universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Lab Automation: A Process-based Approach for Data Collection and Conformance Checking of Testing Settings

verfasst von | submitted by

Jan André Greschner, BSc.

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | degree programme code as it appears on the student record sheet:	UA 066 926
Studienrichtung lt. Studienblatt | degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Math. oec. Dr. Stefanie Rinderle-Ma

Acknowledgements

This master's thesis took forever! Not to suggest that it aged like fine wine, but it certainly fermented long enough to become something ... intoxicating. A journey longer than Frodo's trip to Mordor, and possibly just as perilous. However, it gave me a chance to learn and grow with challenges both as a person and a programmer. Some of my insights are printed on the following pages but most of them are in between them. Computer science is less about machines and more about the infinite expressions of human creativity facilitated by them. It is the art of solving complex problems and the poetry of logical ideas. I want to thank my supervisor Stefanie Rinderle-Ma for giving me the chance to work on an interesting and utmost challenging topic. A huge shoutout to Mr. Mangler, whose ability to send messages at ungodly hours taught me that sleep is just a social construct. His patience and commitment were invaluable in helping me refine my ideas and methodologies. Special thanks are due to my family for their unwavering support and for not asking too many times when I'd be done. This thesis not only represents my academic endeavor but also the collective effort and spirit of all those who supported me along the way. Thank you for not giving up on me or this thesis, which, against all odds, is now done. Phew!

Abstract

The global healthcare system faced unprecedented challenges in response to the COVID-19 pandemic, particularly in the domain of diagnostic testing. The rapid escalation in testing demand highlighted significant inefficiencies and vulnerabilities in laboratory operations, necessitating innovative solutions to enhance testing accuracy, efficiency, and compliance. This thesis explores the integration of process-aware information systems for existing laboratory environments (brownfield), specifically aiming to improve process efficiency, reduce errors, and enhance data traceability with a focus on the reverse transcription polymerase chain reaction (RT-PCR) testing process for SARS-CoV-2.

This thesis employs the design science methodology, focusing on the creation and iterative refinement of process-based artifacts. The thesis develops and implements a process-aware solution to integrate and monitor real-world processes within a medical diagnostic laboratory in Vienna, ensuring the processes adhere to predefined compliance and quality standards.

Key findings from this work demonstrate significant improvements in process efficiency, reliability, and compliance. The integration of the provided solution with the cloud process execution engine (CPEE) and existing laboratory information systems facilitates real-time data capture, streamlines workflow management, and enables systematic and automatic compliance checking against an established rule base. The solution not only reduces manual errors but also enhances real-time data visibility and provides insights to domain experts, facilitating improved decision-making and operational efficiency throughout the testing process.

The results of this thesis suggest that the integration of process-aware information systems can significantly enhance operational efficiency and compliance in laboratory settings, extending beyond pandemic response to other areas of healthcare diagnostics. The findings contribute to the existing body of knowledge in healthcare systems by showcasing the practical applications of process-aware information systems in high-stakes environments. Thus, this thesis supports improved health infrastructure and pandemic preparedness by addressing operational challenges.

Kurzfassung

Als Reaktion auf die COVID-19-Pandemie stand das globale Gesundheitssystem vor beispiellosen Herausforderungen, insbesondere im Bereich der labordiagnostischen Tests. Der rasante Anstieg in Bezug auf den COVID-19-Testbedarf machte erhebliche Ineffizienzen und Schwachstellen im Laborbetrieb deutlich, wodurch innovative Lösungen zur Verbesserung der Testgenauigkeit, Effizienz und Konformität erforderlich wurden. Diese Masterarbeit untersucht die Integration eines prozessorientierten Systems in bestehende Laborumgebungen, mit dem Ziel, die Prozesseffizienz zu erhöhen, die Fehler zu reduzieren sowie die Datennachverfolgbarkeit zu verbessern. Ein besonderer Fokus liegt auf dem Reverse-Transkriptase-Polymerase-Kettenreaktion (RT-PCR)-Testprozess für SARS-CoV-2.

Als Methodik dient der Design-Science-Ansatz, der sich auf die Schaffung und iterative Verfeinerung von prozessbasierten Artefakten konzentriert. Dies umfasst die Entwicklung und Implementierung einer prozessorientierten Lösung zur Integration und Überwachung von realen Abläufen in einem medizinischen Diagnostiklabor in Wien, um sicherzustellen, dass diese Prozesse vordefinierten Konformitäts- und Qualitätsstandards entsprechen.

Ergebnisse dieser Arbeit zeigen signifikante Verbesserungen bezüglich der Prozesseffizienz, Zuverlässigkeit und Konformität. Die Integration der bereitgestellten Lösung mit der Cloud-Processing-Engine (CPEE) sowie den bestehenden Laborinformationssystemen erleichterte die Datenerfassung in Echtzeit, optimierte die Verwaltung der Workflows und ermöglichte eine systematische Konformitätsüberprüfung von zuvor festgelegten Regeln. Der präsentierte Ansatz sorgte nicht nur für eine Verringerung von manuellen Fehlern, sondern auch für eine verbesserte Sichtbarkeit von Echtzeitdaten und bietet Fachleuten Einblicke, die eine bessere Entscheidungsfindung und betriebliche Effizienz während der gesamten Testdauer ermöglichen.

Die Untersuchungsergebnisse deuten darauf hin, dass die Integration prozessorientierter Systeme die betriebliche Effizienz und die Einhaltung von Vorschriften in Laborumgebungen erheblich verbessern kann, was über die Pandemie hinaus auch für andere Bereiche der Gesundheitsdiagnostik gelten kann. Die Ergebnisse tragen zum bestehenden Wissensstand im Gesundheitswesen bei, indem praktische Anwendungen prozessorientierter Systeme in hochsensiblen Umgebungen aufgezeigt werden. Somit wird durch diese Masterarbeit ein Beitrag zur Verbesserung der Gesundheitsinfrastruktur sowie der Pandemieversorgung durch die Bewältigung operativer Herausforderungen geleistet.

Contents

List of Tables

List of Figures

Listings

1 Introduction

1.1 Motivation and Scenario

Since its emergence in late 2019, the COVID-19 pandemic has profoundly impacted global health systems, economies, and societies. The rapid spread of the virus caused an unprecedented public health crisis, necessitating immediate and large-scale responses from governments and healthcare organizations worldwide [2]. The pandemic has not only strained healthcare resources but has also highlighted the critical role of timely and accurate diagnostic testing in managing infectious diseases. The surge in SARS-CoV-2 cases globally resulted in a dramatic increase in the demand for reliable testing methods, primarily reverse transcription polymerase chain reaction (RT-PCR) and antigen tests, to identify infected individuals and curb the transmission of the virus. This sudden and overwhelming need for widespread testing posed significant challenges, pushing the capabilities of existing diagnostic infrastructures to their limits, and revealing vulnerabilities in public health preparedness and response strategies [3, 4].

The rapid and accurate diagnosis of SARS-CoV-2 infections emerged as a cornerstone strategy in addressing the public health crisis caused by the COVID-19 pandemic. This strategy underscores the role of diagnostic testing as a primary tool in epidemiological surveillance and control. Efficient and widespread testing facilitates the rapid identification of infected individuals, enabling timely intervention measures such as contact tracing and the imposition of quarantine protocols. These measures are instrumental in disrupting transmission chains, thereby mitigating the spread of the virus [5].

The vast growth in demand for these diagnostic tests was both sudden and steep. Quantitative analyses [6, 7] have revealed that the requisition for PCR and antigen tests increased exponentially, exceeding the pre-pandemic levels by a substantial margin. This dramatic escalation not only underscores the critical role of these diagnostic tools in pandemic management but also highlights the pressure on healthcare systems worldwide. Moreover, the surge in testing requirements has illuminated several systemic vulnerabilities, most notably in the supply chain for critical testing components and the operational capacity of diagnostic laboratories.

These bottlenecks are exhibited in various forms. On the supply side, there was a notable scarcity of essential testing components, such as reagents and synthesized primers [8]. This shortage was compounded by logistical disruptions, which further impeded the distribution and availability of these critical materials. Concurrently, diagnostic laboratories, which were accustomed to handling a manageable volume of tests under normal circumstances, found themselves overwhelmed [9, 10].

The American Association for Clinical Chemistry (AACC) [6] conducted eight surveys over 32 months, targeting 133 laboratories in the US and 58 laboratories from 37 other countries and focusing on COVID-19 testing, supplies, staffing, and lessons learned. The findings reveal that

over 70% of laboratories had implemented COVID-19 diagnostic testing by May 2020. There was a significant peak in daily testing volumes in January 2022, with a median of 775 tests conducted per day. Furthermore, the surveys identify critical challenges faced by laboratories, particularly in supply chain management and staffing. To illustrate, 55% to 65% of laboratories reported difficulties in procuring essential supplies, with specific emphasis on the scarcity of reagents and test kits. Additionally, staffing shortages posed a significant hurdle to maintaining consistent testing operations which increased over time from 35.4% in May 2020 to 87.5% in January 2022.

This situation underscored the need for a scalable and resilient testing infrastructure capable of responding to public health emergencies. Recognizing these challenges, healthcare systems and diagnostic laboratories worldwide initiated efforts to address these impediments. Strategies to augment testing capabilities included expanding laboratory capacities, streamlining supply chains, and exploring technological innovations to enhance testing efficiency, such as automation strategies [11]. These measures aimed not only to manage the immediate crisis but also to build a more robust diagnostic framework capable of withstanding future public health challenges.

In this thesis, I partnered with a dedicated medical diagnostic laboratory located in Vienna for in-depth investigations to examine the reasons for these challenges. The laboratory was one of the key players during the COVID-19 pandemic, serving testing capabilities for multiple governmental institutions, the aviation industry, and sports institutions. The extensive testing capabilities required by the laboratory render it an exemplary subject for thorough examination. Before the pandemic, the laboratory efficiently managed a limited array of tests through manual processes. However, the sudden spike in testing demand rapidly exceeded the laboratory's manual capacities, resulting in extended waiting times for test results. The laboratory utilized two main strategies to address these challenges. The first strategy involved scaling up personnel and material resources, such as installing additional PCR testing machines. As already discussed above, this approach offers limited scalability due to supply chain and personnel issues. The second, more sustainable approach focused on enhancing the testing process through laboratory automation.

First, the laboratory acquired pipetting robots to automatically transfer liquids into different containers, also known as wells. The utilization of pipetting robots is associated with a multitude of advantages. These robotic systems enhance the precision and accuracy of liquid manipulation, thereby mitigating the potential for human-induced errors. The automation aspect facilitates high-throughput processing, which is crucial for the expeditious and efficient handling of substantial sample volumes. Moreover, these machines contribute to the standardization of experimental outcomes by ensuring uniformity in results. Additionally, by diminishing the necessity for repetitive manual tasks, these robots significantly improve the risk of strain injuries among laboratory personnel, thereby augmenting overall workflow efficiency and productivity.

In the second step, a laboratory information system (LIS) for processing the tested samples was developed, which also includes dedicated hardware. Utilizing a LIS in the context of COVID-19 samples offers numerous advantages, such as a centralized repository with all sample data in one place and streamlined communication between laboratories and healthcare providers. This system also supports the management of large volumes of data, which is essential for handling the high demand for COVID-19 testing. Moreover, the LIS offers interfaces for importing and exporting data such as test results and reports.

The laboratory's LIS currently exhibits deficiencies in the areas of traceability and verification

of test samples. While it provides a high-level status overview for each sample, it cannot break this status down into distinct processing stages. Consequently, it is not fully possible to obtain a detailed view of a sample's status throughout the testing sequence. Furthermore, the testing process has only been partly validated against established business rules. A critical example of this is the requirement for samples to be analyzed within a specific timeframe before the reagents expire.

In light of the heightened demand for COVID-19 PCR testing, the laboratory concurrently prepared multiple well plates (i.e., samples were pipetted onto a so-called well plate, which allowed for the simultaneous processing of multiple samples under identical conditions), resulting in a bottleneck due to the limited availability of only two PCR testing machines. This bottleneck led to a significant backlog of unprocessed well plates. There were instances where well plates were rendered unusable due to the lapse of this critical timeframe, necessitating the repetition of the entire testing procedure, including sample extraction. Such setbacks not only incur additional costs but also demand extra resources. Another example is the export process. The sample result cannot be exported if the corresponding patient data is missing or incomplete in the LIS. Notably, this issue manifests only after the initiation of the export process, resulting in delays in communicating the results.

To address this challenge, an implementation of a process-based tracking system was proposed. In this tracking system, well plates will be tagged with barcodes or QR codes. Laboratory personnel will manually scan these plates at each stage of the process or the scanning will be automated using cameras. This action will activate a service that communicates with a process execution engine, such as the cloud process execution engine (CPEE[1], where the testing process will be explicitly modeled. The process will commence with the generation of a machine-readable code on each new well plate, which will trigger the CPEE to spawn new process instances that can operate concurrently.

As well plates and samples are scanned, their data will be transmitted to the CPEE, enabling the real-time monitoring of each process instance's status. This system will provide a centralized means to track the entire process lifecycle of all active well plates, with process instances displayed on a central dashboard. An additional feature will include business rule enforcement, such as alerts sent when a well plate remains in a specific task beyond a predetermined duration, thus preventing the oversight of well plates. Moreover, results from the PCR testing machines can also be integrated into the CPEE, allowing for specialized responses to specific result statuses. A potential challenge might be the additional task for laboratory staff to scan each well plate at every step, which could lead to unintentional omissions. Such scenarios will require further problem-solving capabilities. Furthermore, the laboratory staff may need training in the business process model and notation (BPMN) language to fully understand the dashboard's visualization and the process itself.

As depicted above, laboratory automation is one approach to handling such a challenge. In an automated process, samples can be directed to the corresponding work steps, which not only ensures a fast laboratory workflow but also enables comprehensive quality assurance through the continuous monitoring of the entire process.

[1] **https://cpee.org/**), last accessed: 2024-05-25

1.2 Research Questions

In the dynamic realm of industry-specific operations, the influence of process-aware information systems on enhancing process efficiency, flexibility, and compliance is undeniable [12]. Such systems, which intelligently integrate technology with business processes, have been pivotal in transforming operational landscapes across various sectors, including healthcare. The escalating demands on healthcare systems, especially as evidenced during the COVID-19 pandemic, underscore the critical role of efficient, flexible, and compliant processes. However, the implementation of process-aware information systems may vary between healthcare and non-healthcare industries as each industry has its own unique challenges and operational dynamics. This potential disparity raises the first research question:

RQ1: How does the implementation of process-aware information systems compare across healthcare and non-healthcare industries in terms of enhancing process efficiency, flexibility, and compliance?

In the rapidly evolving field of medical diagnostics, laboratory workflows are crucial intersections where technology meets biological science. The need for the efficient, accurate, and swift processing of data is more pressing than ever, especially as global health challenges stretch the capabilities of existing systems to their limits. Integrating process-aware information systems into these workflows promises significant enhancements in real-time data capture, log management, and analytical insights, which are essential for improving decision-making and operational efficiency. Such systems automate routine tasks, enhance precision, and provide actionable insights at each diagnostic step, thus transforming traditional laboratory environments. This leads to the second research question:

RQ2: How can a process-aware system be integrated into existing laboratory workflows to offer real-time data capture, log management, and analytical insights, thereby improving decision-making and operational efficiency?

