

Article

Not peer-reviewed version

Using Logs to Mitigate Process Variability and Dependence on Practitioners in Traditional Business Process Automation Software

[Thiago Medeiros de Menezes](#) * and [Ana Carolina Salgado](#)

Posted Date: 9 December 2024

doi: 10.20944/preprints202408.0776.v3

Keywords: Business Process Automation; Requirements Engineering; requirements specification; software architecture; software design; software development; software engineering





Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Article

Using Logs to Mitigate Process Variability and Dependence on Practitioners in Traditional Business Process Automation Software

Thiago Medeiros de Menezes ^{1,2,*}  and Ana Carolina Salgado ² 

¹ Sidia

² CESAR School

* Correspondence: 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 three approaches employed to develop a BPA. TBPA entails developing BPA in a programming language for integrating the relevant applications in the digital ecosystem to execute a given process. Process variability and practitioner unavailability encumber the requirements specification for TBPA software. **Objective:** This work proposes a log-based approach for TBPA software to make software more adaptable to process changes and reduce dependence on practitioners, 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 the approach and report its results in practice. **Results:** The results revealed significant improvements in adaptability to business process changes and decreased the time spent with practitioners, and, efficiency in development. The approach 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 traceability between process requirements and software architecture, as well as the use of logs and process mining. These methods made TBPA software enhanced the software adaptability to changes and minimized the dependence on practitioners during requirements elicitation respectively.

Keywords: Business Process Automation; Requirements Engineering; requirements specification; software architecture; software design; software development; software engineering

1. Introduction

Business Process Automation (BPA) is widely adopted by organizations to improve efficiency, reduce costs, and increase overall business performance [1–9]. BPA software has improved business processes in auditing firms, banks, outsourcing providers, public entities, software industry, and telecom companies [9].

BPA software performs the digital labor of human beings to automate business processes in a particular digital ecosystem [6,9]. It must consider a variety of information systems and applications with different ages, features, compatibilities, interfaces, and data [6]. While 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 [9].

There are three main approaches to develop a BPA: (1) Traditional Business Process Automation, (2) Robotic Process Automation, and (3) Hyperautomation [9]. Traditional Business Process Automation (TBPA) is the traditional approach, which entails developing BPA software in a programming language for integrating the relevant applications in the digital ecosystem to execute a given process [9,10]. This paper considers Business Process Management 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 [1,5,6,9]. 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 [6]. 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 [9,10].

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

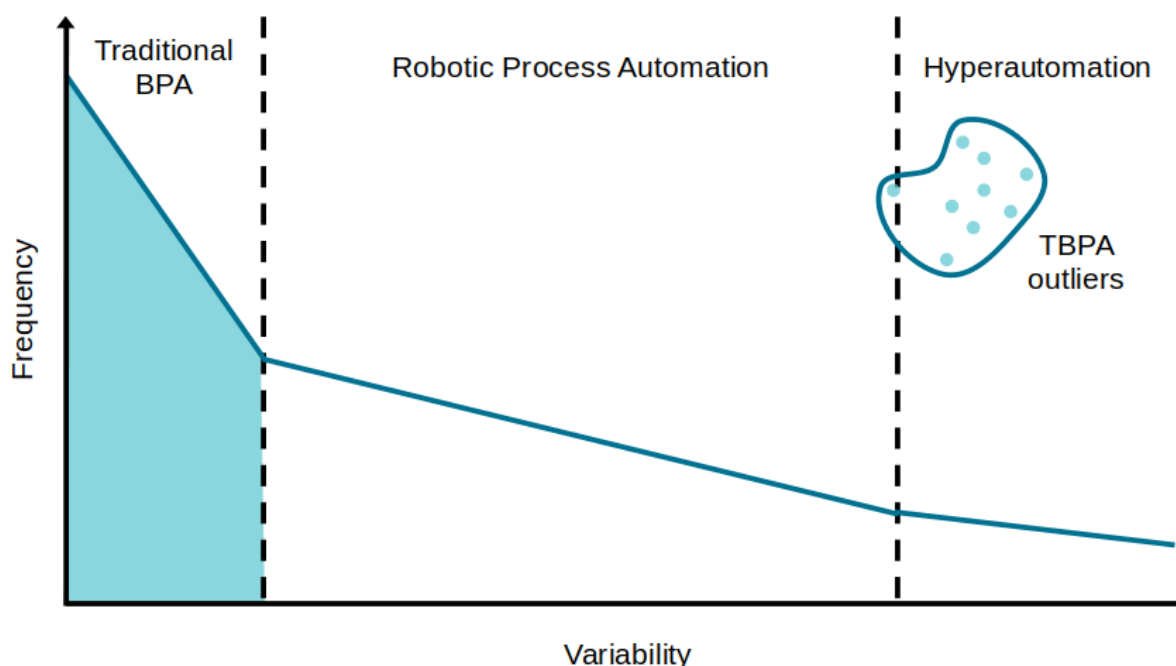


Figure 1. BPA approaches applicability assessment. Adapted from Aalst *et al.* [1].

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 [9]. Hyperautomation is still a vanguard approach [1,10–13] 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 [14,15].

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 [9]. Researchers have provided evidence that software becomes more flexible and capable of adapting to changes when requirements and architecture are well aligned [16–23]. Nonetheless, Menezes [9] 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 [9,24–29] suggested employing event logs and process mining to enhance requirements elicitation. Menezes [9] 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:

RQ₁: How to make TBPA software more adaptable to process changes?

RQ₂: 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.* [4,8,9,30]. 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: Section 2 addresses the background and related works. Section 3 describes the proposed approach. Section 4 presents the case study applied to evaluate the approach and the obtained results. Section 5 discusses the results and relates them to other studies. Finally, Section 6 gives the conclusion.

2. Background and Related Works

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

2.1. Traditional Business Process Automation Software

According to Menezes [9], 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 [31]). It suits to automate repetitive standardized tasks [1,6,9,32], which leads to significant time and cost savings for organizations, as well as greater accuracy and consistency in performing these tasks [3,5,6,9,32–43]. Despite these benefits, Menezes [9] emphasized the challenges in TBPA software development. This research addresses only the ones related to process variability and reliance on practitioners for requirement elicitation.

2.2. 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 [9]. 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 the software changes, studies [16–21] 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. Researchers [17,44–48] introduced different approaches to achieving traceability between requirements and software architecture.

Studies [44–46] proposed different approaches to derive requirements from the business models. Cardoso *et al.* [44] utilized ARchitecture of Integrated Information Systems (ARIS) [49] to specify a Human Resources (HR) system in a large energy company. The researchers manually modeled the business process, derived the requirements from the models, validated them with users, and

finally constructed a prototype. Similarly, Panayiotou *et al.* [45] developed a hybrid model combining business process modeling with Requirements Engineering for Enterprise Resource Planning (ERP) systems in a furniture manufacturer, focusing on the integration of process and technology. The study provides a structured framework suited for ERP implementations, which may involve extensive preliminary modeling and integration efforts. In addition, Aysolmaz *et al.* [46] focused on generating natural language requirements documents from business process models using a semi-automated approach. The approach primarily improved the documentation process by ensuring consistency and completeness in the translation from business process models to requirements. The study [46] was conducted for governmental systems.

While Spijkman *et al.* [17] introduced and applied Requirements for Software Architecture (RE4SA) to customize enterprise software. RE4SA manually links epic stories to software modules and user stories to features to ensure traceability between requirements and architecture. In the study, RE4SA improved communication and collaboration, helped with release planning, prevented architectural drift, and delivered concise and detailed documentation.

Abbas *et al.* [47] conducted research at a railway company to trace requirements and software similarities by using Natural Language Processing (NLP) to increase software reuse for a power propulsion control system. The study employed requirement-level similarity as a proxy for retrieving relevant software. The research [47] obtained 80% software similarity for reuse in 60% of the cases.

