studies [6], [10], [67] may require data preprocessing and model training, making RWL more immediately applicable in certain scenarios.

RWL also focuses on aligning requirements with software architecture but does so by tracing business process changes through logs and process mining. This allows for more immediate and flexible adaptations to changes, whereas RE4SA, proposed by Spijkman et al. [71], may require more upfront planning and resources. RWL is particularly advantageous in environments where business processes are highly dynamic and subject to frequent changes.

RWL minimizes the need for constant stakeholder input by using logs to automatically capture and update requirements, while other approaches may require ongoing stakeholder involvement to manage evolving needs effectively. Another advantage of RWL is its ability to operate without relying on system logs. RWL creates its own logs through the logger in a format optimized for Process Mining algorithms, eliminating the need for data preprocessing to uncover the business process.

Each approach offers unique strengths and addresses specific challenges in software requirements engineering. Business Process Modeling and Requirements Engineering Integration provides a structured method for capturing requirements, while Automated Techniques to Enhance Software Development offer automation and data-driven insights to develop software. Requirements Engineering and Software Architecture Integration focus on aligning technical and business goals.

RWL distinguishes itself by combining the flexibility with the structured alignment of requirements and architecture, facilitated through real-time data from logs and process mining. This makes RWL particularly effective in environments where business processes are highly dynamic and subject to frequent changes. RWL ensures that software remains

TABLE 5. Comparative summary of approaches.

Method	Benefits	Limitations
Requirements with Logs		
• Utilizes log analysis and process mining to refine requirements; • Generates a standardized software architecture for TBPA software.	• Ensures quality and availability of data by generating logs. • Provides a structured, clear, and standardized depiction of business processes; • Facilitates a shared understanding across stakeholders; • Minimizes dependence on stakeholders; • Makes software more adaptable to changes; • Improves the quality and efficiency of software development.	• Requires human intervention to trace requirements into the software architecture; • Still keeps dependence on stakeholders; • Has potential issues related to complex implementation, data privacy, and network traffic overload;
Business Process Modeling and Requirements Engineering Integration Studies [18], [45], [55], [61], [68]		
• Integrates business process models and requirements engineering.	• Provides a structured and consistent depiction of business processes, ensuring traceability and clarity. • Improves communication with stakeholders and increases efficiency in capturing business processes, fostering a shared understanding.	• Can miss crucial requirements due to dependence on stakeholder participation; • Faces challenges with high variability in business processes.
Automated Techniques to Enhance Software Development Studies [6], [10], [66], [67]		
• Employes several techniques such as process mining, AI, ML and NLP.	• Gives more accurate and complete requirements elicitation. • Facilitates better understanding of business processes. • Enhances software reuse.	• Requires stakeholder information and validation. • Has complex implementation. • Is computationally intensive and requires costly resources. • Relies on the existence and quality of data.
Requirements Engineering and Software Architecture Integration Studies [71]		
• Consists of a goal-oriented approach; • Integrates requirements and software architecture; • Uses color-coded diagrams to visualize customisation and new features.	• Improves communication among stakeholders; • Contributes to concise and detailed documentation; • Offers clear overviews of customer-specific environments; • Enhances the management of customization.	• Relies on manual updates, which can lead to outdated diagrams; • Has Complexity can be overwhelming for large-scale software; • Limited tool support for automation of traceability and navigation.

adaptable, aligned with business needs, and resilient to changes, offering a comprehensive solution to the challenges of modern software requirements engineering.

Given that the institute has multiple TBPA tools and high process variability, this research only focused on TBPA software. The results may hold true in other contexts as well,

including RPA, HA, or other software types. In RPA, it can enhance process discovery, reduce practitioner dependency, and improve flexibility by enabling bots to adapt to real-time changes. However, RWL might be too complex for simple RPA tasks, raise data privacy issues, and require substantial integration efforts. For HA, RWL offers comprehensive process automation and real-time adaptability. It can feed valuable insights into AI and ML models, enhancing decision-making and predictive automation. Nonetheless, the high implementation costs, potential data overload, and scalability challenges are notable limitations that need addressing. By extending the application and evaluation of RWL to these different contexts, it can achieve a higher generalization for the approach and uncover broader implications for the research outcomes.