PCR tests have become the gold standard for diagnosing a variety of infectious diseases due to their accuracy and reliability [13, 14]. However, the surge in demand for these tests has strained laboratory capacities, revealing significant challenges in error reduction, traceability, and overall process monitoring. Laboratories face critical issues, such as delays in test processing, errors in sample handling, and challenges in traceability, all of which can have severe implications on patient care and public health management. To address these issues, a transition toward process-based conformance checking within diagnostic laboratories presents a compelling case for enhancing operational efficiency and test accuracy. This method offers a systematic approach to monitoring and verifying the adherence of laboratory workflows to established compliance and quality standards. Thus, the third research question emerges:

RQ3: What are the impacts of employing process-based conformance checking on the error reduction, traceability, and overall monitoring of PCR testing processes within diagnostic

laboratories?

1.3 Contribution

This research explores the integration and evaluation of a process-aware system within laboratory environments, focusing on enhancing efficiency and compliance in SARS-CoV-2 testing processes. A theoretical advancement provided by this work is the application of business process management (BPM) together with the CPEE in a healthcare setting. This work extends the existing knowledge on process integration and is particularly valuable in high-stakes environments such as those that respond to pandemics. The methodological framework developed for this thesis has been robustly applied to evaluate the effectiveness of these systems in a real-world laboratory setting, validating the theoretical models and serving as a replicable model for future endeavors in healthcare systems and laboratories. Furthermore, this research highlights the critical role of scalable and adaptable process-aware information systems in managing healthcare crises, highlighting their potential to significantly influence policymaking and strategic planning in healthcare infrastructure. This is especially relevant in enhancing response capabilities during pandemics. The findings from this study not only aid in managing the current pandemic but also enhance preparedness for future healthcare challenges by providing a foundational model for integrating advanced process-aware information systems in laboratory and healthcare operations. By improving the efficiency, flexibility, and compliance of laboratory operations, this thesis offers valuable insights that have implications for both theory and practice, potentially guiding future innovations in laboratory automation and medical informatics. The artifacts resulting from this work are publicly available on GitHub[2]. The collected process logs generated through the CPEE can be accessed on Zenodo[3].

1.4 Methodology

This thesis adopts design science, a methodology rooted in the creation of artifacts, to answer relevant questions addressing human problems. Design science, characterized by its practical orientation, merges theoretical analysis with the construction of artifacts (e.g., software components), offering tangible solutions and insights into complex problems. It is based on a staged approach, which involves the following steps for creating artifacts [15]:

1. **Identification of the Problem:** The starting point is a comprehensive identification of the problem. To improve practicability, the problem may be atomized, which ensures a nuanced understanding. This granularity not only aids in defining the scope of the research but also facilitates the creation of more targeted and effective solutions.

2. **Definition of Objectives:** Objectives are systematically inferred from the problem statement. This process involves a thorough analysis to establish both qualitative and quanti-

[2] `https://github.com/greschner/cpee-worklist`, last accessed: 2024-06-12
[3] `https://doi.org/10.5281/zenodo.11617408`, last accessed: 2024-06-12

tative goals, ensuring alignment with the broader ambitions of the field and the specific needs of relevant stakeholders.

3. **Artifact Design and Implementation:** In this step, the artifact is designed following a structured process. The potential outcomes can take various types of formats, namely methods, instantiations, and models. This phase is governed by theoretical models and practical considerations to ensure the artifact's relevance and effectiveness.
4. **Demonstration of Artifacts:** The demonstration phase employs various methods such as experimental setups, proof-of-concept models, and case studies. The choice of method is based on the nature of the artifact and the specific objectives of the study, ensuring an appropriate and effective demonstration.
5. **Evaluation:** The evaluation process involves comparing the results against the set objectives. Methods range from quantifiable measures to satisfaction surveys, and each method is selected based on its suitability to measure the artifact's effectiveness. Feedback from this phase informs whether to improve the effectiveness of the artifact by returning to the third step or to continue with the next step.
6. **Communication:** Findings are communicated through academic publications, presentations, and stakeholder reports, ensuring a broad distribution of knowledge and insights. The communication strategy is tailored to reach both the academic community and the involved stakeholders effectively.

The fundamental principle is ensuring that the knowledge and understanding of a design problem and its solution are acquired during the construction and application of an artifact [16]. The approach of iteratively relaxing idealized conditions is utilized to increase the relevance of such artifacts. This approach pursues the goal of a viable solution for the real world [17] and is similar to when new medicines are assigned to practice. First, new drugs are tested in the laboratory in idealized conditions; then, the medications are gradually tested on patients until they are deemed safe enough to be launched in the market.

In the context of a medical diagnostic laboratory, the primary stakeholders are identified as follows:

- **Employees:** These individuals are the laboratory personnel who predominantly engage with and operate the artifacts.
- **Managers:** These individuals are responsible for allocating necessary resources and specifying the expected outcomes.
- **Customers (including government entities):** These individuals are the direct beneficiaries of the artifacts who potentially influence or even establish their specific requirements.

The discrepancy between the actual results and the anticipated outcomes is leveraged to define what is referred to as an "improvement problem". Addressing this problem necessitates the allocation of budget, time, and other relevant resources by the stakeholders. The improvement problems encompass the planning, creation, and evaluation of artifacts. Upon identifying these problems, they are addressed by iteratively applying the stages explained above [17].

1.5 Aim of This Work

This work primarily aims to address challenges in data collection and conformance checking in the domain of medical diagnostic laboratories, with a particular focus on the PCR testing process. The objective is to bridge gaps in current methodologies by introducing and critically evaluating compliance-checking tools and developing advanced automation strategies.

The research problem centers on inefficiencies, procedural discrepancies, and transparency deficiencies in the existing PCR testing process, which is crucial for accurate medical diagnostics. Adopting the previously introduced design science approach, this work focuses on creating and evaluating artifacts designed to address the identified challenges in data collection and conformance checking in the medical diagnostic laboratory domain. This methodology encompasses the iterative development of prototypes, empirical data analysis, and simulations tailored to the context of PCR testing processes. The approach facilitates a comprehensive understanding of both the theoretical constructs underpinning the process improvements and their practical applications in laboratory settings.

A significant aspect of the research involves exploring the integration of state-of-the-art digital tools in laboratory workflows. These tools are critically assessed for their ability to streamline workflows, reduce manual errors, and enhance process reliability. Central to this research is the process-based approach, which transforms implicit processes into explicit, structured workflows. This approach involves systematically deconstructing and refining the PCR testing process and enhancing its efficiency, manageability, and adaptability. The following theoretical underpinnings of this approach form a crucial part of the analysis:

- **Scalability:** The work aims to redefine scalability in terms of processing capacity, responsiveness to demand shifts, and adaptability to evolving PCR testing technologies.
- **Reduction in Errors:** Through a systematic comparative analysis of current versus proposed processes, the work categorizes error types and aims to mitigate them through real-time monitoring and feedback mechanisms.
- **Enhanced Traceability:** The introduction of advanced tracking technologies intends to elevate process transparency in PCR testing, facilitating detailed tracking and logging of each sample throughout its lifecycle.
- **Comprehensive Monitoring:** The thesis envisions the development of an integrated monitoring system that is capable of assimilating and analyzing multiple data streams in real-time to enhance process oversight and decision-making.

The expected impact of this work includes improved process efficiency and accuracy in PCR testing, contributing to the reliability of medical diagnostics. The research aims to offer a scalable and adaptable framework applicable to other diagnostic tests and laboratory settings. As this work primarily focuses on COVID-19 PCR tests, it acknowledges the limitations in generalizability and application across varied laboratory environments.

1.6 Structure of This Work

This work is structured as follows: Section 3 presents a comprehensive review of related work in the context of laboratory automation, process conformance, and technology integration in diagnostic settings. Building on this foundation, Section 4 delves into the current PCR testing process, examining the isolation of SARS-CoV-2 RNA and detailing the involved phases and methodologies. The solution design is introduced based on Section 4, and it details the requirements and architectural considerations and provides an overview of the proposed solution. Section 6 then explores the practical application and discusses the implementation aspects of the solution, highlighting the deployment and operationalization of the system. Section 7 evaluates the solution through quality-based key performance indicators (KPIs), assessing its effectiveness and efficiency in a real-world scenario. The resulting findings are discussed in Section 8, focusing on integration challenges, digitalization prospects, and potential areas for further research. Finally, the thesis concludes with Section 9, which summarizes the research contributions, addresses the research questions, underscores the importance of the findings, and discusses the implications.

2 Fundamentals

2.1 Laboratory Automation

Despite the lack of a universally accepted definition, laboratory automation typically falls into three categories [18]:

- **No automation:** Complete reliance on manual tasks executed solely by humans.
- **Task-targeted automation (TTA):** Automation of specific tasks and machine integration, with manual elements such as sample transportation (i.e., partial automation) [19, 20]. Example: Transferring samples from one well plate onto a different well plate by using an automated pipetting robot.
- **Total laboratory automation (TLA):** A highly interconnected and integrated system minimizing manual intervention, often extended to pre- and post-analytical processes (i.e., full automation). Example: Systems that automatically handle, analyze, and store samples and integrate data into laboratory information systems, such as Becton Dickinson's (BD) Kiestra TLA or Copan's WASPLab [21].

The assessment of proposed automation solutions in laboratory settings necessitates a thorough examination of the interfaces and communication protocols among the laboratory information system (LIS), the laboratory automation system's (LAS) computer, and the analyzers. This evaluation aims to ensure optimal process control capabilities, including the ability to conduct repeat testing, perform tests on sample dilution, and enable reflexive testing, among others, without necessitating manual intervention. The seamless coordination between the LIS and the LAS is especially important to accurately verify test results, thereby facilitating the routing of samples to subsequent processing units such as recappers and archival units. Hawker et al. [22] have stated that it is noteworthy that the effectiveness of communication protocols in facilitating process control functionalities may vary depending on the compatibility between the LAS and the analyzers, particularly when obtained from different vendors. Despite the potential capability of the LAS to interact physically with analyzers from diverse sources, achieving optimal communication and coordination may be more streamlined when utilizing analyzers from the same vendor.

Hawker et al. have further stressed the significance of process control as a fundamental component within laboratory automation and defined several of its key functionalities. The process control software must be able to scan and interpret the identification bar code of samples and retrieve relevant data from the LIS, such as the sample's nature and the tests prescribed. Subsequently, the software should determine the necessary workflows required by each sample, such as centrifugation, decapping, or aliquoting, and delineate the precise sequence of actions for each sample accordingly. Arbruster et al. [23] have further emphasized, that software should

also be equipped to monitor the performance for optimal operational status and autonomously address situations where tests are unavailable. Automatic sample integrity checks should be implemented, employing rules-based decision-making to evaluate sample quality and enforce appropriate actions.

The Clinical and Laboratory Standards Institute (CLSI) communication standard emerged through a collaborative effort with Health Level Seven (HL7) [24, 25]. To address unmet requirements in laboratory automation communications, a novel section, Chapter 13, was introduced in the HL7 standard, specifically in HL7 Version 2.4 and its subsequent iterations [26]. This development aimed to rectify deficiencies in the earlier American Society for Testing and Materials (ASTM) standards that CLSI had previously adopted [27, 28]. Manufacturers and users of laboratory automation systems are strongly advised to use the most recent HL7 and CLSI standards to encourage interoperability between lab components. Notably, CLSI periodically conducts relevance assessments of all its standards and guidelines and, subsequently, revises these standards when deemed necessary [22].

TLA is renowned for its capacity to yield long-term financial efficiencies through the consolidation of diagnostic platforms, leading to significant reductions in manual labor and associated pre-analytical and post-analytical costs [18]. This system enhances operational efficiency within the laboratory, streamlining workflows by minimizing the necessity for personnel to navigate across various analyzers. Moreover, the integration of information technology with TLA systems significantly enhances sample management and traceability, thereby elevating the overall quality and safety of diagnostic procedures. TLA contributes to the standardization and facilitation of certification and accreditation processes within laboratories. Furthermore, the automation inherent in TLA mitigates the incidence of manual errors, thereby enhancing the integrity and reliability of testing. The system's design allows for the use of reduced sample volumes, decreasing patient discomfort and environmental impact due to biological waste. TLA also enables a more cohesive integration of test results, strengthening data management and the identification of errors. Furthermore, it reduces occupational exposure to biological hazards.

However, the implementation of TLA is associated with substantial initial financial costs due to necessary infrastructure modifications and the acquisition of specialized hardware, as discussed previously [18, 21]. Operational costs (OPEX), encompassing heightened maintenance and supply expenses, are also increased. Spatial constraints within existing laboratory environments pose significant challenges to the integration of TLA systems; consequently, TLA is mostly applied by large-scale laboratories. The concentration of multiple pieces of equipment within a confined area can lead to overcrowded working conditions and increased levels of noise and heat. The potential for system failures poses a significant risk to laboratory operations, and the reliance on automation can result in diminished proficiency in manual skills among staff. The need to cater to diverse sample types necessitates specific handling protocols, complicating workflow management, particularly in balancing urgent testing with routine analyses. For end users, the most significant difficulty lies in preventing and responding to system errors. These errors often originate from human actions, such as mishandling plates or misplacing digital files, but can also stem from unpredictable factors, such as incompatible equipment, power issues, or network problems. These errors not only hinder work completion but also require additional staff involvement in troubleshooting them. An overreliance on a single vendor may limit laboratory autonomy, which

could lead to a scenario where the laboratory's functioning is primarily driven by the equipment manufacturer.

In conclusion, while TLA represents a significant advancement in the efficiency, quality, and safety of laboratory testing, it also introduces a series of challenges, including financial, operational, and human resource implications. TLA can, therefore, not be interpreted as a panacea for laboratory automation.

2.2 Business Process Management

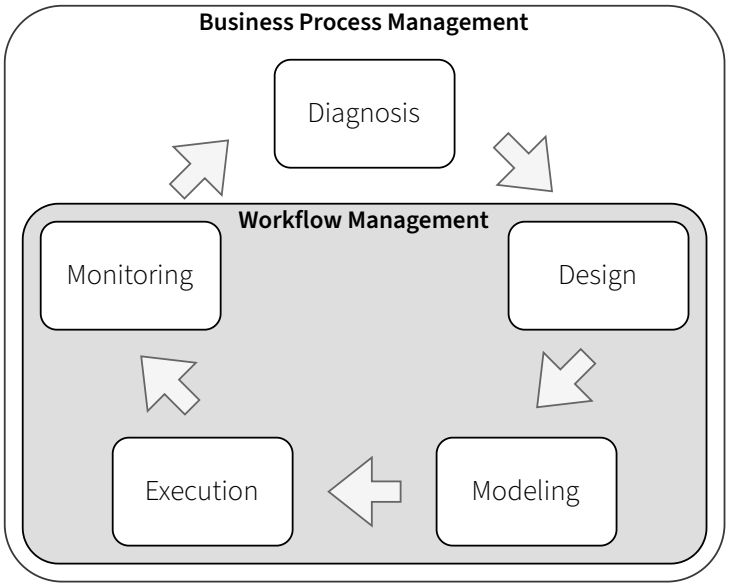


Figure 2.1: BPM life cycle and its relation to workflow management

Business process management (BPM) is a comprehensive methodology aimed at enhancing organizational workflows through efficacy, efficiency, and adaptability to dynamic environments [29]. This approach encompasses five distinct phases, namely the design, modeling, execution, monitoring, and diagnosis of business processes (see Figure 2.1).