Belfadel *et al.* [48] proposed the Enterprise Architecture Capability Profile (EACP), a manual exploitation methodology based on the alignment of enterprise architecture actions with a requirement engineering process. The methodology was designed for service-oriented software to trace the highest compatibility of the software functionalities and the related constraints. According to the authors, the EACP enables the qualification, discovery, reuse, and sustainability of business application development.

Taking insights from the above-cited approaches, this work proposes to utilize high alignment between business process requirements and software architecture to make TBPA software more adaptable to process changes by tracing the process workflow and the involved software in the digital ecosystem into TBPA software components.

2.3. Practitioner Unavailability

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 [9] 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 [9,24–29] 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.

2.4. 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 [50,51].

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 [52–54]. Despite this, the technique relies solely on the existence and availability of logs, potentially neglecting tasks that lack corresponding records [55].

Saito [26] 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.* [29] 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 [1,24–29], 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 [55].

2.5. 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.

Loggers are prevalent in nearly all computing environments to aid in troubleshooting, monitoring, security, and performance analysis [56]. 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.

2.6. 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 [57,58]. 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.

3. 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.* [4,8,9,30], and it was refined over time. RWL uses log analysis and process mining techniques to specify accurate 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 business processes, as well as reducing the reliance 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.

3.1. Approach Overview

Figure 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.

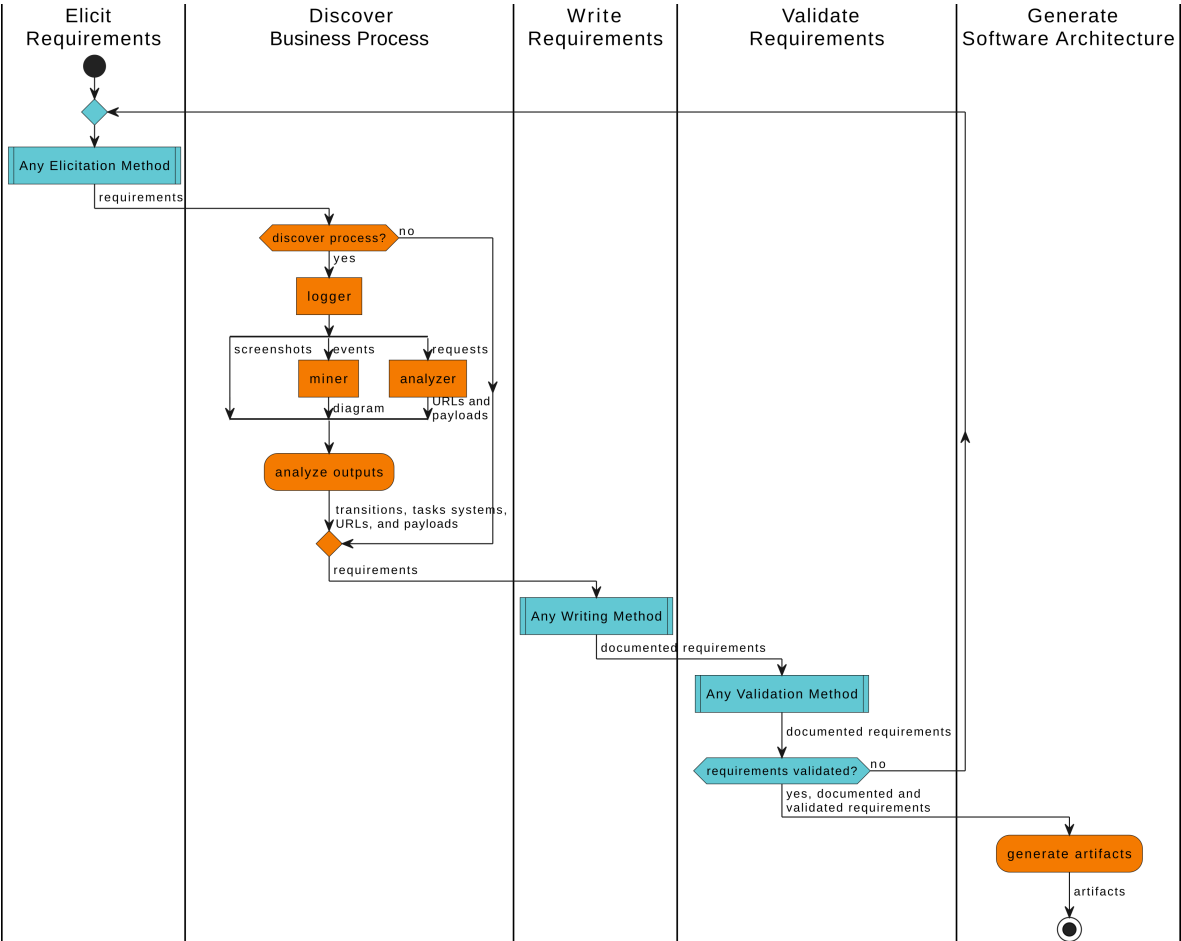


Figure 2. RWL in Requirements Engineering.

3.1.1. 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.

3.1.2. 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

[53,54,59]. This activity outputs the transitions, tasks, systems, Uniform Resource Locators (URLs), payloads, and requirements.

3.1.3. 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.

3.1.4. 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 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.

3.1.5. Generate Software Architecture

Finally, the validated requirements are analyzed to produce the RWL artifacts following the guidelines described in 3.6.

3.2. *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.

Table 1. Logger Data Example.

Activity	UserID	DeviceID	ScreenshotID	PCAPID	Timestamp
PS - Sync Changes	145	226	264877	289784	20230405T150132

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.

3.3. *Process Miner*

There are more than 30 process miners available on the market [60]. 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 [29,51,60].

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 [60].

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.

After mining, the discovered process is evaluated using the metrics of precision, recall, and generalization [29]. These metrics measure the quality of the discovered process based on the behavior observed within the event logs.

Finally, the miner outputs the mined process as a diagram. 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.

3.4. 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.

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.

3.5. 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.

3.6. Software Architecture Generation

Figure 3 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.

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 [31].

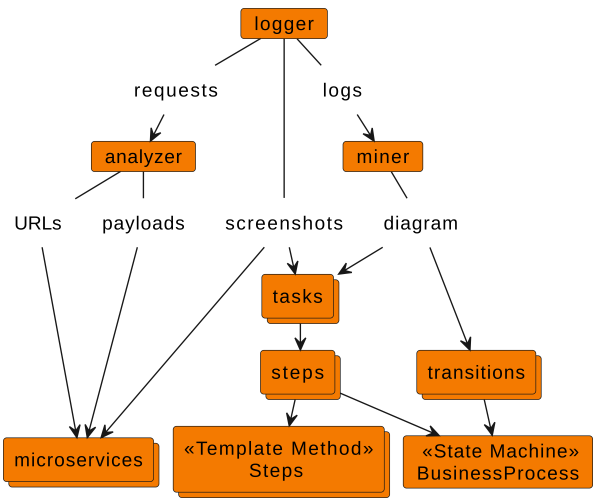


Figure 3. Relationships between outputs and software artifacts.

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 [61], client-server, microservices, domain-driven design, component-based development [62], factory, state machine, and template method [63].

Figure 4 illustrates the RWL architecture to implement TPBA software for a generic business process. Additionally, Figure 5 details the main artifacts of the RWL backend for TBPA software.