VII. CONCLUSION

Business Process Automation is a strategy for enhancing organizational efficiency, reducing costs, and improving overall performance. The development of BPA software, particularly within Traditional Business Process Automation, faces significant challenges such as process variability and practitioner unavailability.

To address these challenges, this research introduced RWL, a novel approach to develop TBPA software based on aligning business process requirements with software architecture, alongside the utilization of logs and process mining, to enhance adaptability and precision in requirements elicitation.

The results of the case study showcased an 80% reduction in time spent in meetings with practitioners, a 68% decrease in the total number of bugs, and a 53% reduction in the time required to resolve them. Furthermore, adaptability to process changes improved by 44%. Additionally, the TBPA software exhibited a 4% error rate throughout the automation process.

These findings successfully addressed the hypotheses as answers to the research questions. RQ_1 was answered by ensuring a high alignment between process requirements and software architecture, which indeed makes TBPA software more adaptable to process changes. RQ_2 was addressed through the use of logs and process mining, thereby minimizing dependence on practitioners.

Despite this, RWL also presented some limitations. The accurate discovery of business processes still required some degree of human intervention. Ensuring the proper integration of these processes into the software architecture posed challenges that necessitated expert oversight. Additionally, the reliance on extensive data collection raised potential privacy issues. Measures to prevent unauthorized access to sensitive information are crucial to mitigate these concerns. The high volume of data transmission involved in real-time log analysis could lead to network congestion, potentially impacting the performance of the digital ecosystem.

In conclusion, the use of RWL showed a more precise understanding of their business processes, minimized the risks related to subjective biases and errors, and standardized the software architecture to maintain traceability across TBPA software requirements. These capabilities achieved quick adaptability to business process changes and decreased the time spent with stakeholders. As businesses increasingly move towards digital transformation, approaches like RWL become more valuable to organizations.

FUTURE WORKS

Future research will be directed towards addressing the limitations identified in the current study, particularly focusing on refining the RWL methodology to reduce human intervention and enhance its ability to manage high process variability. Additionally, upcoming works will explore the applicability of RWL in RPA, HA, and other processes, aiming to extend its utility across various industrial domains. A key aspect of this future research will involve leveraging Generative Artificial Intelligence to develop a tool that automates the software design generation based on the logs, business process, and the RWL guidelines, which is expected to significantly reduce the complexity of the approach and improve the overall efficiency of the system.

These advancements will collectively contribute to the evolution of RWL, improving robustness and scalability of TBPA solutions.

ACKNOWLEDGMENT

This work is the result of the Research and Development Project Model Automation for Software Development, performed by Sidia Instituto de Ciência e Tecnologia in partnership with Samsung Eletrônica da Amazônia Ltda., using resources from Federal Law No. 8.387/1991, and its disclosure and publicity are under the provisions of Article 39 of Decree No. 10.521/2020.

REFERENCES

[1] W. M. P. van der Aalst, *Data Science in Action*. Berlin, Germany: Springer, 2016, doi: 10.1007/978-3-662-49851-4_1.
[2] W. M. P. van der Aalst, "A practitioner's guide to process mining: Limitations of the directly-follows graph," *Proc. Comput. Sci.*, vol. 164, pp. 321–328, Jan. 2019, doi: 10.1016/j.procs.2019.12.189.
[3] W. M. P. van der Aalst et al., "Process mining manifesto," in *Proc. Bus. Process Management Workshops*. Berlin, Germany: Springer, 2012, pp. 169–194.
[4] W. M. P. van der Aalst, M. Bichler, and A. Heinzl, "Robotic process automation," *Bus. Inf. Syst. Eng.*, vol. 60, no. 4, pp. 269–272, Aug. 2018, doi: 10.1007/s12599-018-0542-4.
[5] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004, doi: 10.1109/TKDE.2004.47.
[6] M. Abbas, A. Ferrari, A. Shatnawi, E. P. Enoiu, and M. Saadatmand, "Is requirements similarity a good proxy for software similarity? An empirical investigation in industry," in *Proc. Int. Working Conf. Requirement Eng., Found. Softw. Quality*, Essen, Germany. Cham, Switzerland: Springer, Jan. 2021, pp. 3–18, doi: 10.1007/978-3-030-73128-1_1.
[7] S. Aguirre and A. Rodríguez, "Automation of a business process using robotic process automation (RPA): A case study," *Commun. Comput. Inf. Sci.*, vol. 742, pp. 65–71, Mar. 2017, doi: 10.1007/978-3-319-66963-2_7.
[8] A. Awad, M. Weidlich, and S. Sakr, "Process mining over unordered event streams," in *Proc. 2nd Int. Conf. Process Mining (ICPM)*, Oct. 2020, pp. 81–88, doi: 10.1109/icpm49681.2020.00022.
[9] B. Axmann and H. Harmoko, "Robotic process automation: An overview and comparison to other technology in industry 4.0," in *Proc. 10th Int. Conf. Adv. Comput. Inf. Technol. (ACIT)*, Sep. 2020, pp. 559–562, doi: 10.1109/ACIT49673.2020.9208907.