The BPM lifecycle begins with the design phase, where the current processes and their envisioned improvements are conceptualized. This conceptual phase transitions into the modeling phase, where the designs are represented through a process model. Subsequently, these processes are implemented in the execution phase, followed by a rigorous phase of monitoring to track performance metrics and identify potential bottlenecks or inefficiencies. The insights gathered from the monitoring phase are directed to the diagnosis phase, where optimization efforts are identified and strategic adjustments and refinements are made to enhance the overall process performance.

BPM aims to synchronize processes with the strategic objectives of the organization, focusing on both the administrative and operational scopes of these processes. It advocates for a holistic

management paradigm that seeks to foster organizational efficiency and effectiveness while simultaneously encouraging innovation, adaptability, and technological integration [30]. Business processes are represented in the modeling phase through process models by utilizing modeling languages such as Business Process Modeling Notation (BPMN), event-driven process chains (EPCs), Petri nets, and Unified Modeling Language (UML). BPM is facilitated by a range of BPM systems designed to enhance collaboration, coordination, and decision-making in the realm of business processes [31, 32]. In this landscape of BPM systems, distinctions arise in the manner in which tasks are sequenced and coordinated. One notable category in BPM systems is workflow management systems (WfMSs). WfMS can be defined as follows:

> "A system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications." [33]

Figure 2.1 illustrates the linkage between business process management (BPM) and conventional workflow management systems (WfMSs) in the context of their respective roles in the BPM life cycle, which is based on the work of Mans and van der Aalst [34, 35]. Moreover, the figure explicates that WfMSs demonstrate a robust level of support for the modeling and execution phases of BPM. However, their support is comparatively limited or even rudimentary for the design and control phases and completely absent for the diagnosis phase. Therefore, BPM extends the behavior of workflow management (systems). Within the design phase, WfMSs support the development of process models, which define the sequence of task execution, assign responsibilities to individuals, and outline available information resources. However, they typically lack comprehensive support for process redesign, such as refinements and optimizations, as there is no support for the diagnosis phase. During the monitoring phase, WfMSs generally offer aggregated data about active process instances, such as their current execution status.

In the context of rapidly advancing digital transformation, business processes are increasingly governed and monitored by information systems. These sequences of activities (i.e., tasks) are described through process models. These models are instantiated and executed at runtime, often facilitated by advanced tools such as a process execution engine (e.g., CPEE) [36, 37]. Throughout the execution phase, information on the process instance is recorded in event logs, which serve as a comprehensive reflection of process behavior. Predominantly, these logs are maintained in either the eXtensible Event Stream (XES) or Mining eXtensible Markup Language (MXML) formats [38, 39]. The XES standard, an evolution of the MXML format initiated in 2003, adheres to the XML schema. It structures data into logs containing traces, each comprising a sequence of events. XES includes multiple layers of attributes, such as timestamps and lifecycle stages, with the possibility of nested attributes. These detailed event logs are instrumental in extracting process-driven insights, such as those utilized in process mining.

Three principal methodologies of process mining are described in [40], focusing on *discovery*, *conformance*, and *enhancement* which are illustrated in Figure 2.2. The *Discovery* technique involves generating a process model from existing event logs using algorithms such as the alpha algorithm. *Conformance* compares the existing process with event logs to identify deviations. *Enhancement* involves modifying or expanding the current process model using event logs as a

foundational input, aimed at refining the model. Unlike *Conformance*, which seeks to align the model with reality, *Enhancement* actively evolves the model. Certain process mining techniques enable real-time analysis, including conformance checking during the actual process execution or predictive timing based on historical data. Process mining extends beyond the mere assessment of control flow metrics. It encompasses various dimensions, including time, case, and organizational perspectives. These dimensions are not regarded as entirely distinct; they may intersect. An example is the discovery of a social network model, which demonstrates the interconnectivity between different aspects of process mining [41]. Collectively, this section provides a comprehensive overview of the broad spectrum of process mining and its techniques, highlighting their significance as essential tools for process enhancement [42].

Compliance checking [43, 44, 45, 46] refers to the systematic assessment and verification approach to ensure that the processes under review adhere to predefined constraints (i.e., business rules). The requirement for such constraints arises from various origins, including legislative authorizations, regulatory frameworks, and prescriptive directives, which are frequently presented in textual formats. A significant aspect in the realm of compliance checking involves the transition of these requirements into compliance objectives, followed by their formalization into compliance rules and constraints. The primary distinction between *conformance checking* and *compliance monitoring* lies in the model utilized for log analysis. As discussed previously, *conformance checking* aims to compare the execution log against a process model and is commonly conducted retrospectively. Nonetheless, despite these variances, there is considerable potential for synergy between these domains. Notably, many *conformance checking* methodologies can be easily adapted for real-time applications. There is also a third possibility called *verification*, which checks whether every conceivable execution of a specified process model adheres to a particular property.

There are two primary compliance-checking approaches. The first, known as *forward compliance checking* [43, 47, 48], ensures that processes adhere to compliance standards during design and execution. This approach involves either constructing processes with built-in compliance measures or verifying existing processes for compliance. Alternatively, compliance requirements can be translated into monitoring rules or model annotations, which are then utilized to enforce compliant process executions. Limiting the process to a predetermined sequence of tasks diminishes its capacity to adapt to dynamic environments, such as by accommodating the bypassing of specific process steps during periods of high capacity. Each modification necessitates updating the operational information system, potentially leading to significant adjustment expenses.

The second category, *backward compliance checking* [43, 47, 48], involves retrospectively assessing whether past process executions (i.e., post-mortem) complied with established compliance rules to identify instances of non-compliance and the specific circumstances surrounding their occurrence. Compared to forward compliance checking, this methodology offers the benefit of unrestricted agility and flexibility throughout the execution process.

The concepts of *effectiveness* and *efficiency* are often used in the context of processes. While they may seem similar, they have distinct meanings: Effectiveness is concerned with achieving the right result (doing the right things), while efficiency is related to the best use of resources to achieve that result (doing things right) [49]. Ideally, processes should aim to be both effective and efficient, ensuring that objectives are met in the most resourceful manner.

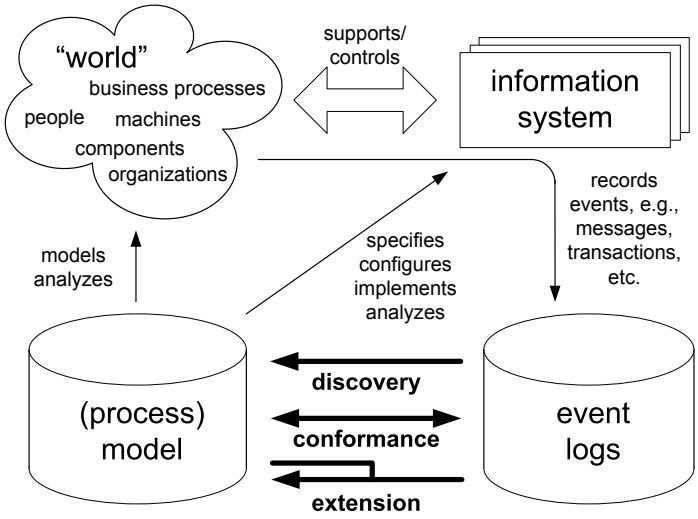


Figure 2.2: Process mining types, namely discovery, conformance, and extension. Source: [1]

2.3 Key Performance Indicators (KPIs)

In the context of laboratory automation, KPIs are quantifiable metrics employed to assess the effectiveness, operational efficiency, and result quality of automated systems and processes within a laboratory environment. These indicators are subject to variation based on the specific objectives and processes of the laboratory. However, they typically concentrate on the following core areas:

Accuracy and Precision of Results

Quantifies This indicator quantifies the degree to which automated systems yield correct results (accuracy) and their ability to produce consistent results under unchanged conditions (precision).

Throughput

This indicator measures the volume of samples processed within a specified time interval, with higher throughput denoting enhanced automation efficiency. Evaluating throughput involves not only counting the number of samples processed but also considering the complexity and diversity of the tests conducted. Moreover, throughput, as a KPI, serves as a benchmark for resource allocation and workflow optimization. By analyzing throughput data, laboratory managers can identify areas that need improvement and make changes, such as automating additional processes or upgrading existing equipment, to further enhance the laboratory's operational efficiency [49].

Turnaround Time (TAT)

The existing literature presents various definitions of turnaround time (TAT) in the clinical domain. TAT can be characterized, for instance, as the duration from the reception of a sample to the

availability of the test result, known as "laboratory TAT", by excluding the pre- and post-analytical phases such as sample collection and transport to the laboratory [50]. Alternatively, it may be defined as the period from the physician's request to the point at which the physician reviews the result, termed "total TAT" [51]. This multiplicity of definitions, often used interchangeably, poses a significant challenge in standardizing and comparing operational processes across the medical domain. Therefore, clarity and precision in the definition of TAT are crucial. A systematic review of the different TAT definitions in the medical domain can be found in the work of Breil et al. [52]. The TAT is often referred to as a key measure for laboratory performance since rapid TATs are essential for clinicians to facilitate the prompt diagnosis and treatment of patients [53, 54, 55]. Consequently, more efficient TATs are often a key objective of automation.

Cost Efficiency

This indicator evaluates the cost-effectiveness of automation in terms of operational costs versus labor and resource savings [56]. Measuring the cost efficiency involves a detailed analysis of the investment in automation technology against the economic benefits derived, such as reduced full-time equivalents (FTEs), decreased reagent usage, and minimized waste. This evaluation provides insights into the long-term financial viability and sustainability of the automated processes within in the laboratory environment.

Equipment Uptime

Equipment uptime is the amount of time the automated systems are operational and available for use without malfunction or maintenance. Evaluating equipment uptime involves monitoring the frequency, duration, and causes of system downtimes. Effective management of equipment uptime requires a proactive approach to maintenance and a robust understanding of the system's operational demands. Strategies to manage equipment uptime may include predictive maintenance, where potential issues are identified and addressed before they lead to system failure, and rapid-response protocol implementation for resolving unexpected equipment malfunctions [57, 58].

Error Rate

Error rate refers to the frequency of errors or failures in automated processes. Lower error rates indicate more reliable automation. The analysis of error rates involves a comprehensive evaluation of instances where automated processes deviate from expected outcomes. As depicted in [56] the error rate can include, but is not limited to, inaccuracies in test results, mechanical failures, software problems, and issues in sample handling. Furthermore, the error rate is not only a measure of the immediate reliability of the automated processes but also serves as an indicator of potential long-term operational risks. Regular monitoring and analysis of the error rate assist in preemptively identifying areas prone to failures or inaccuracies. This proactive approach facilitates timely interventions, such as system recalibrations, software updates, or equipment maintenance, thereby mitigating the risk of significant operational disruptions.

Scalability

Scalability is the ability of the automation system to handle an increased workload or adapt to new tests and processes [56]. In assessing scalability, it is important not only to consider the current needs of the laboratory but also to anticipate future demands. This foresight ensures that the investment in automation remains valuable over time, providing a sustainable solution that supports the laboratory's growth and evolving objectives.

Compliance and Quality Control

This indicator measures adherence to regulatory standards and internal quality control benchmarks, such as regular monitoring and verification of the accuracy and reliability of test results [59, 60, 61]. The compliance and quality control measure also includes effective documentation and traceability mechanisms, which are essential for audit trails and for investigating any discrepancies or anomalies in test results. This KPI also involves the implementation of a continuous improvement process for quality control and compliance measures. Regular reviews and updates to protocols and procedures are necessary in response to changes in regulations or technological advancements.

User Satisfaction

User satisfaction is the feedback from laboratory staff regarding the usability; reliability; and overall impact of the automation on their workflow, such as the simplicity of operating procedures, the level of training required to use the system effectively, the system's role in reducing manual tasks, and the system's effectiveness in accelerating processes [62]. It is essential to monitor and enhance this KPI to ensure that the automation system not only meets technical specifications but also aligns with the needs and preferences of laboratory personnel, thereby driving its successful adoption and utilization.

Data Management and Interoperability

Data management and interoperability refers to the effectiveness of the automated system in managing and integrating data within the laboratory's information systems (LIS). A critical aspect is the system's interoperability, or its ability to seamlessly interact with other software in the laboratory environment, facilitating smooth data exchange and compatibility [23]. Equally important is the ease of data access and retrieval, which focuses on the system's user-centric design and efficiency in providing necessary information. Another important point to consider is real-time data processing, that is, evaluating the system's capability to present information based on current data, which is essential for timely decision-making. Data security and regulatory compliance form a crucial component of this KPI, underscoring the importance of robust measures against unauthorized data breaches and adherence to prevailing regulatory frameworks, especially relevant in the context of handling sensitive or patient data. Lastly, the KPI also evaluates the system's analytical and reporting capabilities. This aspect examines the system's ability to interpret raw

data and generate comprehensive reports, transforming data into actionable insights that support decision-making and strategic planning in the laboratory domain.

Conclusion

Understanding the interrelationships among KPIs for laboratory automation requires a comprehensive approach, as these metrics are intricately related and often influence each other in various ways. The accuracy and precision of results in an automated laboratory setting are inherently tied to the error rate. High levels of accuracy and precision typically indicate a lower incidence of errors, demonstrating the reliability of the automated processes. Furthermore, throughput and turnaround time are directly correlated. An increase in throughput, or the number of samples processed within a given timeframe, usually results in shorter turnaround times, thereby enhancing the overall efficiency of the laboratory. Moreover, turnaround time is a critical factor influencing user satisfaction. Laboratories that achieve faster turnaround times often see higher levels of satisfaction among users, particularly in time-sensitive environments such as clinical diagnostics [63]. The user satisfaction is also affected by the efficiency of data management and integration. Systems that offer seamless and error-free integration of data typically receive more favorable responses from users, as they reduce manual intervention and potential errors. Greater equipment uptime suggests a more efficient use of resources, improving cost efficiency and directly impacting the turnaround time. The consistent availability and reliability of equipment allow samples to be processed without unnecessary delays, contributing to faster result delivery. This efficiency, however, must be balanced against the need for regular maintenance to prevent equipment failures. The error rate is closely linked to compliance and quality control. A lower error rate is often a result of stringent compliance with regulatory standards and robust quality control measures, underscoring the importance of these practices in maintaining the integrity of laboratory results. Scalability impacts both the throughput and turnaround time. An automation system that can scale effectively is capable of handling increased workloads without compromising speed or efficiency, which are vital factors for laboratories experiencing growing testing demands. The effectiveness of data management and integration plays a crucial role in ensuring the accuracy and precision of results. Proper management of data ensures the availability of accurate information for analysis, thereby influencing the reliability of the automated processes.