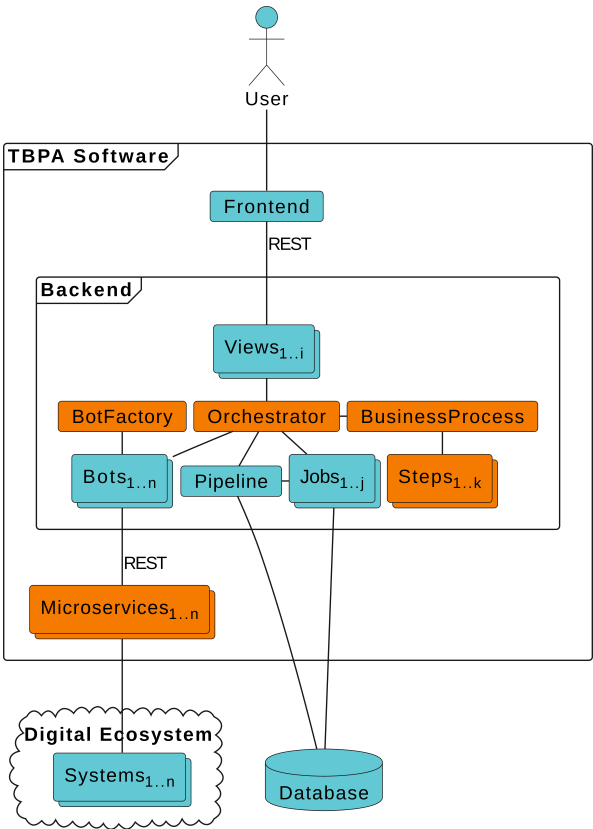


Figure 4. RWL architecture.

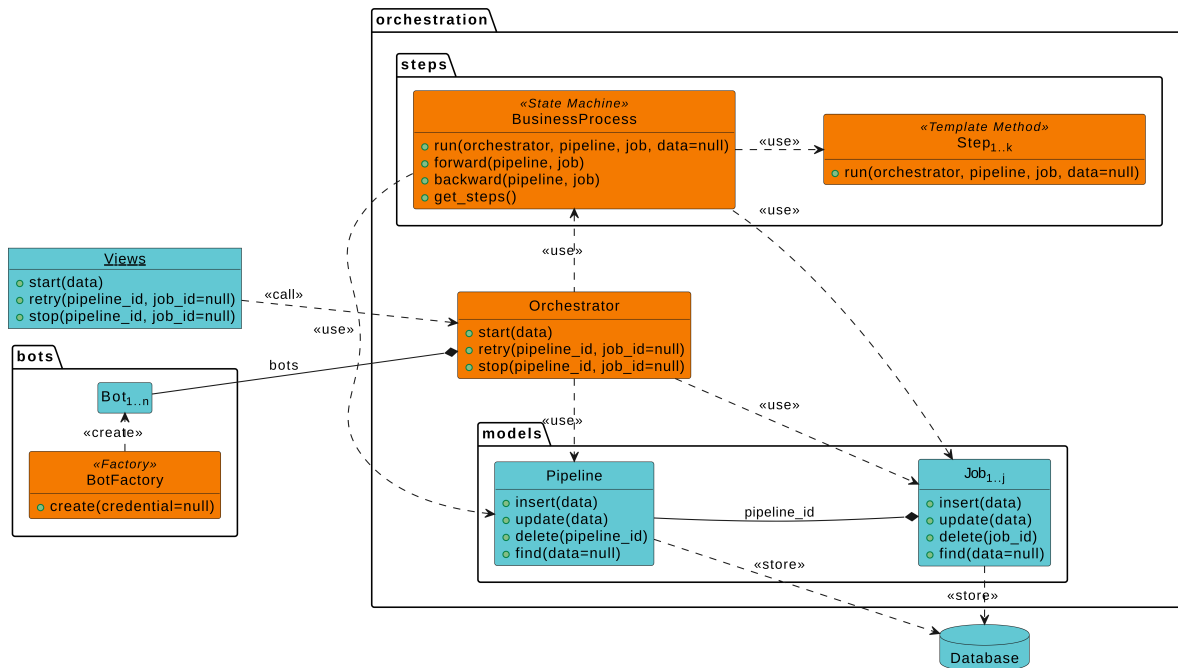


Figure 5. RWL backend design.

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, making it easier to understand and manage, ensuring that the process is always followed correctly [63]. 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 [63]. 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 [62]. 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 [61]. 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 [63].

Finally, Views are responsible for handling REST requests from the Frontend, calling the methods from the Orchestrator, and returning the respective responses [62]. 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.

4. 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 [64,65], 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.

4.1. Objectives and Hypotheses

Considering the research questions presented in Section 1, 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 reliance 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 [16–21,44,66–69], a high alignment between requirements and architecture makes software more adaptable to changes. Moreover, Menezes [9] argued that the usage of logs and process mining has the potential to reduce reliance on practitioners. Based on this context, the following hypotheses were formulated to answer the research questions:

- For RQ₁:
 - H₁: High traceability between business process requirements and software architecture improves the adaptability of TBPA software to process changes.
- For RQ₂:
 - H₂: Logs and process mining aid elicitors to discover the digital ecosystem technologies and the business process without assistance from practitioners;
 - H₃: Logs and process mining give an overview of the whole business process and the digital ecosystem that assists elicitors to elicit more precise and reliable requirements, which has the potential to reduce the reliance on practitioners;

4.2. 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) [4]. The institute has a dedicated team responsible for developing TBPA software to automate these processes.

Figure 6 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.

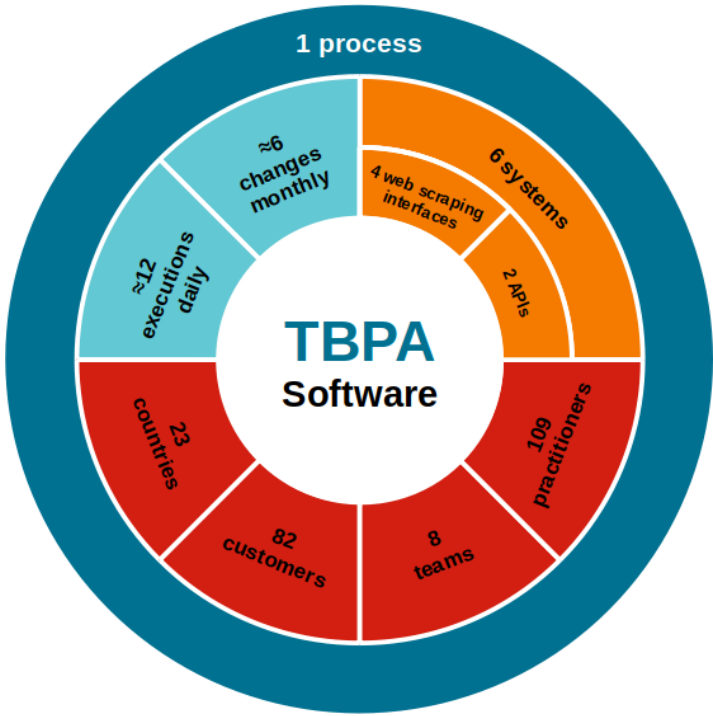


Figure 6. TBPA software for automating device specification updates for Android in Latin America.

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

4.3. 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 [70];
- 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 [71];
 - *Total Practitioner Meetings* (TPM): amount of issues due to clarify the development of an issue with practitioners [71].

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

- Data related to development efficiency:
 - *Bug Resolution Time* (BRT): time, in hours, taken to resolve a bug [70];
 - *Issue Resolution Time* (IRT): time, in hours, taken to complete an issue (bug, improvement, process change, practitioner meeting, task, or other) [70];
 - *Total Bugs* (TB): amount of bugs reported within the project [71,72];
 - *Total Issues* (TI): amount of all issues within the project (bugs, improvements, process changes, practitioner meetings, tasks, and others) [71].
- Data related to BPA performance:

- *Completion Pipeline Time* (CPT): time, in minutes, taken by the TBPA software to complete an pipeline correctly [71];
- *Total Completed Pipelines* (TCP): amount of pipelines executed and finished correctly by the TBPA software [71];
- Data related to code size:
 - *Number of Files* (NoF): amount of files found within the project [73];
 - *Lines of Code* (LoC): amount of code lines found within the project [73–75];

4.4. Sister Project

This work utilized Replicated Product Design (RPD), a method that consists of replicating the existing product with a new method or tool and measuring the metrics for both versions [64]. This way, a new version of the same TBPA software was developed employing RWL. TBPAS₁ denotes the previous version without RWL, while TBPAS₂ refers to the new one with the approach.