[10] B. Aysolmaz, H. Leopold, H. A. Reijers, and O. Demirörs, "A semi-automated approach for generating natural language requirements documents based on business process models," *Inf. Softw. Technol.*, vol. 93, pp. 14–29, Jan. 2018, doi: 10.1016/j.infsof.2017.08.009.
[11] S. Badhiye, P. Chatur, and B. Wakode, "Data logger system: A survey," *Int. J. Comput. Technol. Electron. Eng.*, vol. 24, pp. 24–26, Jul. 2011.
[12] H. O. Barbosa, B. A. Bonifácio, T. M. Menezes, L. F. Uebel, F. B. Pires, and A. F. Neto, "Uma análise do uso de ferramentas em desenvolvimento distribuído de software para atualização da plataforma android," in *Proc. IADIS Ibero Amer. WWW/Internet*, 2019, pp. 39–46.
[13] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed., Reading, MA, USA: Addison-Wesley, 2003.
[14] A. Berti, S. van Zelst, and D. Schuster, "PM4Py: A process mining library for Python," *Softw. Impacts*, vol. 17, Sep. 2023, Art. no. 100556, doi: 10.1016/j.simpa.2023.100556.
[15] S. Bhatia and J. Malhotra, "A survey on impact of lines of code on software complexity," in *Proc. Int. Conf. Adv. Eng. Technol. Res. (ICAETR)*, Aug. 2014, pp. 1–4, doi: 10.1109/ICAETR.2014.7012875.
[16] P. Bornet, I. Barkin, and J. Wirtz, *Intelligent Automation: Welcome to the World of Hyperautomation*. Singapore: World Scientific, 2021, doi: 10.1142/9789811235849_0001.
[17] J. vom Brocke, M. Jans, J. Mendling, and H. A. Reijers, "A five-level framework for research on process mining," *Bus. Inf. Syst. Eng.*, vol. 63, no. 5, pp. 483–490, Oct. 2021, doi: 10.1007/s12599-021-00718-8.
[18] E. C. S. Cardoso, J. P. A. Almeida, and G. Guizzardi, "Requirements engineering based on business process models: A case study," in *Proc. 13th Enterprise Distrib. Object Comput. Conf. Workshops*, Sep. 2009, pp. 320–327, doi: 10.1109/EDOCW.2009.5331974.
[19] T. Diamantopoulos, N. Saoulidis, and A. Symeonidis, "Automated issue assignment using topic modelling on jira issue tracking data," *IET Softw.*, vol. 17, no. 3, pp. 333–344, Jun. 2023, doi: 10.1049/sfw2.12129.
[20] R. Diouf, E. N. Sarr, O. Sall, B. Birregah, M. Bousso, and S. N. Mbaye, "Web scraping: State-of-the-art and areas of application," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 6040–6042, doi: 10.1109/BigData47090.2019.9005594.
[21] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements Of Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Wesley, 1994.
[22] D. Garlan and M. Shaw, "An introduction to software architecture," in *Advances in Software Engineering and Knowledge Engineering*, vol. 1, V. Ambriola and G. Tortora, Eds. NJ, USA: World Scientific Publishing Company, 1993.
[23] Gartner. *Gartner Forecasts Worldwide Hyperautomation-Enabling Software Market to Reach Nearly $600 Billion by 2022*. Accessed: Mar. 1, 2022. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2021-04-28-gartner-forecasts-worldwide-hyperautomation-enabling-software-market-to-reach-nearly-600-billion-by-2022
[24] Gartner. *Gartner Glossary*. Accessed: Feb. 8, 2022. [Online]. Available: https://www.gartner.com/en/glossary
[25] Gartner. *Gartner Says Worldwide Spending on Robotic Process Automation Software to Reach $680 Million in 2018*. Accessed: Aug. 3, 2020. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-11-13-gartner-says-worldwide-spending-on-robotic-process-automation-software-to-reach-680-million-in-2018
[26] M. Ghasemi, "What requirements engineering can learn from process mining," in *Proc. 1st Int. Workshop Learn. Disciplines Requirements Eng. (D4RE)*, Aug. 2018, pp. 8–11, doi: 10.1109/D4RE.2018.00008.
[27] M. Ghasemi, "Towards goal-oriented process mining," in *Proc. IEEE 26th Int. Requirements Eng. Conf. (RE)*, Aug. 2018, pp. 484–489, doi: 10.1109/RE.2018.00066.
[28] M. Gillani, H. A. Niaz, and A. Ullah, "Integration of software architecture in requirements elicitation for rapid software development," *IEEE Access*, vol. 10, pp. 56158–56178, 2022, doi: 10.1109/ACCESS.2022.3177659.
[29] S. Gupta, S. Rani, and A. Dixit, "Recent trends in automation—A study of RPA development tools," in *Proc. 3rd Int. Conf. Recent Develop. Control, Autom. Power Eng. (RDCAPE)*, Oct. 2019, pp. 159–163, doi: 10.1109/RDCAPE47089.2019.8979084.
[30] J. D. Hernandez-Resendiz, E. Tello-Leal, U. M. Ramirez-Alcocer, and B. A. Macías-Hernández, "Semi-automated approach for building event logs for process mining from relational database," *Appl. Sci.*, vol. 12, no. 21, p. 10832, Oct. 2022, doi: 10.3390/app122110832.
[31] I. Herraiz, G. Robles, J. M. Gonzalez-Barahona, A. Capiluppi, and J. F. Ramil, "Comparison between SLOCs and number of files as size metrics for software evolution analysis," in *Proc. Conf. Softw. Maintenance Reeng. (CSMR)*, 2006, pp. 8–pp, doi: 10.1109/CSMR.2006.17.