In conclusion, these KPIs provide a comprehensive framework for assessing and improving laboratory automation. They interconnect in various aspects; some focus more on quality (e.g., error rate, compliance, and data management) while others are oriented toward quantity (e.g., throughput, TAT, and scalability). Collectively, these KPIs serve as fundamental benchmarks for optimizing the effectiveness and efficiency of laboratory operations. Compared to this work, the objective of this thesis is to conduct a comprehensive assessment of the laboratory processes with a focus on evaluating quality metrics, specifically error rates and compliance controls.

3 Related Work

In the context of healthcare-related processes, various modeling languages have been developed such as GLIF, Asbru, EON, PROforma, GUIDE, and PRODIGY which are quite similar to traditional workflow languages [34]. BPMN is recognized as the leading standard in the realm of business process engineering. When it is directly compared to PROforma, BPMN effectively meets the demands of the healthcare sector [64, 65]. As a result, this work utilizes the BPMN 2.0 standard as the main modeling language.

WfMSs have been effectively used in industries with well-defined and stable processes, such as banking and insurance, where there are fewer unexpected behaviors [66]. In contrast, healthcare processes are more complex and variable, and they involve various medical disciplines. As a result, the application of WfMSs becomes more challenging in healthcare [67, 68, 69].

While WfMSs are not commonly adopted across the healthcare sector, there are some successful examples of their use [34, 70, 71, 72, 73, 74, 75]. However, these examples are usually confined to either rigid organizational processes or single-user medical treatments with guideline-based decision support. Furthermore, there is a lack of detailed analysis on how WfMSs are applied in healthcare and the outcomes of their implementation, especially in terms of quantitative evaluations of their impact on process efficiency, waiting times, and patient experiences before and after system implementations.

One study presented and evaluated paperless WfMS for radiology [70]. This system was designed to improve workflow in a filmless and speech-recognition environment by prioritizing cases, facilitating critical result communication, and documenting these processes. The study found significant improvements in reducing case turnaround times, decreasing interruptions, and modestly improving patient perceptions of service timeliness. However, there was no significant change in staff satisfaction levels. Compared to this work, this thesis mainly focuses on improving the quality of healthcare processes, not reducing the turnaround time (i.e., quantity).

Healthcare-related processes need a formal representation (i.e., process model) to be interpreted by a WfMS, as shown in a previous study [76]. However, due to the complexity and dynamics of patient treatment processes, WfMS faces a significant challenge in this regard. Patients require highly individualized treatment plans. Additionally, the involvement of diverse medical teams, each with different areas of expertise, complicates standardization. Moreover, the critical and unpredictable nature of intensive care necessitates flexible and adaptive workflows, which are often not possible with rigid standard systems. This dynamic environment makes it difficult to apply one-size-fits-all workflow solutions. Sedlmayr et. al [76] propose a methodology for formalizing and automating clinical guidelines to improve quality and efficiency in patient care. This process involves capturing medical processes, translating text-based guidelines into executable workflow formats, and integrating these formats into patient data management systems. The paper outlines a three-level representation of know-how in patient care, which highlights the system's role in specifying and executing medical processes on top of operational patient data

management systems.

The ADEPT project [34, 77, 78] has had a significant influence on the adaptability and impact on healthcare processes. ADEPT has successfully introduced novel flexibility mechanisms and implemented an advanced WfMS. With a primary focus on supporting healthcare processes effectively, the ADEPT project offers an advanced process management approach, that proposes a higher flexibility compared to contemporary WfMS. In particular, the ADEPT2 (successor of ADEPT) WfMS [79, 80] facilitates run-time alterations, enabling real-time modifications to active process instances, including the dynamic addition, deletion, and repositioning of process steps. Furthermore, this system allows for the evolution of process schemas while accommodating instance migration. This WfMS ensures flexibility by offering dynamic modifications to process schemas which can be seamlessly propagated to running process instances under the precondition that there is no conflict with existing instance states or any prior alterations.

Considering the analytical laboratory domain, Goodman et al. [81] have proposed a WfMS named LabFlow, dedicated to genetics. LabFlow employs a workflow model where objects move through tasks programmatically, which also supports parallelism and sub-tasks. The solution, developed in Perl5, provides an object-oriented framework for creating and executing workflows. It is designed to enhance laboratory efficiency by managing a wide spectrum of tasks, from simple computations to complex lab procedures, while maintaining robustness and adaptability to evolving lab techniques. Compared to CPEE, which is utilized in this thesis, LabFlow lacks a high level of implementation, such as a graphical interface for designing and monitoring workflows. Classes and methods need to be programmatically instantiated and invoked to use LabFlow. Furthermore, LabFlow defines its own workflow model, which is similar to Petri Nets [82], whereas CPEE adheres to the BPMN standard, which allows the definition and execution of highly customizable workflows. In summary, while LabFlow specializes in biology research workflows with a focus on parallelism and sub-workflows, CPEE offers a more generalized, flexible, and cloud-oriented workflow management solution suitable for various industries and IT environments. Unlike the domain-specific orientation of LabFlow toward biological research, CPEE serves a broader spectrum of industries and IT processes, offering a versatile cloud-centric workflow management solution [37, 83].

Several studies present a multitude of techniques for process discovery and conformance checking are presented in [40, 84, 85]. These techniques are categorized into two distinct types, namely offline and online techniques. Offline techniques focus on analyzing process execution logs after the completion of the process instance. This approach offers the benefit of having access to the entire dataset of the process. However, a significant limitation is the inability to correct malfunctions in process instances during runtime. In contrast, online methods are designed to evaluate process execution logs in real-time by processing event streams. These methods offer the advantage of enabling immediate corrections during process execution [86, 87].

Burattin et al. [88]presented a generic framework for online conformance checking that addresses weak starting points through the utilization of behavioral patterns. The methodology for assessing conformance is structured into three distinct components, that is compliance, completeness, and confidence. Primarily, conformance is evaluated by mapping and quantifying each discrepancy between observed and anticipated process behaviors, imposing a cost for deviations. Notably, the current framework does not differentiate between deviations that are intentional (such as a repair event) and those that are unintentional (such as missing events). An enhanced

cost function has been introduced to overcome this limitation. This function considers the data elements of subsequent events, allowing for a dynamic adjustment of the cost value [89]. The proposed method is versatile, lending itself to both online and offline applications. Empirical evaluation using real-world datasets has yielded encouraging results, particularly in the context of data elements that consistently exhibit similar outcomes.

In the medical domain, particularly in melanoma surveillance, process mining, conformance, and compliance-checking techniques have been employed to enhance the efficacy of patient monitoring [90, 65, 91]. The research was part of the Evidence-Based Medical Compliance Cluster (EBMC^2) project, which is a collaboration between the University of Vienna and the Medical University of Vienna. The research of Rinner et al. [90] aims to examine the compliance of skin cancer treatment processes with established medical guidelines by analyzing melanoma patients' surveillance data (event data) against established medical surveillance guidelines, represented as a process model. A novel technique termed "time boxing" has been introduced to effectively differentiate between follow-up visits, which integrated time information into event names. The project is a combined effort to bridge the gap between clinical research and practical healthcare delivery. Through the rigorous application of evidence-based principles, the project not only assesses compliance with established medical protocols but also integrates patient-specific and institutional variables into the analysis, enhancing the accuracy and applicability of its findings in improving evidence-based medical practices. In contrast to this work, which involves analyzing the process behavior near real-time generated data, the paper examined the process behavior based on existing long-term clinical data.

In the context of data-aware compliance checking, an advanced abstraction methodology was introduced in research by Knuplesch et al. [92] to address the challenge of state explosion, characterized by an excessively large number of states leading to elevated complexity by implementing a pre-processing step.

A framework for compliance monitoring that aims to effectively identify and manage violations of compliance rules during process execution has been introduced by Ly et al. [93]. This framework utilizes compliance rule graphs (CRG), a novel graph-based approach for representing compliance rules, which facilitates individual monitoring of rule activations and provides insights for compliance enforcement. The paper elaborates on the architectural design of this framework, the foundational principles of compliance rule graphs, and a proof-of-concept implementation by utilizing the SeaFlows Toolset [94]. The practical applicability and efficiency of the suggested approach are exemplified through a case study in the banking domain, underscoring its versatility and utility in diverse business compliance monitoring scenarios. This work also focuses on ensuring compliance during execution time by using a rule-based approach. However, the compliance rules are not represented through dedicated CRGs; instead, they are directly incorporated into the process model by utilizing BPMN. While CRG offers a specialized framework for rule monitoring and feedback, BPMN provides a more holistic view of both the process and its compliance aspects, integrating compliance monitoring within the broader BPM framework.

In laboratory automation, business process management is a strategic approach that aims to optimize and streamline laboratory processes by using structured methodologies and tools. The BPMN 2.0 standard is highlighted as a crucial framework for achieving these objectives. BPMN is recognized for its ability to provide a system-independent, interdisciplinary graphical process

control notation that is not only suitable for process analysis but also executable [95]. The significance of BPM in laboratory automation arises from its potential to enable end-to-end workflow automation and improved system integration. Laboratories often involve complex processes and data-intensive tasks, and they require real-time monitoring and control. Neubert et al. [95] have explored the integration of BPM in laboratory automation, particularly focusing on the potential of laboratory execution systems (LESs) to simplify the application of business process management systems (BPMSs). The integration of BPM solutions into structured laboratory automation presents unique challenges, particularly in scenarios involving complex processes, real-time-critical system integration, and long process chains across distributed automation systems and mobile laboratory robots. The paper suggests that LESs can significantly simplify the implementation of BPMS in hierarchical laboratory automation, particularly in complex scenarios. LESs can reduce the number of different interfaces between BPMSs and subsystems and simplify complex process modeling, making the integration of complex laboratory workflows more manageable. The paper also presents an example application that demonstrates the BPMS–LES approach in a mixture of manual and automated subprocesses. The use of LESs is shown to result in reduced integration effort for complex laboratory workflows.

This work adopts a comparable methodology by dividing the laboratory testing process into more manageable and streamlined sub-processes with the primary objective of mitigating complexity. Nevertheless, the coordination and oversight of these individual process instances are supervised by a centralized correlator, which subsequently triggers the process execution engine. It is noteworthy that the correlator assumes the function of a LES in this context, with the distinction being that the actual process execution occurs within the BPMS rather than within the LES.

Holzmüller-Laue et al. [96] have discussed the application of model-driven process automation in research & development (R&D) laboratories by using the BPMN 2.0 standard to integrate both automated and manual tasks across various laboratory systems. The paper highlights the challenges of current laboratory automation, which often features isolated subsystems and manual tasks that are not well integrated into an overarching workflow. By implementing a BPM-based approach, approach, the authors aim to achieve more unified control of both data and process flows. The provided approach promises improvements in process documentation, quality assurance, and complex data flow management, ultimately supporting more effective R&D activities in life sciences. The paper examines a broad application of model-driven workflow automation across various laboratory environments, while this thesis is more centered on compliance and error reduction, specifically targeting high-demand RT-PCR tests. Both works aim to optimize laboratory workflows but apply different methodologies and scopes within the context of lab automation.

The field of laboratory automation has undergone substantial evolution since the seminal contributions of Dr. Masahide Sasaki [97]. This evolution is characterized by the replacement of repetitive manual tasks with automation technologies, notably robotics, leading to enhancements in efficiency, throughput, and quality and a reduction in human error. Initially, early laboratory automation systems were predominantly adopted by high-volume laboratories, primarily due to the prohibitive costs of investment. However, there has been a significant decrease in the return on investment threshold, from handling 15,000 to 25,000 samples per day in 1995, to as low as 1,000

samples per day by 2017. This shift has rendered TLA systems more feasible for implementation in medium-sized laboratories [97, 19].

The implementation of a novel TLA system in the work of Yu et al. [98] has led to a marked decrease in both operational costs and the number of required working steps. Specifically, there was an 86% reduction in process steps, coupled with an annual cost savings of approximately USD 232650. This saving is equivalent to the labor costs of 2.5 full-time employees.

However, it is important to note that this TLA system was installed in a hospital with over 500 beds, a scale significantly larger than the laboratory considered in this study. Due to spatial constraints, replicating such a large-scale system is not feasible for small-sized laboratories, as is the case in this work, which poses a substantial challenge.

A possible solution to this challenge could be the use of collaborative robots (cobots), which are smaller, lightweightlighter, and easier to program compared to traditional laboratory automation systems. These cobots can perform tasks like such as handling and processing samples, thus significantly reducing the manual workload of laboratory technicians and improving their working conditions. A sThe successful integration and operation of these cobots in routine laboratory work is has been demonstrated in research by Heidrich [99], highlighting their potential to enhance efficiency and alleviate staffing issues in small labs. The research also underscores the economic viability and practicality of employing cobots in such settings, making it a promising solution for laboratory automation. Cobots have not been considered for implementation in this work. However, there is potential for their integration in future studies to enhance automation levels even further.

A paper by Kannan et al. [100], has provided a comprehensive review and discussion of the recent advancements and challenges within the realm of robotic technologies and laboratory automation. The author addresses contemporary issues, including the impact of the COVID-19 pandemic and the H1N1 virus. The paper focuses on leveraging automated sample handling and transport mechanisms as protective measures against infectious diseases. Moreover, the paper proposes the utilization of robotics as a potential solution to mitigate the spread of such diseases. Several critical challenges are highlighted, such as shortages in healthcare personnel, concerns over quality control, and societal resistance.

A framework for introducing laboratory automation has been provided by Hawker [22]. The cornerstone of this framework is workflow mapping, which is identified as a vital strategy for facilitating the transition toward automated systems. By employing workflow maps, it becomes feasible to pinpoint bottlenecks in laboratory operations, thereby delineating specific requirements for automation. The paper notes that process models have limitations in their capacity to inform decisions about automation capabilities. In contrast to this paper, this thesis integrates a process-based approach as part of the automation solution itself.

4 Use Case

The total testing process (TTP) in a laboratory is typically divided into three distinct phases, namely the pre-analytical, analytical, and post-analytical phases [101]. In some literature, the phases are further separated by introducing the "pre-pre-" and "post-post-" analytical phases to distinguish the initial test selection and result interpretation tasks from the more straightforward sample collection and transport and result reporting tasks [102, 103]. Based on the research of [101, 104, 105], the phases are structured as follows:

1. **Pre-Analytical Phase:** This initial phase encompasses all procedures and processes preceding the laboratory analysis of the sample. Key components of this phase include:
    - **Sample Collection:** Procurement of the appropriate biological sample (e.g., blood, urine, tissue) utilizing standardized techniques.
    - **Transportation:** Delivery of the sample to the laboratory under controlled conditions that preserve its viability and integrity.
    - **Storage:** Optimal storage of the sample, particularly if immediate analysis is not feasible.
    - **Preparation:** Processing the sample to render it suitable for analysis, which may encompass centrifugation, aliquoting, or chemical modification.
    - **Patient Data Management:** Accurate recording and management of patient information and test details.
2. **Analytical Phase:** This phase is the central phase of the testing process, where the laboratory executes the substantive analytical work. It involves the following components:
    - **Testing Procedure:** This procedure includes the application of specific methodologies and instrumentation for the examination of the sample.
    - **Quality Assurance:** Quality assurance concerns the implementation of rigorous quality control protocols to ensure the reliability and reproducibility of the test outcomes.
    - **Data Acquisition:** Data acquisition is the systematic collection of analytical data, either manually or via automated systems.
3. **Post-Analytical Phase:** This phase is the concluding phase, which involves the management and interpretation of the analytical results, such as:
    - **Result Validation:** Confirming the results in order to affirm their validity within the context of established quality benchmarks.
    - **Reporting:** Communication of the results to the healthcare provider.