TBPAS₁ was developed by a team of 20 developers 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, TBPAS₂ started in February, 2022. It had a different team of 14 developers and 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. Figure 7 compares the distribution of roles for each TBPAS development team.

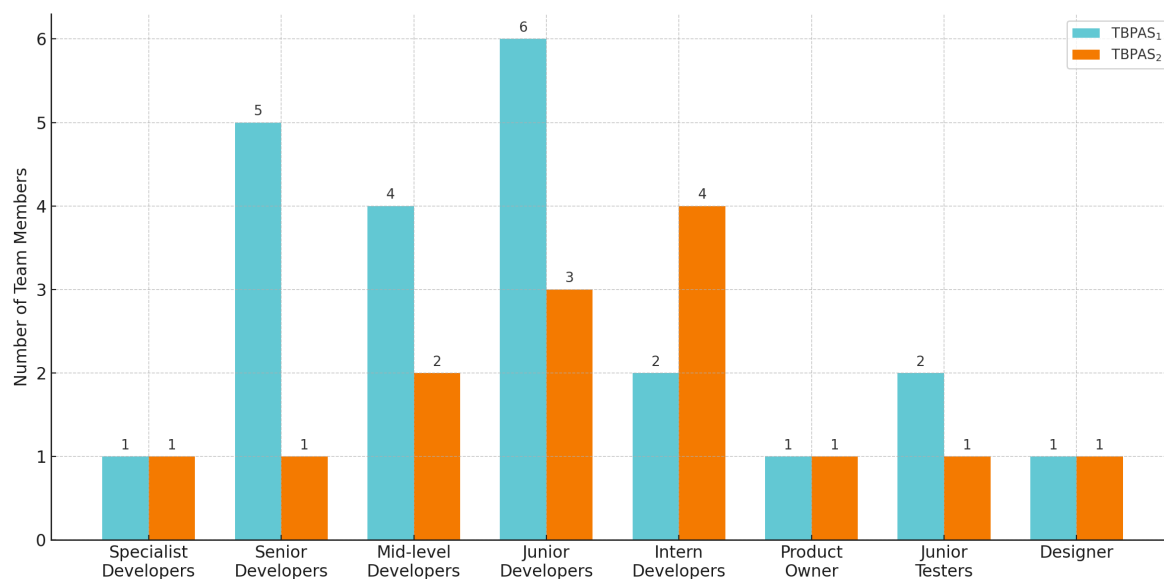


Figure 7. Comparison of TBPAS₁ and TBPAS₂ development teams.

The TBPAS₁ development team consisted of 1 specialist, 5 senior developers, 4 mid-level developers, 6 junior developers, 2 interns, 1 product owner, 2 junior testers, and 1 designer. In contrast, the TBPAS₂ development team was smaller, comprising 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.

In addition, Figures 2, 4, and 5, introduced in Section 3, also overview the main differences between the versions, in which blue items represent TBPAS₁, and blue and orange items TBPAS₂.

4.5. 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.

4.6. Results

In this study, data analysis was conducted using quantitative methods by comparing metrics [64]. The comparison results were calculated using the equations below.

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.

$$Metric_{Result} = \frac{Metric_{TBPAS_1} - Metric_{TBPAS_2}}{Metric_{TBPAS_1}} \times 100\% \tag{1}$$

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

$$TCP_{Result} = \frac{TCP_{TBPAS_2} - TCP_{TBPAS_1}}{TCP_{TBPAS_1}} \times 100\% \tag{2}$$

Table 2 compares the metrics of both versions. The metrics are grouped by equation and arranged in descending order, highlighting the best results at the top.

Table 2. TBPA Software Development Performance Metrics

Metric	Unit	TBPAS ₁	TBPAS ₂	Result
		01/05/2020 01/07/2022	01/02/2022 17/08/2023	
Equation 1				
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				
Total Completed Pipelines (TCP)	%	58	96	65%

Figure 8 illustrates the comparison of metrics for TBPAS₁ and TBPAS₂. Each chart within the figure represents a specific performance metric, allowing for a clear visual comparison between the two versions.

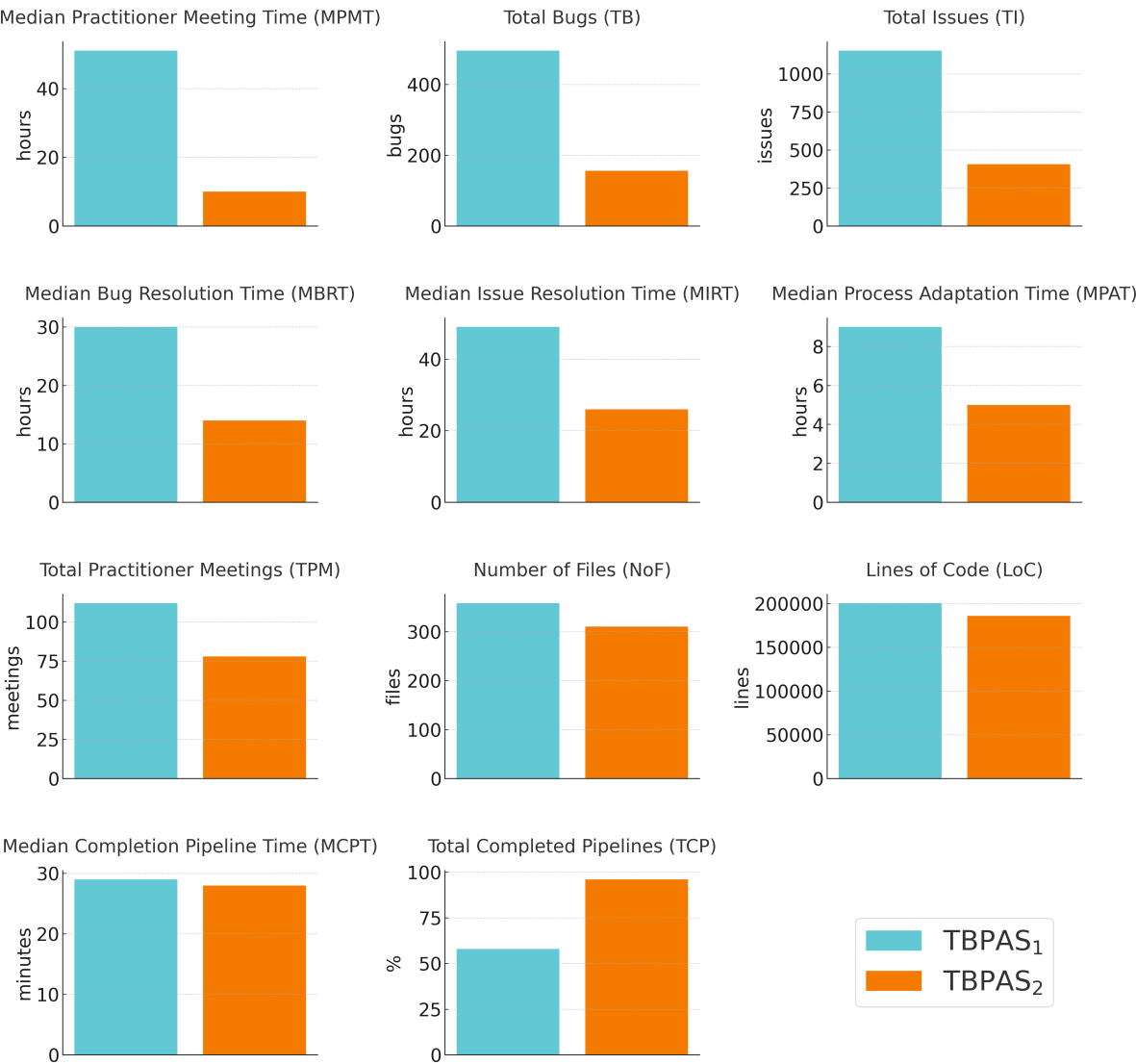


Figure 8. TBPAS Software Development Performance Metrics.