[32] P. Hofmann, C. Samp, and N. Urbach, "Robotic process automation," *Electron. Markets*, vol. 30, pp. 99–106, Mar. 2020, doi: 10.1007/s12525-019-00365-8.
[33] F. Huang and M. A. Vasarhelyi, "Applying robotic process automation (RPA) in auditing: A framework," *Int. J. Accounting Inf. Syst.*, vol. 35, Dec. 2019, Art. no. 100433, doi: 10.1016/j.accinf.2019.100433.
[34] F. Ip, J. Crowley, and T. Torlone, *Democratizing Artificial Intelligence With UiPath: Expand Automation in Your Organization to Achieve Operational Efficiency and High Performance*. Birmingham, U.K.: Packt Publishing, 2022.
[35] P. Jamshidi and C. Pahl, "Business process and software architecture model co-evolution patterns," in *Proc. 4th Int. Workshop Modeling Softw. Eng. (MISE)*, Jun. 2012, pp. 91–97, doi: 10.1109/MISE.2012.6226021.
[36] M. Khder, "Web scraping or web crawling: State of art, techniques, approaches and application," *Int. J. Adv. Soft Comput. Appl.*, vol. 13, no. 3, pp. 145–168, Dec. 2021, doi: 10.15849/ijasca.211128.11.
[37] T. Kiblawi, M. Muhanna, and A. Qusef, "The role of software architecture in business model transformability," in *Proc. IEEE Jordan Int. Joint Conf. Electr. Eng. Inf. Technol. (JEEIT)*, May 2023, pp. 68–73, doi: 10.1109/jeeit58638.2023.10185701.
[38] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," *IEEE Softw.*, vol. 12, no. 4, pp. 52–62, Jul. 1995, doi: 10.1109/52.391832.
[39] G. Lasso-Rodriguez and K. Winkler, "Hyperautomation to fulfil jobs rather than executing tasks: The BPM manager robot vs human case," *Romanian J. Inf. Technol. Autom. Control/Revista Română de Informatică si Automatică*, vol. 30, no. 3, pp. 7–22, Dec. 2023, doi: 10.33436/v30i3y202001.
[40] P. Lewicki, J. Tochowicz, and J. van Genuchten, "Are robots taking our jobs? A RoboPlatform at a bank," *IEEE Softw.*, vol. 36, no. 3, pp. 101–104, May 2019, doi: 10.1109/MS.2019.2897337.
[41] G. Lucassen, F. Dalpiaz, J. M. Van Der Werf, and S. Brinkkemper, "Bridging the twin peaks - the case of the software industry," in *Proc. IEEE/ACM 5th Int. Workshop Twin Peaks Requirements Archit.*, May 2015, pp. 24–28, doi: 10.1109/TWINPEAKS.2015.13.
[42] Y.-W. Ma, D.-P. Lin, S.-J. Chen, H.-Y. Chu, and J.-L. Chen, "System design and development for robotic process automation," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Dec. 2019, pp. 187–189, doi: 10.1109/SMARTCLOUD.2019.00038.
[43] A. Maalla, "Development prospect and application feasibility analysis of robotic process automation," in *Proc. IEEE 4th Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, vol. 1, Dec. 2019, pp. 2714–2717, doi: 10.1109/IAEAC47372.2019.8997983.
[44] H. M. Marin-Castro and E. Tello-Leal, "Event log preprocessing for process mining: A review," *Appl. Sci.*, vol. 11, no. 22, p. 10556, Nov. 2021, doi: 10.3390/app112210556.
[45] D. Mateus, D. S. da Silveira, and J. Araújo, "A systematic approach to derive user stories and gherkin scenarios from BPMN models," in *Business Modeling and Software Design*, B. Shishkov, Ed., Cham, Switzerland: Springer, 2023, pp. 235–244, doi: 10.1007/978-3-031-36757-1_15.
[46] A. Megargel, C. M. Poskitt, and V. Shankararaman, "Microservices orchestration vs. choreography: A decision framework," in *Proc. IEEE 25th Int. Enterprise Distrib. Object Comput. Conf. (EDOC)*, Oct. 2021, pp. 134–141, doi: 10.1109/EDOC52215.2021.00024.
[47] T. M. de Menezes, "User experience evaluation for automation tools: An industrial experience," *Int. J. Cybern. Informat.*, vol. 11, no. 2, pp. 53–60, Apr. 2022, doi: 10.5121/ijci.2022.110205.
[48] T. Menezes, "A review to find elicitation methods for business process automation software," *Software*, vol. 2, no. 2, pp. 177–196, Mar. 2023, doi: 10.3390/software2020008.
[49] Z. Ming, A. B. Nellippallil, R. Wang, J. K. Allen, G. Wang, Y. Yan, and F. Mistree, "Requirements and architecture of the decision support platform for design engineering 4.0," in *Architecting A Knowledge-Based Platform for Design Engineering 4.0*. Cham, Switzerland: Springer, 2022, pp. 1–22, doi: 10.1007/978-3-030-90521-7_1.
[50] R. Mitchell, *Web Scraping With Python: Collecting More Data From the Modern Web*. Sebastopol, CA, USA: O'Reilly Media, 2018.
[51] R. Mo, S. Wei, Q. Feng, and Z. Li, "An exploratory study of bug prediction at the method level," *Inf. Softw. Technol.*, vol. 144, Apr. 2022, Art. no. 106794, doi: 10.1016/j.infsof.2021.106794.
[52] H. Nordal and I. El-Thalji, "Modeling a predictive maintenance management architecture to meet industry 4.0 requirements: A case study," *Syst. Eng.*, vol. 24, no. 1, pp. 34–50, Jan. 2021.