- **Interpretation:** Provision of interpretative insights, where necessary, to assist in clinical decision-making.
- **Record Retention (if necessary):** Systematic archiving of patient data and test outcomes for compliance and future reference.

4.1 SARS-CoV-2 Testing Process

In a more general overview, the SARS-CoV-2 RT-PCR testing process can be broken down into six steps:

Sample Collection

Initially, a biological sample is obtained from the individual, typically involving the collection of nasopharyngeal or oropharyngeal swabs [106]. This method ensures the retrieval of cells and fluids from the upper respiratory tract, where the virus predominantly resides. An alternative method, particularly suitable for individuals who may find swabs uncomfortable or invasive, is the gargle test [107]. In this method, the individual is instructed to gargle a sterile saline solution for a specified duration, typically a few seconds. This action enables the saline to interact with the throat and oral cavity and collect respiratory secretions. The individual then expels the gargled solution into a sterile container. The gargle test is considered less invasive and can be particularly advantageous in testing children or those with certain medical conditions that contraindicate the use of swabs. The gargle test must be performed correctly to ensure the collection of a sample with sufficient viral material for accurate testing.

Transport and Storage

After collection, the collected samples should be immediately transported under controlled conditions to a diagnostic laboratory. It is imperative to maintain the integrity of the sample during this phase to prevent degradation or contamination. Samples can be preserved at a temperature of 2 – 8°C for a maximum of 72 hours post-collection [108]. For extended preservation, it is recommended to freeze the samples at -20°C or, ideally -70°C [109].

Sample Preparation

Upon arrival at the laboratory, the sample undergoes a preparation process. This process involves the extraction of nucleic acids, specifically RNA in the case of SARS-CoV-2, which is the genetic material of the collected sample. Furthermore, it includes the addition of a biochemical reaction mixture such as primers, nucleotide bases, enzymes to start and complete the reaction, and fluorescent indicators [110]. The process involves first pipetting the samples onto a 96-well plate first. A 96-well plate is a flat plate featuring 96 small, cylindrical containers (wells) arranged in a grid, used for simultaneous tests [111] (see Figure 4.1). This standard tool in analytical research and clinical diagnostic testing laboratories has a matrix of eight rows and 12 columns, totaling 96 wells. Each well is typically shaped like a cylinder and is used to perform separate reactions or hold

samples. The 96-well plate is versatile in its applications It allows the simultaneous processing of multiple samples under identical conditions, which is crucial for high-throughput screening, enzyme-linked immunosorbent assays (ELISA), cell culture, molecular cloning, and various other laboratory procedures. The design of the plate facilitates efficient use of space and resources while enabling automation and multiplexing in experiments. Afterward, the samples of a maximum of four 96-well plates are pipetted onto a 384-well plate by an automatic pipetting robot (see Figure 4.2). The reason for this intermediate step is that the PCR machine only accepts 384-well plates.

Figure 4.1: 96 (micro)well plate

Amplification and Detection

The extracted RNA is then subjected to a molecular technique known as reverse transcription polymerase chain reaction (RT-PCR) [112] by inputting the 384-well plate into a PCR machine. This technique amplifies specific segments of the viral RNA, if present, to detectable levels. During RT-PCR, the presence of the virus is determined by the identification of unique genetic sequences specific to SARS-CoV-2. Primers bind to specific sequences of the virus in the DNA. The reaction mixture containing the sample undergoes multiple thermal cycles, during which the quantity of the targeted viral sequence doubles with each cycle, resulting in an exponential increase. As the DNA strands are amplified, the fluorescent dye attached to the DNA becomes more abundant, and its fluorescence level increases, which is measured after each cycle [113]. The onset of this exponential amplification correlates with the sample's viral load, which means an earlier increase indicates a higher concentration of the virus [114]. The PCR machine outputs a file containing the raw results including the *cycle threshold* value (in short C_t value) which is then manually imported into an analyzing software.

The C_t value refers to the number of cycles of amplification required to generate a signal that is detectable above the baseline (background) signal. As previously stated each cycle of the PCR process doubles the amount of targeted genetic material, starting from what is present in the sample. Therefore, the C_t value inversely correlates with the amount of target genetic material present in the sample at the onset of the PCR process [115, 116].

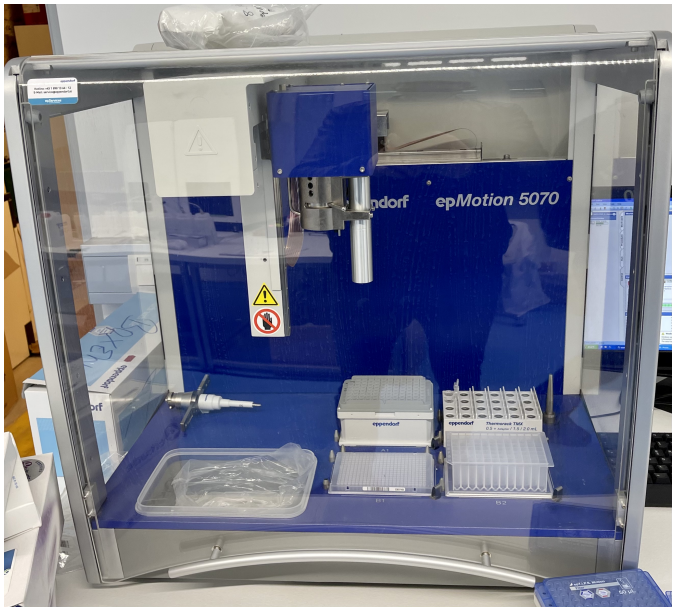


Figure 4.2: Eppendorf epMotion 5070 automated pipetting robot

- A low C_t value indicates a high amount of target genetic material in the sample, suggesting a strong presence of the pathogen.
- A high C_t value Indicates a low amount of target genetic material, suggesting a minimal presence of the pathogen.

Data Analysis and Interpretation

The results from the RT-PCR are analyzed meticulously. A positive result indicates the presence of SARS-CoV-2 RNA, confirming an active infection. A negative result suggests the absence of the virus, though this result does not completely rule out infection, especially in the early stages of infection or in cases of inadequate sample collection.

Reporting Results

Finally, the outcomes of the test are communicated to the patient, and in the event of a positive result, an automatic message is sent to the Epidemiologisches Meldesystem (EMS), which was mandatory at the time of writing the law [117].

4.2 Process Model and Workflow

A more detailed view of the individual tasks in the laboratory is presented below:

The description provided here intentionally adopts a higher level of abstraction in its exposition of the testing methodology, since the focus is on the workflow itself. Figure 4.3 depicts a streamlined process model, where the testing process is segmented into distinct pools, each representing a unique perspective. The subsequent sections expound upon these swimlanes. This approach has the benefit of segregating distinct elements, thereby diminishing overall complexity and delineating the intricacies of the process. Notably, throughout the testing phase, the sole identifier utilized for each sample is its unique barcode ID, which ensures the distinctiveness of each sample. The correlation of the samples with corresponding patient data is deferred until post-confirmation of test results. Consequently, this necessitates the importation of patient data by this phase of the process at the latest, as any delay would preclude the possibility of effective data matching. The extended exposition of the testing methodology, as illustrated in Figure 4.3, presupposes the completion of several preliminary stages, specifically the collection, transportation, and storage of samples, which have already been conducted before the commencement of the testing process.

Spawn 96-Well Plates

The process for detecting SARS-CoV-2 via PCR testing commences upon receipt of a patient-derived sample. This sample may originate from various sources, including, but not limited to, on-site laboratory collection or external entities such as government facilities and airports. Each sample is attached to a barcode label on its vessel, serving as a unique identifier and establishing a link between the sample and related patient information (e.g., name, insurance details). Subsequently, an available 384-well plate is identified and allocated for future use. Afterward, a new 96-well plate a critical component for PCR analysis (refer to Figure 4.3) is prepared. This plain instance waits until the 96-well plate is positioned on an automated pipetting robot. This transition initiates the continuation of the process, which involves decisions regarding either the activation of the pipetting robot or the preparation of another 96-well plate. Up to four 96-well plates can be placed on the automatic pipetting robot. The "Start automatic pipetting robot for current 384-well plate" task triggers the automatic pipetting robot process. In general, this process instance contains the central logic for controlling the process behavior, that is, the management process or the so-called plain instance. The plain instance is responsible for all subsequent process instances and provides a foundation for managing and orchestrating. The plain instance must be initiated before any other process elements, as it forms the foundational structure upon which subsequent processes are built. These subsidiary processes, termed "child processes", are inherently dependent on the plain instance and do not operate independently. The interdependence of these processes under the governance of the plain instance is crucial for effective workflow management.

Single 96-Well Plate

This process defines the flow for a single 96-well plate instance within a laboratory environment. Upon successful reception of samples in the laboratory, laboratory workers allocate these raw samples into a well plate, comprising up to 96 discrete wells organized in a matrix configuration. Each sample initiates a distinct subprocess, as outlined in the sample flow diagram. There are two scenarios for sample allocation:

1. In the single-well scenario, each well is occupied by an individual sample.

2. In the multi-well scenario, multiple samples are co-located within a single well, enhancing throughput capabilities. However, this method necessitates the individual retesting of all co-located samples in the event of a positive test result, thereby increasing the operational workload.

For quality control purposes, two wells on the plate are designated for control samples: One well contains a blank control expected to yield a negative result, and the other, contains a positive control. Discrepancies in expected outcomes, such as a positive result from the blank control, indicate anomalies during testing. After Post-sample allocation, the well plate undergoes RNA extraction via an automated machine, distinct from the PCR machine, which introduces necessary reagents, such as primers. Subsequently, the plate is transferred to an automated pipetting robot, triggering a concurrent waiting task, "Wait for well plate finished", within the workflow system. This phase involves awaiting confirmation of individual sample results. Upon individual verification of all samples, the entire well plate is collectively validated.

Automated pipetting robot

The automated pipetting robot, designed for high-throughput laboratory settings, precisely transfers liquid samples from a standard 96-well plate to a larger-format 384-well plate. This process involves the allocation of samples into the well plate while adhering to predefined volume and pattern requirements. The initiation of this transfer simultaneously generates a subprocess dedicated to managing the newly filled 384-well plate. This transition is a necessary intermediary step, as the PCR testing machine is exclusively compatible with 384-well plates. Upon the successful completion of the pipetting task, the robot's system triggers the activation of the waiting task of the 384-well plate subprocess.

384-Well Plate

The process remains in a state of suspension until operations are completed by the automated pipetting robot. In instances where the quantity of 96-well plates loaded onto the robot is less than four, an extra positive control sample is incorporated. This control adds another layer for validating the accuracy and reliability of the testing process. Subsequently, the 384-well plate is accurately positioned onto the PCR testing machine, where the primary testing sequence is conducted in an entirely automated manner. The biochemical reactions within the PCR machine are out of the scope of this thesis and will, therefore, not be discussed in detail. This phase of testing has a duration of approximately 40 minutes, during which the PCR machine cycles through various temperature changes, facilitating the enzymatic amplification of the target RNA sequences. Upon completion of this phase, a file containing the raw aggregated data is generated using the propriety .eds file format. Afterward, two operations are initiated concurrently: First, the test outcomes are rendered through the dedicated software of the PCR machine by importing the previously generated .eds file. Second, these results are integrated into the laboratory's information management system for further analysis. This system serves as a central repository for all test data

and is crucial for maintaining records, tracking sample progress, matching with patient data, and ensuring data integrity and compliance with regulatory standards. Each sample is then subject to result interpretation.

Sample flow

The flow of the sample is outlined by the red pool, denoting the flow of an individual sample. This approach is important due to its role in determining the diagnostic outcome for the patient. Initially, the sequence commences with the anticipation of PCR test outcomes from a designated 384-well plate. The process advances after the ingestion of raw data from the PCR machine corresponding to the above-mentioned 384-well plate into the laboratory information management system. A person with medical certification undertakes the examination and ratification of positive test outcomes. In contrast, negative results undergo automatic validation. Samples deemed non-viable during this phase are excluded and require repetition. Subsequently, the sample undergoes confirmation and matching with corresponding patient data. This patient information is integrated into the database via an import mechanism (see import flow). The sample then awaits the overarching approval of the 96-well plate. Upon approval of the 96-well plate, the outcome (excluding expelled samples) is mandatorily reported to the EMS. In the concluding phase, the sample outcome is exported in multiple formats (e.g., PDF, CSV), including patient notification regarding the test result. This export flow is initiated manually.

Import flow

In laboratory environments, the import and management of patient data vary significantly depending on the source of the samples. When samples are collected and processed within the laboratory's facilities (in-house), the associated patient information is already integrated into the laboratory's information management system. This integration facilitates the streamlined handling and tracking of samples throughout the testing process. However, in instances where samples are obtained from external sources (e.g., from pharmacies), a more complex and manual data management approach is required. These external samples, lacking prior integration into the laboratory's existing data systems, necessitate a separate and, often, manual entry of patient information. This discrepancy in data handling processes can introduce challenges in maintaining the consistency and accuracy of patient records. Moreover, the timing of patient data transmission from external sources to the laboratory is not standardized and can occur at various, unpredictable points in time during the testing sequence. To address these challenges, the laboratory implements a systematic process for importing patient data. This process includes the periodic querying and updating of information from mail and file transfer protocol (FTP) servers, which serve as channels for receiving patient data from external entities. The data thus retrieved is then saved and integrated into the laboratory's internal information system. Crucially, the received and imported patient data say nothing about the quality of the data.