Table 3 presents the results from Shapiro-Wilk and Mann-Whitney U tests for time-related metrics. Shapiro-Wilk assesses the normality of the data. For all metrics, 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 metrics did not follow a normal distribution for either of the versions. This justifies the use of non-parametric tests like the Mann-Whitney U in this context. Mann-Whitney U was used to compare the distributions of the time data between both TBPAS versions. All metrics yielded extremely low p-values, indicating that TBPAS₁ and TBPAS₂ have distinct time distributions in all five metrics.

Table 4 shows the distribution of time for each related metrics across both versions. The sample sizes for TBPAS₁ and TBPAS₂ indicate that TBPAS₁ had significantly more data points across all variables. Looking at the percentiles, it is clear that TBPAS₁ generally has higher values across the board, indicating longer durations for issues, bugs, changes, and meetings compared to TBPAS₂. These results suggest that TBPAS₁ typically requires more time to resolve issues, fix bugs, implement process changes, and have meetings with practitioners compared to TBPAS₂, which may reflect differences in operational efficiency.

Table 3. Statistical Test Results for Independent Samples

Data	Shapiro-Wilk		Mann-Whitney U	
	W	p	Statistic	p
Issue Resolution Time (IRT)	0.723	<.001	131792	<.001
Bug Resolution Time (BRT)	0.837	<.001	21421	<.001
Process Adaptation Time (PAT)	0.941	<.001	1330	<.001
Practitioner Meeting Time (PMT)	0.877	<.001	758	<.001
Completion Pipeline Time (CPT)	0.456	<.001	48663	0.084

Table 4. Descriptive Results

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

Data analysis reveals significant differences between TBPAS₁ and TBPAS₂ in terms of the time required to handle issues, bugs, process changes, and meetings. TBPAS₁ consistently shows higher values across all percentiles, suggesting longer time for these activities compared to TBPAS₂. The Mann-Whitney U test confirms that these differences are statistically significant, and the Shapiro-Wilk test indicates that the data distributions are not normal. This may point to variations in the complexity, efficiency, or operational procedures between both automations.

PAT has decreased by 44%, from 9 to 5 hours. Such reduction supports hypothesis H₁. 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.

The reduction in PMT and TPM supported H₂ and H₃. They showcased a shift towards 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 also improved. BRT and IRT decreased by 53% and 47%, respectively, indicating faster resolution times. Furthermore, TB and TI decreased by 68% and 65%, respectively. Such reductions, corroborating with PAT, demonstrate the benefits of making TBPA software more adaptable to process changes. In particular, the decreases in TB and TI also indicate that 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 gains obtained in CPT were not relevant, the gains in TCP indicated a marked improvement in the efficiency and completion rates of the pipeline, which underlined the effectiveness of the approach.

Lastly, NoF and LoC demonstrated controlled reductions in code size. 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 9 and 10 present the created versus resolved issues report for TBPAS₁ and TBPAS₂, 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 a production environment.

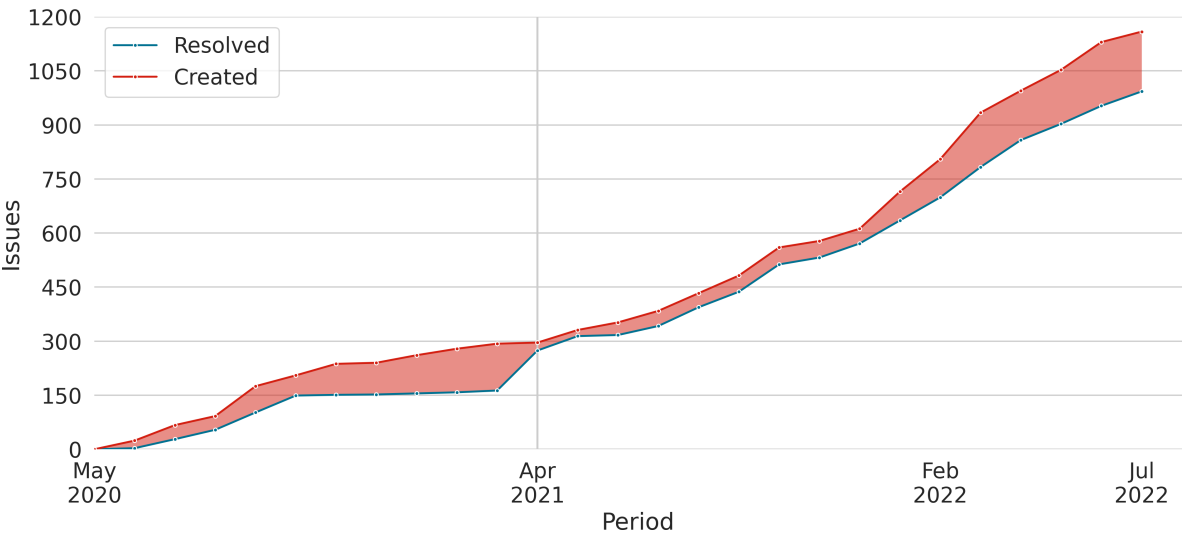


Figure 9. Created vs. resolved issues report for TBPAS₁.

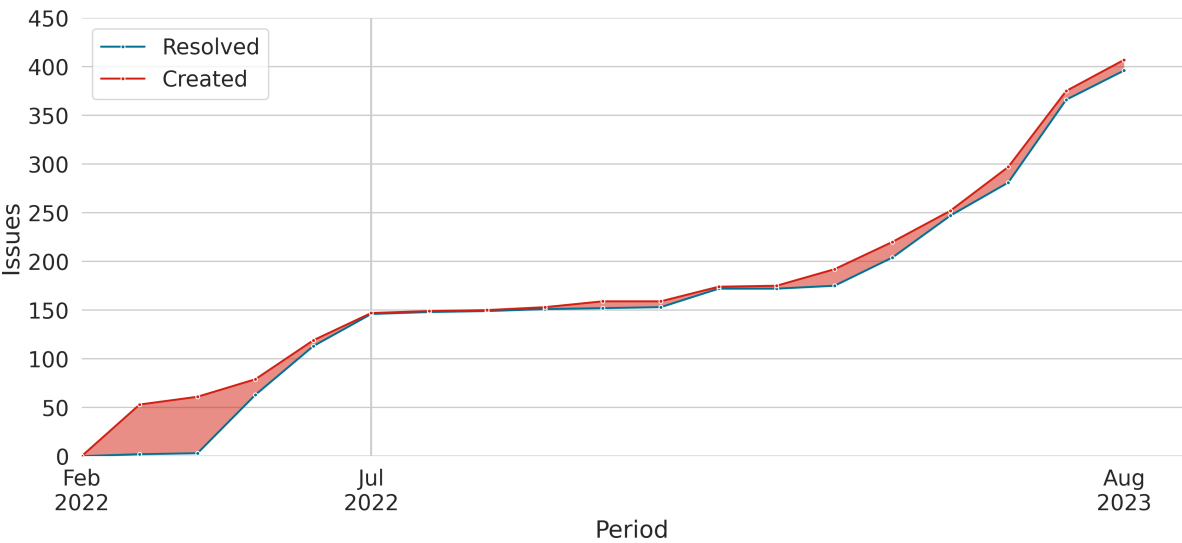


Figure 10. Created vs. resolved issues report for TBPAS₂.

Figure 9 reveals a ratio between created and resolved issues, indicating that the TBPAS₁ 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, Figure 10 demonstrates that TBPAS₂ team effectively delivered results, keeping up with the incoming workload.