[53] IEEE Task Force on Process Mining. *Process Mining Manifesto*. Accessed: Oct. 12, 2023. [Online]. Available: https://www.tf-PM.org/upload/1580738212409.pdf
[54] F. C. M. Ortiz and C. J. Costa, "RPA in finance: Supporting portfolio management: Applying a software robot in a portfolio optimization problem," in *Proc. 15th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2020, pp. 1–6, doi: 10.23919/CISTI49556.2020.9141155.
[55] N. A. Panayiotou, S. P. Gayialis, N. P. Evangelopoulos, and P. K. Katimertzoglou, "A business process modeling-enabled requirements engineering framework for ERP implementation," *Bus. Process Manage. J.*, vol. 21, no. 3, pp. 628–664, Jun. 2015, doi: 10.1108/bpmj-06-2014-0051.
[56] A. Parant, F. Gellot, A. Philippot, and V. Carré-Ménétrier, "Model-based engineering for designing cyber-physical systems control architecture and improving adaptability from requirements," in *Proc. Int. Workshop Service Orientation Holonic Multi-Agent Manuf.* Cham, Switzerland: Springer, Jan. 2022, pp. 457–469, doi: 10.1007/978-3-030-99108-1_33.
[57] S. Parchande, A. Shahane, and M. Dhore, "Contractual employee management system using machine learning and robotic process automation," in *Proc. 5th Int. Conf. Comput., Commun., Control Autom. (ICCUBEA)*, Sep. 2019, pp. 1–5, doi: 10.1109/ICCUBEA47591.2019.9128818.
[58] V. Poosapati, V. K. Manda, and V. Katneni, "Cognitive automation opportunities, challenges and applications," *J. Comput. Eng. Technol.*, vol. 9, no. 5, pp. 89–95, 2018.
[59] S. Pourmirza, S. Peters, R. Dijkman, and P. Grefen, "A systematic literature review on the architecture of business process management systems," *Inf. Syst.*, vol. 66, pp. 43–58, Jun. 2017, doi: 10.1016/j.is.2017.01.007.
[60] Roger S. Pressman and Bruce R. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed., New York, NY, USA: McGraw-Hill, 2014.
[61] A. Przybylek, "A business-oriented approach to requirements elicitation," in *Proc. 9th Int. Conf. Eval. Novel Approaches to Softw. Eng. (ENASE)*, Apr. 2014, pp. 1–12.
[62] G. Rathee, F. Ahmad, R. Iqbal, and M. Mukherjee, "Cognitive automation for smart decision-making in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2152–2159, Mar. 2021, doi: 10.1109/TII.2020.3013618.
[63] M. Richards and N. Ford, *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA, USA: O'Reilly Media, 2020.
[64] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, Mar. 2008, doi: 10.1016/j.is.2007.07.001.
[65] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009, doi: 10.1007/s10664-008-9102-8.
[66] S. Saito, "Identifying and understanding stakeholders using process mining: Case study on discovering business processes that involve organizational entities," in *Proc. IEEE 27th Int. Requirement Eng. Conf. Workshops (REW)*, Sep. 2019, pp. 216–219, doi: 10.1109/REW.2019.00045.
[67] S. Sholiq, R. Sarno, and E. S. Astuti, "Generating BPMN diagram from textual requirements," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 10, pp. 10079–10093, Nov. 2022, doi: 10.1016/j.jksuci.2022.10.007.
[68] C. E. Da Silva, L. Medeiros, Y. Justino, and E. L. Gomes, "A box analogy technique (BoAT) for agile-based modelling of business processes," in *Proc. IEEE 30th Int. Requirements Eng. Conf. (RE)*, Aug. 2022, pp. 231–242, doi: 10.1109/RE54965.2022.00029.