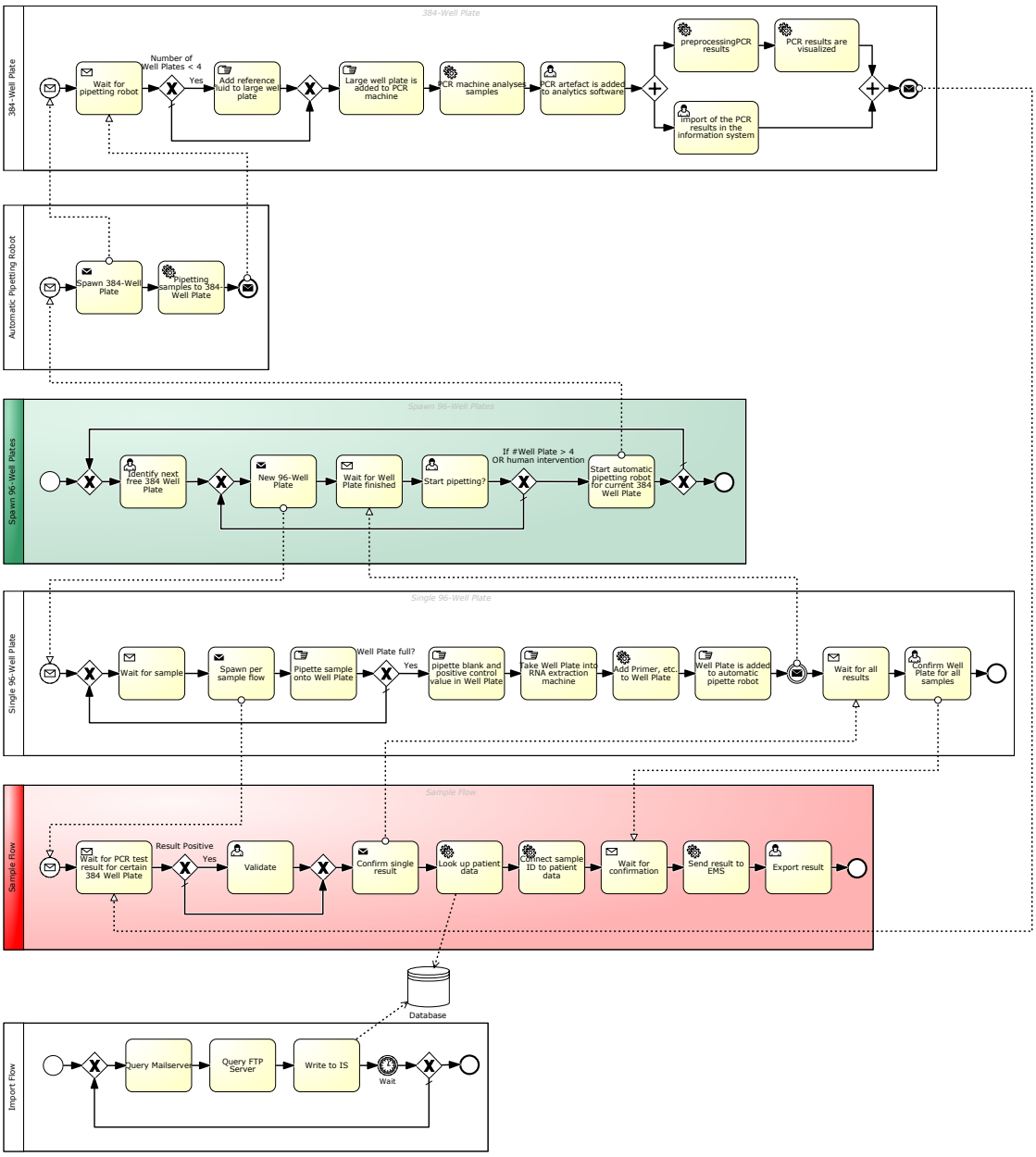


Figure 4.3: A Business Process Modeling and Notation (BPMN) representation of the COVID-19 testing process

5 Solution Design

5.1 Requirement Analysis

The initial phase involves establishing the essential requirements for the proposed solution based on the problem statement and the objectives. The appropriate architecture can then be formulated based on these established requirements. It is important to differentiate between general requirements, which include broad statements outlining the overarching goals and anticipated outcomes of the desired solution, and technical requirements, which involve precise and detailed directives about the technical facets. This distinction aids in facilitating the effective communication and comprehension of the technical specifications crucial for a successful implementation.

5.1.1 General Requirements

The following general requirements are crucial for the effective implementation of the solution.

1. **Rule-Based Engine:** The solution should incorporate a rule-based engine capable of interpreting and enforcing complex rules. The engine should be able to handle rule variations, exceptions, and dependencies to accurately assess compliance.

2. **Real-Time Monitoring:** Compliance is an ongoing process, and it is crucial to monitor it continuously. The solution should provide real-time monitoring capabilities, enabling the laboratory to detect and address compliance issues promptly.

3. **Audit Trail and Reporting:** The solution should maintain a comprehensive audit trail of compliance checks and activities performed. It should generate detailed reports and logs, allowing the laboratory to demonstrate compliance, identify patterns, and facilitate auditing processes.

4. **Notifications and Alerts:** The solution should be able to generate notifications and alerts when non-compliance issues are detected. Furthermore, the solution should provide timely notifications to relevant stakeholders, enabling them to take corrective action promptly.

5. **Security and Compliance:** Compliance-related data is often sensitive and confidential. The solution should adhere to robust security standards, ensuring data privacy, encryption, access control, and protection against unauthorized access or breaches.

6. **User-Friendly Interface:** The solution should have a user-friendly interface that allows non-technical users to easily configure compliance rules, monitor compliance status, view reports, and manage compliance-related tasks effectively.

7. **Interoperability:** The solution should integrate seamlessly with various data sources within the organization, such as databases, file systems, application programming interfaces (APIs), or other software applications, to access and analyze relevant data for compliance checking.

5.1.2 Technical Requirements

The following technical requirements are necessary for effectively implementing the solution.

1. The selected architectural design should facilitate scalability, adaptability, and maintainability.
2. The application should possess deployability and management ease, enabling efficient scalability and updates.
3. The solution should include all necessary dependencies and essential libraries to minimize potential conflicts.
4. The application should demonstrate consistent operability across diverse environments, ensuring portability, particularly when running on multiple platforms such as x86 and ARM.
5. Incoming requests should be directed to the appropriate backend servers and accompanied by suitable status codes and error messages to effectively handle both successful and unsuccessful requests.
6. Services should be able to access and manipulate data in a standardized and efficient manner.
7. Communication should exclusively occur over a secured connection.
8. All data must be securely stored within a local database.
9. The entire solution must run locally in the provided environment of the laboratory.

5.2 Architectural Considerations

Architectural considerations were derived based on the general and technical requirements defined in the previous chapter.

To fulfill the first and second technical requirements, a microservice architectural pattern was chosen over a monolithic approach. In a microservice-oriented architecture, the solution is structured as a collection of small, loosely coupled, and independently deployable services. Each solution is responsible for a specific business capability or use case. This approach is advantageous as the separation enables development teams to work on individual components independently, thus augmenting both efficiency and productivity by negating dependencies. This property of decentralized development concurrently enhances the ability to implement technology-agnostic solutions, as the solution may be built using different technology stacks. Furthermore, microservice architecture also significantly enhances scalability. Specifically, it facilitates the scaling of

individual components as per their respective demands rather than scaling the entire application. This targeted scaling reduces resource requirements and optimizes the performance of high-load components, thereby contributing to cost-effectiveness. Moreover, resilience is a core advantage of using a microservice architecture. This property means that a failure of a single service does not immediately lead to a system-wide outage. Instead, the architecture is designed to prevent failure propagation, thereby ensuring that the overall application remains operational even if a single module faces an anomaly. However, the potential downsides of utilizing the microservice architecture include complications in managing services; increased complexity in data consistency maintenance (i.e., maintaining data integrity); challenges in deployment; potential for higher latencies and overheads due to the need for inter-service communication; difficulties in testing, debugging, and troubleshooting; and susceptibility to network failures.

The exchange of information between services occurs through representational state transfer (REST) APIs, ensuring the security of data transmission through the implementation of the Hypertext Transfer Protocol Secure (HTTPS) protocol. Data resources are effectively interchanged using the JavaScript Object Notation (JSON) format. REST APIs offer a straightforward and intuitive mechanism for data exchange between disparate systems. As JSON is a lightweight data format, it facilitates ease of comprehension and composition for developers. JSON's minimal syntax requirements and familiar object-oriented structure contribute to simplifying the manipulation of data. This functionality corresponds to the specified fifth general requirement and the sixth and seventh technical requirements.

Storing the data in a local database required making a design decision of whether to store the data in a relational database (RDMS) or a document-oriented database (i.e., NoSQL DB). Relational databases use structured tables with predefined schemas, while document databases store data in flexible, semi-structured document formats. NoSQL databases are highly scalable and can handle large amounts of data (e.g., by using MapReduce) and high write loads more easily than traditional RDBMSs. Since the main use case is to store large amounts of event data, which is often semi-structured, this thesis used a document-oriented (NoSQL) database.

Considering the second, third, and fourth technical requirements, the thesis utilized containerization instead of virtualization for the microservices. Containerization and virtualization represent two different strategies for segregating computing resources, and each strategy has unique strengths and weaknesses. On the one hand, containerization encapsulates the software application environment, isolating each application within its own ecosystem but sharing the same operating system. On the other hand, virtualization creates a full-fledged virtual machine with its own operating system (OS), over a hypervisor layer. The lightweight nature of containers causes a less resource-intensive environment compared to virtualization, improving the system's overall performance. Additionally, the reduced boot-up time for containers translates to a faster deployment of applications. Containerization also offers greater scalability and portability, facilitating the easy replication, distribution, and migration of applications across multiple platforms and cloud environments.

Various solutions are available when considering options for notifications and alerts. One commonly employed method involves the utilization of automatically generated email messages. This method necessitates the establishment and configuration of a dedicated local email service, as outlined in the ninth technical requirement. Alternatively, the implementation of a short message

service (SMS) could be considered. However, due to concerns regarding cost and security (as SMS messages are typically unencrypted), SMS does not appear to be a suitable choice for this particular use case. Instead, this thesis decided to employ a bot to fulfill this requirement. Bots possess the advantage of enabling real-time communication, thereby accelerating response times and enhancing overall transactional efficiency. The promptness and effectiveness of bots offer a significant edge over traditional communication methods such as email or SMS. Furthermore, email and SMS messages have a limited potential for dynamic user interaction. Their unidimensional style of communication lacks the interactive engagement provided by bots, which can engage in intricate conversations through the integration of artificial intelligence and natural language processing techniques. This functionality fosters an enhanced user experience and contributes to increased satisfaction levels.

To address the fifth technical requirement, the thesis determined that it would be beneficial to implement a reverse proxy. One notable advantage of employing a reverse proxy is its ability to effectively distribute client requests among multiple servers, thereby ensuring optimal performance and mitigating service disruptions. This capability to evenly allocate computational workload is crucial for effectively managing high-traffic applications, minimizing the risk of individual server overload, and facilitating uninterrupted service delivery. Moreover, reverse proxies offer an additional layer of security by acting as an intermediary between the client and the servers. This attribute strengthens the system's resilience against potential malicious activities, such as distributed denial of service (DDoS) attacks. Consequently, this heightened security framework provides a more robust defensive mechanism, enhancing the overall integrity and reliability of the network environment. Furthermore, the inclusion of a reverse proxy can enhance operational efficiency through its SSL termination feature. By handling the resource-intensive SSL/TLS handshake process, the reverse proxy relieves application servers from this demanding task, enabling them to concentrate on their core functions and maximizing the overall performance of the system.

5.3 Architectural Overview

Seven individual services were implemented based on the discussed architectural considerations and requirements:

1. **Backend Service:** encapsulates the correlator and primarily communicates with the CPEE.
2. **Process Execution Engine:** executes and manages the workflow.
3. **Storage:** stores all the data.
4. **Bot:** handles notification and alerting. It can also be utilized for reporting and monitoring proposals.
5. **Frontend:** user interface (GUI) for reporting and visualizing of the log data.
6. **Logging Service:** receives all raw log data from various sources and stores it in the database. The service notifies the backend service when a new log has been received.
7. **Reverse Proxy:** hides backend and frontend services from being accessed directly.

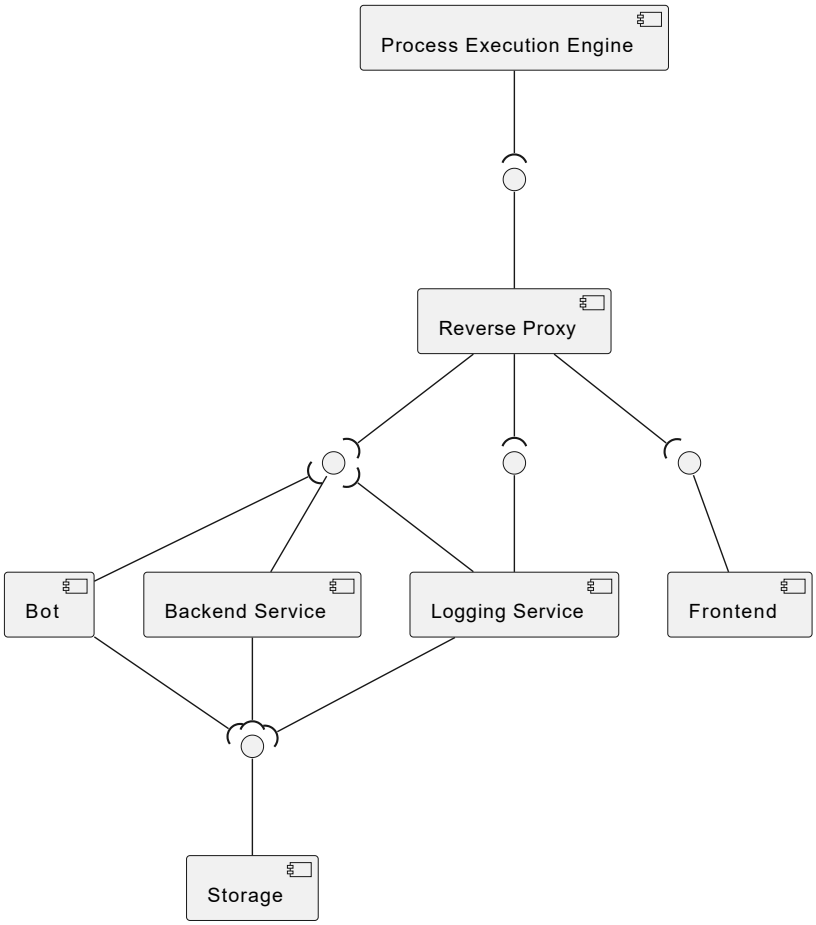


Figure 5.1: UML component diagram of the proposed service architecture

5.4 Evaluation Regarding Tracking and Automation Capabilities

5.4.1 384-Well Plate

The manual tasks associated with inserting the reference fluid and adding the 384-well plate into the thermocycler require cobots, which are a possible solution if both the automated pipetting robot and the thermocycler are in proximity. Unfortunately, spatial constraints within the existing laboratory arrangement necessitated the separate placement of these devices in different rooms, posing significant challenges to automation efforts.

The laboratory's thermocycling processes use three distinct models of QuantStudio™ namely, QuantStudio™ 3, 5 (see Fig. 5.2), and 7, manufactured by Applied Biosystems which is a brand of Thermo Fisher Scientific Inc. are used. All three devices are connected to the network via LAN. The thermocycler is equipped with a predefined program, which is manually started by personnel after the insertion of the 384-well plate. Upon completion, the resulting.eds file is automatically stored on a local server message block (SMB) share, from where it is manually imported into both the

provided analytics software (QuantStudio™ Design and Analysis Software) and the LIS. Although the automation of file importation into the LIS can be achieved by implementing a script that monitors the designated folder within the SMB share and retrieves the file upon its appearance, automated integration into the provided analytics software is currently unfeasible, as the vendor's software solution lacks this capability at the time of writing this thesis.