In summary, the results for PAT, BRT, IRT, TB, TI, and TCP supported H₁, while PMT, TPM, TB, and TI confirmed H₂ and H₃. TBPAS₂ presented fewer meetings and issues. TBPAS₂ also reduced the time spent with bug fixes, process changes, general issues, and meetings with practitioners. 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, TBPAS₂

outperformed TBPAS₁. For this reason, TBPAS₁ was considered deprecated, and it was replaced by TBPAS₂ in July 2022.

4.7. 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 TBPAS₂ 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 speeds 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.

4.8. Threats to Validity

It is important to highlight that two developers have already worked on both TBPA versions. They had significant knowledge about the processes as well. Together, these developers worked on 138 of 1153 issues in TBPAS₁ (12%) and 126 of 407 issues in TBPAS₂ (31%), which may introduce bias into the results.

4.9. 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.

5. 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 reliance 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. Figures 9 and 10 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 [9,25,26] and studies [16–21]. 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 reliance 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.

Table 5. Comparative Summary of Approaches

Approach	Method	Benefits	Limitations
RWL	It utilizes log analysis and process mining to refine requirements and generate a standardized software architecture for TBPA software.	Enhances the speed and accuracy of requirement elicitation, ensures and improves traceability between software specifications and business processes, even with process changes.	It requires human intervention to precisely obtain the business process and trace it into the architecture, can be complex to implement, and may result in data privacy issues and network traffic overload.
	[44] It employs business process models to systematically extract and document software requirements.	It promotes clear communication through visual models, improves alignment between software and business processes, and enhances traceability.	Managing large models can be challenging and resource-intensive, dependent on the quality of the models.
	[45] It integrates business process modeling with ERP system requirements to enhance customization and alignment.	It detailed process documentation, supports customization, and improves specification precision.	High complexity and resource demands, dependence on accurate modeling, and potential for over-engineering.
	[46] It generates requirements documents from business process models using semi-automated tools to bridge the gap between process models and requirements.	Automation increases efficiency, reduces inconsistencies, and enhances traceability between requirements and architecture.	Dependent on model quality, requires a high initial investment, and focuses primarily on process-driven requirements.
	[17] It integrates requirements with software architecture to manage complex software projects.	Promotes better communication, enables concurrent development, and supports systematic documentation.	Complex to implement, dependent on the quality of the initial requirements, and potential for over-engineering.
	[26] It uses process mining to identify and model business processes that involve various organizational entities.	It identifies stakeholders accurately, generates detailed documentation, and facilitates system customization.	Dependent on detailed logs, computationally intensive, and requires high-quality data.
	[47] It utilizes NLP techniques to link software requirements with similar existing software components for code reuse.	It provides efficient requirement retrieval, enhances mapping accuracy, and supports the identification of reusable components.	High dependency on data quality, complex implementation, and potential misalignment of requirements with code.
	[48] It combines requirements with enterprise architecture to efficiently reuse existing software capabilities.	It enhances compatibility between stakeholder requirements and solutions, promotes systematic reuse of software capabilities, and aligns solutions with business goals.	Requires significant expertise, is reliant on accurate architectural models, has scalability concerns and limited flexibility for rapid development cycles.

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 [44], ERP for furniture manufacturer [45], governmental systems [46], enterprise software customization [17], industry software [26], power propulsion control software [47], and service-oriented software [48]. Each context addresses specific challenges in which similarities and differences have emerged.

Regarding requirements elicitation involving stakeholders, RWL and Saito [26] differ significantly from other approaches by utilizing semi-automated methods based on logs and process mining. The former concentrated on process workflow and the involved systems, while the latter sought to identify organizational entities. Similarly, the study [46] employed a semi-automated method to translate business process models into natural language requirements, but it focused on documentation consistency rather than real-time software adaptability. In contrast, researchers [44,45] involved stakeholders through the use of business process models. Both approaches emphasized clear communication and collaboration with stakeholders to visually map out and document the requirements. This engagement helped ensure that the requirements were well understood and agreed upon by all parties, fostering better alignment between business needs and the software being developed. On the other hand, the studies [17,48] utilized broader enterprise architecture frameworks, which promoted a more holistic view of the digital ecosystem and the business strategy and improved stakeholder communication and concurrent development. However, the works [17,44,45,48] relied on the accuracy of the available data that have risks associated with potential inconsistencies and variations in understanding of stakeholders or outdated documentation [1,9,50]. Compared to RWL, the approaches highlighted the essential role of stakeholders in providing context and validating the requirements, showing the need for balanced stakeholder engagement and automated data analysis for optimal results. In addition, the requirements can be challenging to manage and maintain, particularly in large or complex projects.

In terms of adaptability to support constant changes, RWL adapts software to process changes through continuous log analysis and process mining. This real-time feedback loop and the standardized architecture allow rapid implementation on software, ensuring the TBPA software remains aligned with evolving business processes. Nevertheless, the approach still requires human intervention to have an effective analysis of the business process, and the initial technical complexity of implementing and maintaining such a system can be substantial. Approaches such as the ones introduced by studies [44,45] offer adaptability by involving stakeholders in updating visual models to reflect process changes. While the method ensures thorough understanding and consensus, it can be resource-intensive and slow to implement. The approaches proposed by researchers [17,48] provide adaptability through systematic documentation. In contrast, the work [47] allows adaptability by leveraging textual analysis to identify similarities between new requirements and existing software components. This method is particularly useful in large organizations with extensive software libraries, enabling efficient reuse of code and reducing development time and costs. Unlike RWL, which is dependent on the logs, the NLP approach can extract value from existing documentation and codes, making it versatile across different technological environments. However, these approaches still depend on the initial quality of the data and stakeholder engagement to remain effective.

Due to its strengths, RWL stands out when considering software development efficiency. The logs of the process execution and the mined process enhance requirement elicitation, minimizing human error and reducing the time and effort typically associated with traditional elicitation. The software architecture allows quick traceability between the requirements and the code, decreasing the need for extensive rework and iterative cycles typically required to address misaligned requirements. Thus, the continuous refinements and the inherent architecture make RWL uniquely suited for environments where business processes are frequently changing, thus maintaining alignment with operational needs.

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.

6. Conclusion

This research introduced RWL, a novel approach to Requirements Engineering in TBPA based on employing log analysis and process mining techniques. The application of RWL has demonstrated a significant reduction in reliance on practitioner input during requirements elicitation, along with enhanced adaptability of software to process changes.

The case study at a technology institute showed that the integration of logs and process mining could systematically improve the alignment between business process requirements and software architecture. The findings 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.

Therefore, the findings successfully addressed the hypotheses as answers to the research questions. RQ₁ was answered by ensuring a high alignment between process requirements and software architecture, which indeed makes TBPA software more adaptable to process changes. RQ₂ was addressed through the use of logs and process mining, thereby minimizing reliance on practitioners.

In conclusion, RWL provided 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.

7. Future Works

Future works will apply RWL to develop TBPA software for other processes and report the results. They will also explore how to improve the accuracy of the discovered business process and generate the software architecture automatically using Generative Artificial Intelligence in order to reduce the approach's complexity. Additionally, such works will investigate the applicability of RWL in RPA and HA, the reasons behind the decrease in software size, and verify whether RWL software design may reduce the size and complexity.

Acknowledgments: This work is the result of the R&D 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. van der Aalst, W.M.; Bichler, M.; Heinzl, A. Robotic Process Automation. *Business and Information Systems Engineering* **2018**, *60*, 269–272. doi:10.1007/s12599-018-0542-4.
2. Gartner Says Worldwide Spending on Robotic Process Automation Software to Reach \$680 Million in 2018. <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>, 2018. Accessed at 2020-08-03.
3. Lewicki, P.; Tochowicz, J.; van Genuchten, J. Are Robots Taking Our Jobs? A RoboPlatform at a Bank. *IEEE Software* **2019**, *36*, 101–104. doi:10.1109/ms.2019.2897337.