[69] S. Silva, A. Tuyishime, T. Santilli, P. Pelliccione, and L. Iovino, "Quality metrics in software architecture," in *Proc. IEEE 20th Int. Conf. Softw. Archit. (ICSA)*, Mar. 2023, pp. 58–69, doi: 10.1109/ICSA56044.2023.00014.
[70] L. Sousa, J. Nascimento, R. Souza, J. Aragão, T. Menezes, and E. Andrade, "BTS-validator: Identificando aplicações potencialmente prejudiciais embarcadas no Android por meio de relatórios," in *Proc. Anais da XX Escola Regional de Redes de Computadores*, Oct. 2023, pp. 115–120, doi: 10.5753/errc.2023.894.
[71] T. Spijkman, S. Brinkkemper, F. Dalpiaz, A.-F. Hemmer, and R. van de Bospoort, "Specification of requirements and software architecture for the customisation of enterprise software: A multi-case study based on the RE4SA model," in *Proc. IEEE 27th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2019, pp. 64–73, doi: 10.1109/REW.2019.00015.
[72] R. Syed, S. Suriadi, M. Adams, W. Bandara, S. J. J. Leemans, C. Ouyang, A. H. M. T. Hofstede, I. van de Weerd, M. T. Wynn, and H. A. Reijers, "Robotic process automation: Contemporary themes and challenges," *Comput. Ind.*, vol. 115, Feb. 2020, Art. no. 103162, doi: 10.1016/j.compind.2019.103162.
[73] D. H. Timbadia, P. J. Shah, S. Sudhanvan, and S. Agrawal, "Robotic process automation through advance process analysis model," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Feb. 2020, pp. 953–959, doi: 10.1109/ICICT48043.2020.9112447.
[74] R. Uskenbayeva, Z. Kalpeyeva, R. Satybaldiyeva, A. Moldagulova, and A. Kassymova, "Applying of RPA in administrative processes of public administration," in *Proc. IEEE 21st Conf. Bus. Informat. (CBI)*, vol. 2, Jul. 2019, pp. 9–12, doi: 10.1109/CBI.2019.10089.
[75] T. Wang and B. Li, "Analyzing software architecture evolvability based on multiple architectural attributes measurements," in *Proc. IEEE 19th Int. Conf. Softw. Quality, Rel. Secur. (QRS)*, Jul. 2019, pp. 204–215, doi: 10.1109/QRS.2019.00037.
[76] J. Wewerka and M. Reichert, "Towards quantifying the effects of robotic process automation," in *Proc. IEEE 24th Int. Enterprise Distrib. Object Comput. Workshop (EDOCW)*, Oct. 2020, pp. 11–19, doi: 10.1109/edocw49879.2020.00015.
[77] W. William and L. William, "Improving corporate secretary productivity using robotic process automation," in *Proc. Int. Conf. Technol. Appl. Artif. Intell. (TAAI)*, Nov. 2019, pp. 1–5, doi: 10.1109/TAAI48200.2019.8959872.
[78] Q. Yao, J. Zhang, and H. Wang, "Business process-oriented software architecture for supporting business process change," in *Proc. Int. Symp. Electron. Commerce Secur.*, 2008, pp. 690–694, doi: 10.1109/ISECS.2008.46.
[79] I. Zada, S. Shahzad, M. N. Alatawi, S. Ali, and J. A. Khan, "LAGSSE: An integrated framework for the realization of sustainable software engineering," *Res. Square*, Aug. 2023, doi: 10.21203/rs.3.rs-3274778/v1.
[80] I. Zada, S. Shahzad, S. Ali, and R. M. Mehmood, "OntoSuSD: Software engineering approaches integration ontology for sustainable software development," *Softw., Pract. Exper.*, vol. 53, no. 2, pp. 283–317, Feb. 2023, doi: 10.1002/spe.3149.