Notably, at present, the thermocycler does not offer any APIs to end-users, restricting access solely to authorized original equipment manufacturer (OEM) partners. However, a third-party-developed software development kit (SDK) called QSLib enables programmatically controlled interaction with the thermocycler through the Standard Commands for Programmable Instruments (SCPI) interface [118, 119]. While utilizing, this SDK facilitates direct data retrieval from the instrument and the use of control commands, it is recommended not to do so due to the associated risks, including voiding of warranty and potential irreparable damage to the instrument resulting from the execution of raw commands.

Figure 5.2: QuantStudio™ 5 thermocycler by Applied Biosystems

5.4.2 Automatic Pipetting Robot

The Eppendorf epMotion 5070 automated pipetting robot (see Figure 4.2) is solely connected via USB to its proprietary MultiCon system at present, and it is not integrated into any information system or linked to the local network. Consequently, obtaining direct access to log data via APIs poses a challenge. The MultiCon system comprises a touchscreen-based Windows 10 PC equipped with an RJ45 network port. Unfortunately, extensive internet research has not clarified whether the MultiCon system provides any APIs. One potential solution for monitoring this system involves installing cameras connected to a Raspberry Pi equipped with image recognition capabilities, such as convolutional neural networks (CNN), to automatically detect specific positions of the robot.

5.4.3 Spawn 96-Well Plates

Both 384 and 96-well plates can be made traceable through the application of barcodes, facilitating streamlined identification processes through straightforward barcode scanning. Automation of the identification of the next available 384-well plate can be achieved by integrating a barcode scanner next to a collection of unused 384-well plates. This setup automatically conducts barcode scans upon the removal of a plate from the stack, ensuring efficient plate selection.

5.4.4 Single 96-Well Plates

The process of dispensing samples onto the 96-well plate positions is systematically logged in the LIS. To facilitate this task, the laboratory has developed a proprietary hardware solution termed *Sortinator*, which serves as a pipetting aid. After scanning the barcode of the sample (specifically, a swab) using a dedicated barcode scanner, the LIS promptly retrieves the corresponding position on the well plate, indicated by an LED marker. Subsequently, laboratory personnel pipette the sample into the marked position, followed by the scanning of the sample, thus continuing the iterative process. As a consequence of this procedure, the LIS is endowed with the precise positional data of each sample on the well plate, enabling the automation of subsequent steps.

Figure 5.3 illustrates the *Sortinator* device, both with and without a 96-well plate inserted. The device's interface provides real-time information regarding the current position on the plate, along with its plate identifier. Additionally, relevant status updates are displayed below. Connectivity with a computer is established through a USB cable featuring a Mag6 connector (that uses magnetic coupling) for swift disconnection during workspace or device cleaning routines. At present, this hardware architecture centers around a microcontroller responsible for interfacing with the LEDs via the I2C protocol. The casing design is made via 3D printing, facilitating rapid and facile production, if required. Currently, the laboratory operates four such devices. The software driving the device is developed in Java and is compatible with any barcode scanner featuring a Com USB interface. The initialization and termination of plate operations as well as other commands are achieved by scanning the designated command barcodes, which encompasses actions such as pooling, setting blank values, and other configurable parameters. The software interface also includes provisions for ensuring adherence to pipetting protocols, thereby minimizing the risk of duplications. Additionally, supplementary plate information and configurable settings are accessible through the software interface (refer to Figure 5.4).

In the realm of task automation, the deployment of cobots or a total laboratory automation system are viable solutions. However, logistical and financial constraints currently prevent the implementation of either solution.

5.4.5 Sample Flow

Each sample instance originates from its respective parent well plate instance. Significantly, the entire sample flow is conducted within the LIS, enabling facile monitoring through the implementation of listeners to these tasks. However, the sample flow necessitates interaction with both the 96- and 384-well plates to receive sample results and 96-well plate confirmations. The "Validate" Task can also be monitored by observing user inputs. Theoretically, this task could be

Figure 5.3: *Sortinator* device with (right) and without (left) a 96 well plate

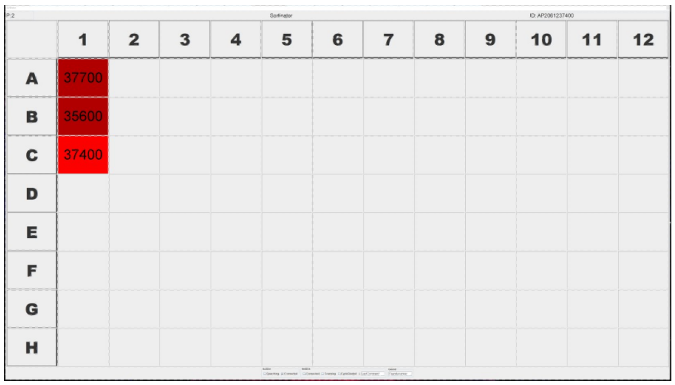


Figure 5.4: LIS representation of a 96-well plate showing *C1* as the current position

automated by validating samples against pre-established rules or leveraging artificial intelligence. However, legal constraints prevent the automatic validation of positive samples, mandating the involvement of trained specialists for this purpose. The traceability of the "Export result" task is facilitated through user-triggered events. It can be automated through two potential approaches:

1. Programmatically modifying the LIS by implementing new export rules and protocols, and
2. Removing the export functionality from the LIS and integrating export rules directly into the process model, which is subsequently executed by a WfMS. However, this approach may result in a highly complex process model, potentially compromising readability, particularly in instances involving a variety of diverse export rules and methods dependent on the sample ID.

5.4.6 Import Flow

The import flow is outside this thesis's scope due to the necessity for direct access to patient data, a practice prohibited by the laboratory owing to the highly sensitive nature of this information.

Nevertheless, the import process has been completely automated through the retrieval of patient data via mail and FTP servers, followed by its integration into the LIS database. The execution of the import process is handled by the LIS itself.

5.5 Evaluation Design

The proposed approach in this work integrates theoretical models with practical applications, ensuring that the enhancements proposed are both feasible and effective in improving the operational efficiency and reliability of laboratory testing processes. These improvements are evaluated in the following stages.

5.5.1 Comparative Analysis with a Four-Step Integration Strategy

A comparative analysis and classification of the practical implementation provided in this work is conducted against the documented four-step integration strategy of Mangler et al. [120]. This strategy was originally intended for small- to medium-sized enterprises in the manufacturing sector, tailored to the context of laboratory operations. The strategy also involves addressing identified digitalization gaps in the current test setting and discussing potential advanced automation capabilities where process-aware information systems assume full control over operations.

5.5.2 Empirical Data Collection and Analysis

The design involves the integration of real-time data collection mechanisms into the LIS without altering existing laboratory operations. As a result, the implemented solution leverages extensive empirical data derived from the laboratory's daily operations of SARS-CoV-2 testing over a set period. This collected dataset includes hundreds of thousands of log entries such as sample scans. The sample scans offer a comprehensive analysis of operational dynamics, including the tracking of test positivity rates and sample validation processes, providing in-depth insights into operational trends, effectiveness, and areas for improvement in the lab's testing processes. The data is visualized and analyzed using tools such as PowerBI and the implemented frontend service.

5.5.3 Process Compliance

A critical component of the evaluation involves the application of predefined compliance rules to the empirical dataset to identify and rectify process anomalies and inconsistencies. The identified violations in a process-aware information system are compared with conventional manual methods in terms of tracking speed by calculating the delta time, which is defined as the time difference between the occurrence and detection of the violation. This systematic compliance check highlights opportunities for process improvements and enhances the system's adaptability to evolving operational needs.

6 Implementation

6.1 Technologies

The following key technologies were used to implement the artifacts.

6.1.1 Cloud Process Execution Engine (CPEE)

The CPEE [37] is an advanced, open-source process engine designed to facilitate business process management (BPM) and manufacturing orchestration management (MOM). It represents a service-oriented approach, characterized by a highly scalable and distributed architecture. The CPEE significantly eases integration with external services and systems, underscoring its adaptability for diverse application scales. Central to the CPEE is its emphasis on low-code, model-based process execution, which democratizes the orchestration of software, machinery, and human resources. This feature is crucial in enabling dynamic modifications and evolutionary adaptations in process management. The engine employs a RESTful interface for control mechanisms, ensuring the efficient management of process instances and data flows. This functionality is complemented by a data stream interface that communicates state information about instances and activities to connected microservices, enhancing operational oversight. The CPEE's transparency and modularity render it a valuable asset in academic and research contexts, facilitating experimental and educational endeavors in BPM. Its robust and flexible architecture also makes it a sought-after solution in industrial domains, where scalability and adaptability are imperative.

6.1.2 Docker

Docker [121, 122] is an open-source platform designed to facilitate the development, shipment, and deployment of applications by utilizing containerization technology. It enables the encapsulation of an application and its dependencies into a container, which can be then executed in any environment that supports Docker, ensuring consistency across various computing environments. This technology is predicated on the concept of containerization, where each container operates independently, sharing only the host system's kernel but otherwise running in isolated user spaces. Docker automates the deployment of applications inside lightweight, portable containers, significantly enhancing the efficiency and flexibility of application deployment and scaling. Containers are more lightweight than traditional virtualization approaches, which allows more efficient use of system resources and faster startup times.

6.1.3 MongoDB

MongoDB [123, 124] is a document-oriented database management system that diverges from the traditional table-based relational database structure as it stores data in flexible, JSON-like documents. This approach enables the representation of hierarchical relationships, supports array-based data types, and allows for a more dynamic and scalable data model. MongoDB is designed to facilitate high availability, horizontal scaling, and geographic distribution, which are achieved through features such as replication and sharding. Its architecture allows developers to build applications with greater speed and efficiency, owing to its flexible schema and ability to handle large volumes of data and complex transactions. MongoDB's operational model promotes the use of documents that can vary in structure, offering a more nuanced and adaptable approach to data management that aligns with the demands of modern software development practices.

6.1.4 Nginx

Nginx [125, 126] is a high-performance web server software designed to serve web content and act as a reverse proxy and email proxy server. It is distinguished by its efficiency, stability, comprehensive features, easy configuration, and minimal use of resources. Nginx operates on an asynchronous, event-driven architecture, which enables it to handle thousands of concurrent connections on a single thread without significant overhead. The deployment is widespread across various online platforms, ranging from small-scale projects to large-scale enterprises, due to its robust performance capabilities and ability to efficiently distribute web content and manage server loads.

6.1.5 Node.js

Node.js [127, 128] is a cross-platform, open-source runtime environment, that enables the execution of JavaScript code beyond the confines of a web browser and represents a significant advancement in the development of highly scalable, server-side applications. It is predicated on the V8 JavaScript engine, developed by Google, which compiles JavaScript directly into native machine code. This feature enables Node.js to offer exceptional performance and efficiency for real-time applications. Node.js adopts an event-driven, non-blocking I/O model, which facilitates concurrent request handling, thereby significantly enhancing the scalability and throughput of applications. This model is particularly well-suited for applications that require real-time data processing, such as chat applications, live streaming services, and online gaming platforms.

6.1.6 VueJS

VueJS [129] is an open-source JavaScript framework that follows the model-view-viewmodel (MVVM) pattern, designed specifically for developing user interfaces and single-page applications (SPA). It is characterized by its progressive nature: It can be integrated into projects with varying complexities, from small-scale applications to extensive, sophisticated single-page applications, by adopting incremental adoption. This framework facilitates the creation of dynamic user interfaces

through a declarative rendering approach, utilizing a reactive data-binding system that ensures efficient updates and renderings of the user interface in response to data changes.

6.2 Architectural Implementation

Based on the architectural overview provided in Section 5.3, the following implementation architecture emerges:

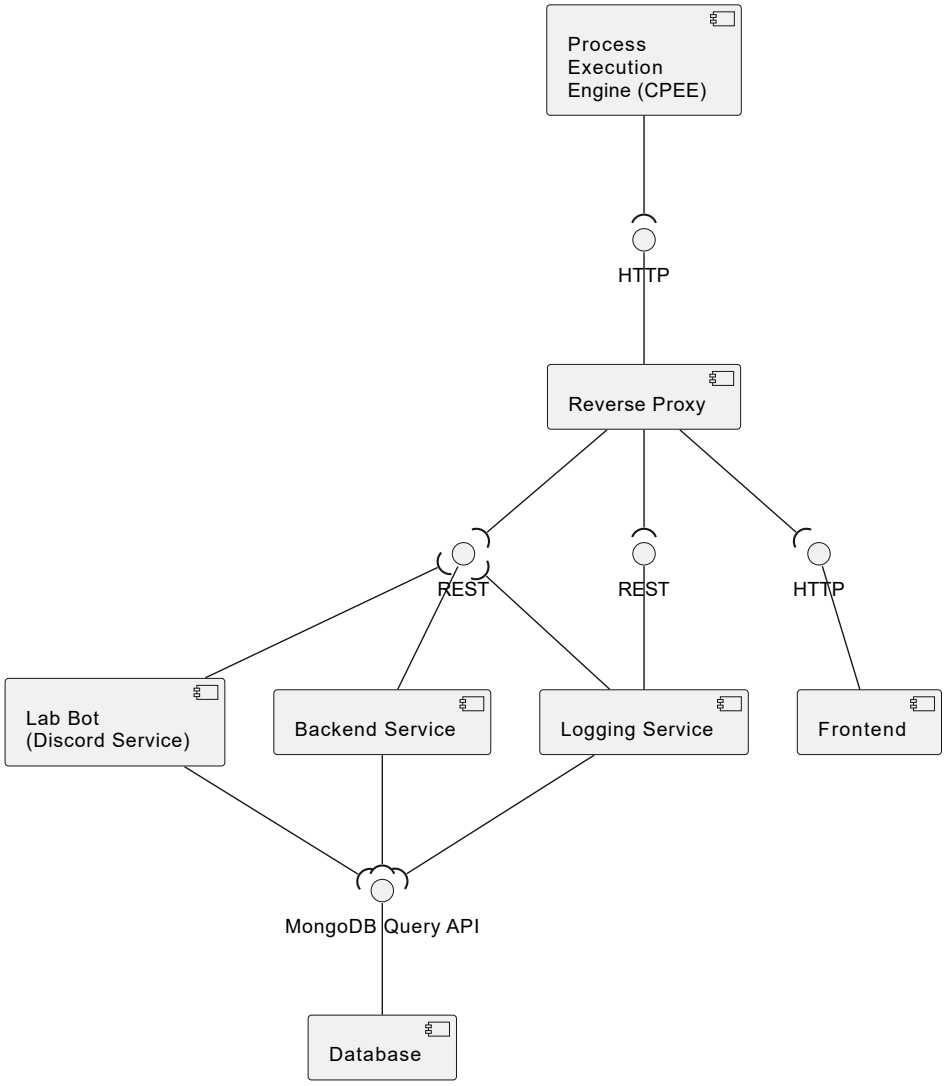


Figure 6.1: UML component diagram of the implemented service architecture

In this thesis, all services, except for the CPEE, have been deployed by utilizing an open container initiative (OCI) based image. The individual services are discussed in more detail in the following

subsections.