4. Barbosa, H.O.; Bonifácio, B.; Menezes, T.M.d.; Uebel, L.F.; Pires, F.B.; Neto, A.F. Uma Análise do Uso de Ferramentas em Desenvolvimento Distribuído de Software para Atualização da Plataforma Android. *WWW/INTERNET 2019* **2019**, pp. 39–46.
5. Axmann, B.; Harmoko, H. Robotic Process Automation: An Overview and Comparison to Other Technology in Industry 4.0. 2020 10th International Conference on Advanced Computer Information Technologies (ACIT), 2020, pp. 559–562. doi:10.1109/acit49673.2020.9208907.
6. Hofmann, P.; Samp, C.; Urbach, N. Robotic process automation. *Electronic Markets* **2020**, *30*, 99–106. doi:10.1007/s12525-019-00365-8.
7. Gartner Forecasts Worldwide Hyperautomation-Enabling Software Market to Reach Nearly \$600 Billion by 2022. <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>, 2021. Accessed at 2022-03-01.
8. de Menezes, T.M. User Experience Evaluation for Automation Tools: An Industrial Experience. *International Journal on Cybernetics and Informatics* **2022**, *11*, 53–60. doi:10.5121/ijci.2022.110205.
9. Menezes, T. A Review to Find Elicitation Methods for Business Process Automation Software. *Software* **2023**, *2*, 177–196.
10. Bornet, P.; Barkin, I.; Wirtz, J. *INTELLIGENT AUTOMATION: Welcome to the World of HYPERAUTOMATION: Learn How to Harness Artificial Intelligence to Boost Business & Make Our World More Human*; World Scientific, 2021.
11. Gartner. Gartner Glossary. <https://www.gartner.com/en/glossary>, s.d. Accessed at 2022-02-08.
12. Rathee, G.; Ahmad, F.; Iqbal, R.; Mukherjee, M. Cognitive automation for smart decision-making in industrial internet of things. *IEEE Transactions on Industrial Informatics* **2020**, *17*, 2152–2159.
13. Ip, F.; Crowley, J.; Torlone, T. *Democratizing Artificial Intelligence with UiPath: Expand automation in your organization to achieve operational efficiency and high performance*; Packt Publishing, 2022.
14. Poosapati, V.; Manda, V.K.; Katneni, V. Cognitive Automation Opportunities, Challenges and Applications. *Journal of Computer Engineering and Technology* **2018**, *9*, 89–95.
15. LASSO-RODRIGUEZ, G.; Winkler, K. Hyperautomation to fulfil jobs rather than executing tasks: the BPM manager robot vs human case. *Romanian Journal of Information Technology & Automatic Control/Revista Română de Informatică și Automatică* **2020**, *30*.
16. Lucassen, G.; Dalpiaz, F.; Van Der Werf, J.M.; Brinkkemper, S. Bridging the Twin Peaks—The Case of the Software Industry. 2015 IEEE/ACM 5th International Workshop on the Twin Peaks of Requirements and Architecture. IEEE, 2015, pp. 24–28.
17. Spijkman, T.; Brinkkemper, S.; Dalpiaz, F.; Hemmer, A.F.; van de Bospoort, R. Specification of requirements and software architecture for the customisation of enterprise software: A multi-case study based on the RE4SA model. 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). IEEE, 2019, pp. 64–73.
18. Nordal, H.; El-Thalji, I. Modeling a predictive maintenance management architecture to meet industry 4.0 requirements: A case study. *Systems Engineering* **2021**, *24*, 34–50.
19. Parant, A.; Gellot, F.; Philippot, A.; Carré-Ménétrier, V. Model-based engineering for designing cyber-physical systems control architecture and improving adaptability from requirements. *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, 2021, pp. 457–469.
20. Gillani, M.; Niaz, H.A.; Ullah, A. Integration of Software Architecture in Requirements Elicitation for Rapid Software Development. *IEEE Access* **2022**, *10*, 56158–56178.
21. Ming, Z.; Nellippallil, A.B.; Wang, R.; Allen, J.K.; Wang, G.; Yan, Y.; Mistree, F. Requirements and architecture of the decision support platform for design engineering 4.0. In *Architecting a Knowledge-Based Platform for Design Engineering 4.0*; Springer, 2022; pp. 1–22.
22. Zada, I.; Shahzad, S.; Ali, S.; Mehmood, R.M. OntoSuSD: Software engineering approaches integration ontology for sustainable software development. *Software: Practice and Experience* **2023**, *53*, 283–317.
23. Zada, I.; Shahzad, S.; Alatawi, M.N.; Ali, S.; Khan, J.A. LAGSSE: An Integrated Framework for the Realization of Sustainable Software Engineering **2023**.
24. Ghasemi, M. Towards Goal-Oriented Process Mining. 2018 IEEE 26th International Requirements Engineering Conference (RE), 2018, pp. 484–489. doi:10.1109/re.2018.00066.

25. Ghasemi, M. What Requirements Engineering can Learn from Process Mining. 2018 1st International Workshop on Learning from other Disciplines for Requirements Engineering (D4RE), 2018, pp. 8–11. doi:10.1109/d4re.2018.00008.
26. Saito, S. Identifying and Understanding Stakeholders using Process Mining: Case Study on Discovering Business Processes that Involve Organizational Entities. 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), 2019, pp. 216–219. doi:10.1109/rew.2019.00045.
27. vom Brocke, J.; Jans, M.; Mendling, J.; Reijers, H.A. A five-level framework for research on process mining, 2021.
28. Marin-Castro, H.M.; Tello-Leal, E. Event log preprocessing for process mining: a review. *Applied Sciences* **2021**, *11*, 10556.
29. Hernandez-Resendiz, J.D.; Tello-Leal, E.; Ramirez-Alcocer, U.M.; Macías-Hernández, B.A. Semi-Automated Approach for Building Event Logs for Process Mining from Relational Database. *Applied Sciences* **2022**, *12*, 10832.
30. Sousa, L.; Nascimento, J.; Souza, R.; Aragao, J.; Menezes, T.; Andrade, E. BTS-Validator: identificando Aplicações Potencialmente Prejudiciais embarcadas no Android por meio de relatórios. Anais da XX Escola Regional de Redes de Computadores; SBC: Porto Alegre, RS, Brasil, 2023; pp. 115–120. doi:10.5753/errc.2023.894.
31. Diouf, R.; Sarr, E.N.; Sall, O.; Birregah, B.; Bouusso, M.; Mbaye, S.N. Web scraping: state-of-the-art and areas of application. 2019 IEEE International Conference on Big Data (Big Data). IEEE, 2019, pp. 6040–6042.
32. Uskenbayeva, R.; Kalpeyeva, Z.; Satybaldiyeva, R.; Moldagulova, A.; Kassymova, A. Applying of RPA in Administrative Processes of Public Administration. 2019 IEEE 21st Conference on Business Informatics (CBI), 2019, Vol. 02, pp. 9–12. doi:10.1109/cbi.2019.10089.
33. Aguirre, S.; Rodriguez, A. Automation of a business process using robotic process automation (RPA): A case study. *Communications in Computer and Information Science* **2017**, *742*, 65–71. doi:10.1007/978-3-319-66963-2_7.
34. Gupta, S.; Rani, S.; Dixit, A. Recent Trends in Automation-A study of RPA Development Tools. 2019 3rd International Conference on Recent Developments in Control, Automation Power Engineering (RDCAPE), 2019, pp. 159–163. doi:10.1109/rdcape47089.2019.8979084.
35. Huang, F.; Vasarhelyi, M.A. Applying robotic process automation (RPA) in auditing: A framework. *International Journal of Accounting Information Systems* **2019**, *35*, 100433. <https://doi.org/10.1016/j.accinf.2019.100433>.
36. Parchande, S.; Shahane, A.; Dhore, M. Contractual Employee Management System Using Machine Learning and Robotic Process Automation. 2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA), 2019, pp. 1–5. doi:10.1109/iccubea47591.2019.9128818.
37. Ma, Y.; Lin, D.; Chen, S.; Chu, H.; Chen, J. System Design and Development for Robotic Process Automation. 2019 IEEE International Conference on Smart Cloud (SmartCloud), 2019, pp. 187–189. doi:10.1109/SmartCloud.2019.00038.
38. Maalla, A. Development Prospect and Application Feasibility Analysis of Robotic Process Automation. 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2019, Vol. 1, pp. 2714–2717. doi:10.1109/iaeac47372.2019.8997983.
39. Ortiz, F.C.M.; Costa, C.J. RPA in Finance: supporting portfolio management : Applying a software robot in a portfolio optimization problem. 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), 2020, pp. 1–6. doi:10.23919/cisti49556.2020.9141155.
40. Syed, R.; Suriadi, S.; Adams, M.; Bandara, W.; Leemans, S.J.; Ouyang, C.; ter Hofstede, A.H.; van de Weerd, I.; Wynn, M.T.; Reijers, H.A. Robotic Process Automation: Contemporary themes and challenges. *Computers in Industry* **2020**, *115*, 103162. doi:<https://doi.org/10.1016/j.compind.2019.103162>.
41. Timbadia, D.H.; Jigishu Shah, P.; Sudhanvan, S.; Agrawal, S. Robotic Process Automation Through Advance Process Analysis Model. 2020 International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 953–959. doi:10.1109/iciict48043.2020.9112447.
42. Wewerka, J.; Reichert, M. Towards Quantifying the Effects of Robotic Process Automation. 2020 IEEE 24th International Enterprise Distributed Object Computing Workshop (EDOCW), 2020, pp. 11–19. <https://doi.org/10.1109/edocw49879.2020.00015>.
43. William, W.; William, L. Improving Corporate Secretary Productivity using Robotic Process Automation. 2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), 2019, pp. 1–5. doi:10.1109/taai48200.2019.8959872.