THIAGO MEDEIROS DE MENEZES received the Doctorate degree in software engineering from the CESAR School, in 2024. His research centered on the intersection of software engineering and business process automation with the CESAR School. He has been a Software Developer, since 2011, contributing to organizations in Brazil, Serbia, Slovenia, and Canada. He has developed a wide range of projects, specializing in designing and implementing software solutions tailored to meet business needs. Currently, he acts as a Tech Leader and a Researcher with the Sidia Institute of Science and Technology, where he leads multidisciplinary teams to develop business process automation software. His work focuses on optimizing operational efficiency, improving the quality of processes, and driving innovation within the institute. His academic and professional journey reflects a passion for leveraging technology to solve complex challenges and create impactful solutions.

ANA CAROLINA SALGADO (Member, IEEE) received the Doctorate degree in computer science from the University of Nice, France, in 1988. She acted as a Full Professor with the Center for Informatics of Universidade Federal de Pernambuco (UFPE), Brazil, from 1988 to 2017. Currently, she holds the position of the Academic Director of the CESAR School, a role in which she shapes the future of technology education. Her main research interests include databases and big data integration. She has published more than 150 technical articles in conference proceedings and journals. She has served on program committees of many conferences and acted as referee for international conferences and journals. In her academic activities, she has advised 48 M.Sc. and 19 Ph.D. theses. She is a member of ACM and the Brazilian Computer Society.