44. Cardoso, E.C.S.; Almeida, J.P.A.; Guizzardi, G. Requirements engineering based on business process models: A case study. 2009 13th Enterprise Distributed Object Computing Conference Workshops. IEEE, 2009, pp. 320–327.
45. Panayiotou, N.; Gayialis, S.P.; Evangelopoulos, N.P.; Katimertzoglou, P.K. A business process modeling-enabled requirements engineering framework for ERP implementation. *Bus. Process. Manag. J.* **2015**, *21*, 628–664. doi:10.1108/BPMJ-06-2014-0051.
46. Aysolmaz, B.; Leopold, H.; Reijers, H.; Demirörs, O. A semi-automated approach for generating natural language requirements documents based on business process models. *Inf. Softw. Technol.* **2018**, *93*, 14–29. doi:10.1016/J.INFSOF.2017.08.009.
47. Abbas, M.; Ferrari, A.; Shatnawi, A.; Enou, E.P.; Saadatmand, M. Is requirements similarity a good proxy for software similarity? an empirical investigation in industry. Requirements Engineering: Foundation for Software Quality: 27th International Working Conference, REFSQ 2021, Essen, Germany, April 12–15, 2021, Proceedings 27. Springer, 2021, pp. 3–18.
48. Belfadel, A.; Laval, J.; Bonner Cherifi, C.; Moalla, N. Requirements engineering and enterprise architecture-based software discovery and reuse. *Innovations in Systems and Software Engineering* **2022**, pp. 1–22.
49. Scheer, A.W. *ARIS—business process modeling*; Springer Science & Business Media, 2000.
50. Process Mining Manifesto. <https://www.tf-pm.org/upload/1580738212409.pdf>, 2012. Accessed at 2021-01-22.
51. Berti, A.; van Zelst, S.; Schuster, D. PM4Py: a process mining library for Python. *Software Impacts* **2023**, *17*, 100556.
52. van der Aalst, W.; Weijters, T.; Maruster, L. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* **2004**, *16*, 1128–1142. doi:10.1109/tkde.2004.47.
53. Van Der Aalst, W.; Adriansyah, A.; De Medeiros, A.K.A.; Arcieri, F.; Baier, T.; Blickle, T.; Bose, J.C.; Van Den Brand, P.; Brandtjen, R.; Buijs, J.; others. Process mining manifesto. Business Process Management Workshops: BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I 9. Springer, 2012, pp. 169–194.
54. van der Aalst, W. *Data Science in Action*; Springer Berlin Heidelberg: Berlin, Heidelberg, 2016; pp. 3–23. doi:10.1007/978-3-662-49851-4_1.
55. Rozinat, A.; Aalst, W. Conformance checking of processes based on monitoring real behavior. *Information Systems* **2008**, *33*, 64–95. doi:10.1016/j.is.2007.07.001.
56. Badhiye, S.S.; Chatur, P.; Wakode, B. Data logger system: A Survey. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)* **2011**, pp. 24–26.
57. Mitchell, R. *Web scraping with Python: Collecting more data from the modern web*; "O'Reilly Media, Inc.", 2018.
58. Khder, M.A. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. *International Journal of Advances in Soft Computing & Its Applications* **2021**, *13*.
59. Awad, A.; Weidlich, M.; Sakr, S. Process Mining over Unordered Event Streams. 2020 2nd International Conference on Process Mining (ICPM), 2020, pp. 81–88. doi:10.1109/icpm49681.2020.00022.
60. Van Der Aalst, W.M. A practitioner's guide to process mining: Limitations of the directly-follows graph, 2019.
61. Megargel, A.; Poskitt, C.M.; Shankararaman, V. Microservices Orchestration vs. Choreography: A Decision Framework. 2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC), 2021, pp. 134–141. doi:10.1109/EDOC52215.2021.00024.
62. Richards, M.; Ford, N. *Fundamentals of software architecture: an engineering approach*; O'Reilly Media, 2020.
63. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.M. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley Professional, 1994.
64. Kitchenham, B.; Pickard, L.; Pfleeger, S.L. Case studies for method and tool evaluation. *IEEE software* **1995**, *12*, 52–62.
65. Runeson, P.; Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* **2009**, *14*, 131–164.
66. Yao, Q.; Zhang, J.; Wang, H. Business process-oriented software architecture for supporting business process change. 2008 International Symposium on Electronic Commerce and Security. IEEE, 2008, pp. 690–694.
67. Jamshidi, P.; Pahl, C. Business process and software architecture model co-evolution patterns. 2012 4th International Workshop on Modeling in Software Engineering (MISE). IEEE, 2012, pp. 91–97.

68. Pourmirza, S.; Peters, S.; Dijkman, R.; Grefen, P. A systematic literature review on the architecture of business process management systems. *Information Systems* **2017**, *66*, 43–58.
69. Kiblawi, T.; Muhanna, M.; Qusef, A. The Role of Software Architecture in Business Model Transformability. 2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). IEEE, 2023, pp. 68–73.
70. Silva, S.; Tuyishime, A.; Santilli, T.; Pelliccione, P.; Iovino, L. Quality Metrics in Software Architecture. 2023 IEEE 20th International Conference on Software Architecture (ICSA). IEEE, 2023, pp. 58–69.
71. Diamantopoulos, T.; Saoulidis, N.; Symeonidis, A. Automated issue assignment using topic modelling on Jira issue tracking data. *IET Software* **2023**.
72. Mo, R.; Wei, S.; Feng, Q.; Li, Z. An exploratory study of bug prediction at the method level. *Information and software technology* **2022**, *144*, 106794.
73. Herraiz, I.; Robles, G.; González-Barahona, J.M.; Capiluppi, A.; Ramil, J.F. Comparison between SLOCs and number of files as size metrics for software evolution analysis. Conference on Software Maintenance and Reengineering (CSMR'06). IEEE, 2006, pp. 8–pp.
74. Bhatia, S.; Malhotra, J. A survey on impact of lines of code on software complexity. 2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014). IEEE, 2014, pp. 1–4.
75. Wang, T.; Li, B. Analyzing software architecture evolvability based on multiple architectural attributes measurements. 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS). IEEE, 2019, pp. 204–215.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.