6.2.1 Backend Service

The backend service contains the correlator, which acts as the main orchestrator for matching waiting process instances in the process engine with incoming event logs from the laboratory information system. The backend service receives event logs from the logging service and saves them in a database. Subsequently, the correlator engine is activated. Moreover, the backend service receives all callback requests from the CPEE and saves them in the database. This approach ensures the persistent storage of all states, even after an intentional or unintentional restart. Access to all currently saved callbacks is feasible through an HTTP/S GET request. The application is being realized using Node.js. In addition, three distinct service endpoints have been implemented:

Endpoint	**/pcheck**
HTTP-Method	POST
Request	Arrays containing created and finished plate IDs
Response	Arrays containing plate IDs for which the initiation duration has surpassed four hours without completion
Description	The request necessitates an enumeration of created and finalized plate IDs. This functionality can be integrated into the process model for adherence monitoring. The returned information can be used to notify the laboratory personnel (see */notifyall* endpoint below).

Endpoint	**/notifyall**
HTTP-Method	POST
Request	Message and log level for the notification
Response	HTTP status code representation as to whether the notification has been broadcasted successfully
Description	This request initiates a broadcast from the laboratory bot across all communication channels, containing the provided log level and message content. It facilitates the transmission of notifications from process instances in a simple way.

6.2.2 Database

In the database infrastructure, a local MongoDB instance has been employed for production purposes, while a MongoDB Atlas Free instance is utilized for development purposes. The database contains the following datasets:

1. Raw log data concerning the execution of individual tasks.
2. Correlator-relevant information, encompasses messages originating from both the producer (i.e., executed tasks within the laboratory) and the consumer (i.e., pending process instances within the CPEE).

Endpoint	**/timeout**
HTTP-Method	POST
Request	Duration in minutes for how long the timeout should run
Response	HTTP status code representation as to whether the timeout has been set successfully
Description	This request represents an enhanced variant of the default timeout endpoint from the CPEE. It calls the task back in a process instance after a predefined period. Uniquely, this advanced version also allows resuming the execution before the timeout has been exceeded, potentially initiated by a concurrent task. The timeout is internally aligned using the CPEE instance's universally unique identifier (UUID), retrievable from the header.

3. Pending timeouts, associated with process instances.

6.2.3 Lab Bot

Section 5.2 explores the benefits of using chatbots over traditional communication methods such as email or SMS. Presently, the market offers a vast number of bot service providers for diverse messaging platforms, including Microsoft Teams, Webex, Discord, and Telegram. While the market hosts a broader array of bot platforms, this analysis focuses exclusively on the aforementioned four for the sake of limiting complexity. The primary challenge involves selecting the most appropriate platform that aligns with the laboratory's specific requirements. Microsoft Teams and Webex have been preemptively excluded from consideration, as they are not currently utilized within the laboratory environment, and there are no existing software licenses for these platforms. Consequently, the evaluation narrows down to analyzing the capabilities of bots on the Discord and Telegram platforms.

Discord-based bots typically exhibit advanced integration capabilities. These bots can seamlessly interact with the rich feature set of the Discord platform, such as voice channels, server management, and extensive role-based permissions. This integration enables a more holistic and dynamic interaction within the Discord environment, tailored to the specific needs of each server. Moreover, Discord bots often demonstrate greater customization and flexibility. They are designed to cater to a wide range of functionalities, from simple commands to intricate automation and interactive games. This versatility is augmented by a robust development community, which continuously contributes to the bot ecosystem with innovative applications and features.

The user interface and experience in Discord are generally considered to be more user-friendly and engaging compared to Telegram. Discord provides a more immersive environment with its support for voice and video communication, along with text-based interaction. This multimodal communication framework enhances user engagement and interaction on the platform. On the other hand, Telegram bots are recognized for their simplicity and ease of use, making them suitable for straightforward tasks and interactions. One advantage of Discord over Telegram is that the server region can be set explicitly, which ensures that the data is only processed in the defined region. Due to the sensitivity of the processed data and advanced integration capabilities, this thesis implemented a Discord-based bot solution.

The lab bot functions as the central communication hub in the entire system architecture. Additionally, it facilitates the direct management of process instances. Its primary functionalities include the following:

- Notifications of unintended deviations during process execution, such as violations of compliance rules. For example, if a sample remains in a specific phase beyond a predetermined duration, laboratory personnel receive notifications to initiate corrective actions. These alerts encompass a log level and a modifiable message text.
- Health status of the CPEE instance, encompassing metrics on process instances in various states, such as ready, running, and stopped. These metrics include data on orphan samples (i.e., sample process instances lacking an associated, completed well plate process instance), as well as details on the current plain instance and active well plate instances, including CPEE URL, instance creation timestamp, and execution status. For well plates, the quantity of enclosed samples is also visible.
- Compilation of statistical data on process instances executed on a given day, including counts of created well plates, finished well plates, validated well plates, scanned samples, deleted samples, positive samples, negative samples, and the percentage of positive tested samples.
- Provision of in-depth data regarding specific well plates or samples, such as event chronology and overall status.
- Governance of overall laboratory process instances within the CPEE, encompassing the termination and initiation of new plain instances or the elimination of all subsidiary process instances of a plain instance. This feature is particularly advantageous in the context of process modification trials, enabling a comprehensive reset of the executing process hierarchy through a single command execution.

The bot has been implemented by using Node.js together with the discord.js[1] library, which interacts with the Discord API by using web sockets. The bot communicates with the CPEE via the provided RESTful API by using Axios[2]. The bot has direct access to the database to receive sample and well plate data.

A list of all available commands is provided below:

6.2.4 Frontend

The frontend architecture is built upon the Vue.js JavaScript framework, encompassing three distinct functionalities:

Dashboard

The overarching purpose of the dashboard is to provide stakeholders with a centralized platform for accessing and analyzing crucial business intelligence insights derived from log data. By aggregating

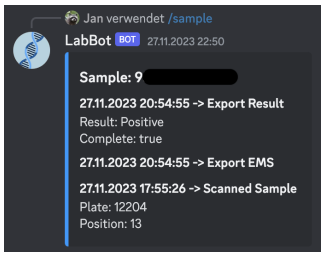
[1] **`https://discord.js.org`**, last accessed: 2024-06-01
[2] **`https://axios-http.com`**, last accessed: 2024-06-01

Command	Description
/lab	Returns the current (health) status of the lab-related process instances (see Screenshot 6.2c).
/lab_action start	Spawn and start a new plain instance if not already exist.
/lab_action restart	Abandons all lab-related instances (including samples and wellplates) and spawns a new plain instance (see Screenshot 6.2b).
/lab_action reset	Abandons all wellplate and sample instances while the plain instance stays untouched.
/lab_action abandon	Abandons all lab-related instances including the plain instance.
/plate [*plateid*]	Returns the current wellplate status of *plateid* (see Screenshot 6.2e).
/sample [*sampleid*]	Returns the current sample status of *sampleid* (see Screenshot 6.2a)
/stats [*date*]	Returns statistical information on the provided *date*. If no date is provided, then the current date is used. The data must be provided in the following date format: DD.MM.YYYY (see Screenshot 6.2d).

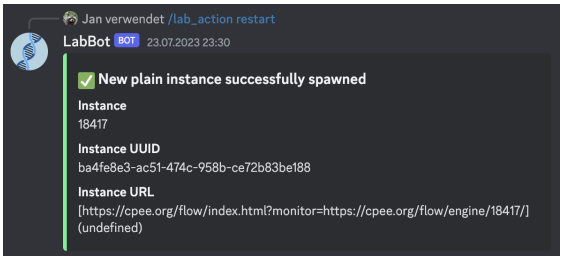
and presenting relevant metrics and trends, the dashboard aims to empower users to glean actionable insights, thereby fostering informed strategic planning and resource allocation. The dashboard furnishes an overview of relevant data within specified time intervals, including the following:

- The absolute count of scanned and exported samples
- The total count of created and finished well plates
- The average sample count per well plate
- The average daily sample processing rate
- The percentage distribution of positive and negative test outcomes
- The percentage distribution of user-assigned task types
- A bar chart illustrating the temporal distribution of a selected task type, with the x-axis denoting dates and the y-axis representing the absolute occurrence count of the specified task type
- A line graph displaying the relative frequency of positive sample outcomes over time, with the x-axis indicating dates and the y-axis depicting the relative percentage of positive samples (see Figure 6.4)

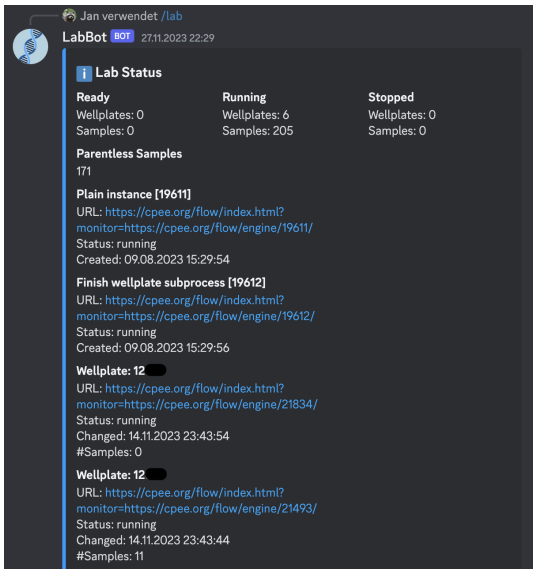
Both the line and bar charts support dynamic zoom functionality, allowing users to delve deeper into the data by selecting specific areas of interest. To reset the zoom level to its default state, users can simply double-tap the graph.



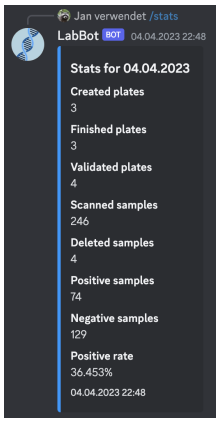
(a) /sample [*sampleid*]



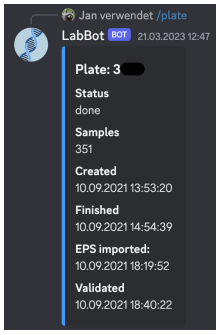
(b) /lab_action restart



(c) /lab



(d) /stats



(e) /plate [*plateid*]

Figure 6.2: Excerpt of possible bot commands

Logs

The logs section provides a visual representation of the collected raw logs in tabular format, as depicted in Figure 6.6. Users can conduct searches for relevant log entries based on either a sampleid for specific sample searches or a plateid for specific well plate searches. The table comprises several columns, including the following:

- **ID:** Corresponding task type ID
- **Name:** Task type designation
- **Info:** Displays relevant information associated with the task type. For example, the "Export result" task type exhibits the sample result, categorized as either positive or negative. Conversely, the "Export EMS" task type showcases the sample ID.
- **Timestamp:** Indicates the time at which the log entry was generated

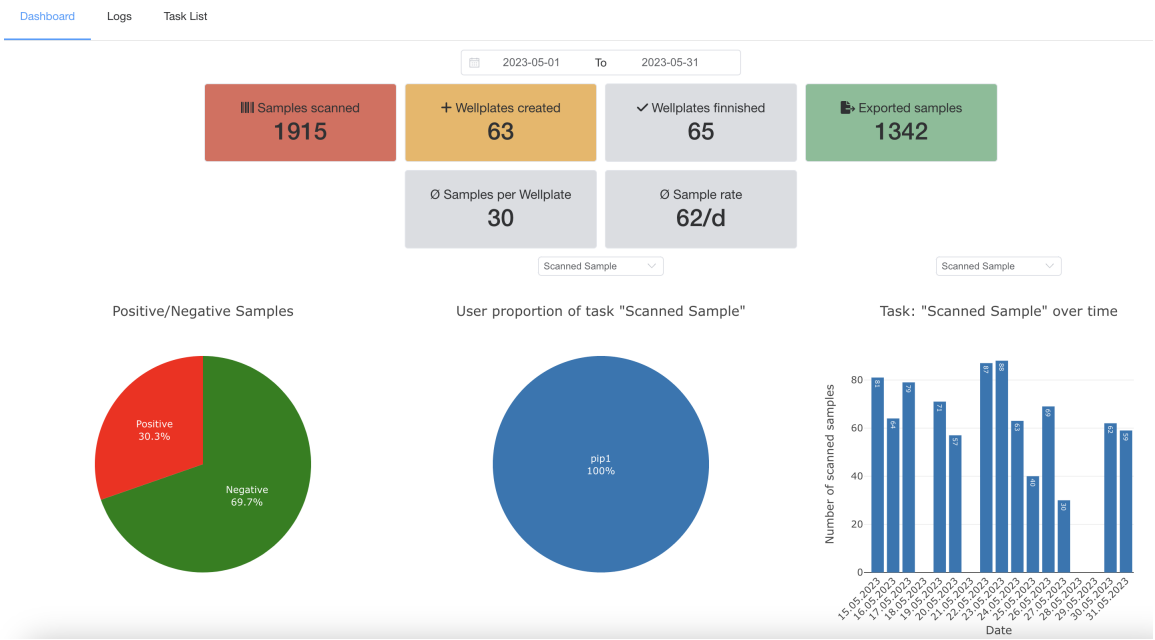


Figure 6.3: Dashboard overview

- **User:** Identifies the user who initiated the task execution
- **MAC Address:** Reflects the medium access control address (MAC) address of the client responsible for initiating the task execution

Except for the *Info* column, all columns can be sorted in either ascending or descending order. Furthermore, the *ID*, *Name*, and *User* columns support filtering, enabling more precise searches.

To access additional information regarding a specific log entry, users can click on the arrow adjacent to the checkbox, thereby expanding the selected row, as illustrated in Figure 6.7. This action reveals comprehensive details about the respective log entry, depending on the associated task type.

The entirety of the log data can be exported via the *Export* button, facilitating two distinct methods:

1. Manually select the data that is intended to be exported by selecting the checkboxes.
2. Directly clicking on the *Export* button, which prompts a popup window and allows users to customize the exported data according to their preferences. Adjustments may include specifying the desired time interval, selecting relevant task types for inclusion, and determining the export format (see Figure 6.5).

The exported data can be formatted in either JSON or CSV format. The export process begins after clicking the *Export* button located at the bottom right corner. The export functionality serves the primary purpose of advanced analysis that needs to be performed by laboratory personnel, which the provided tool does not offer.

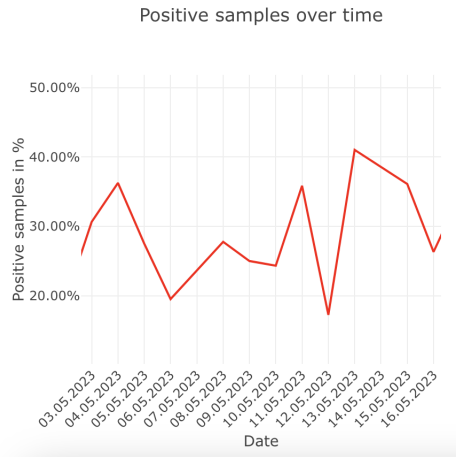


Figure 6.4: Line chart displaying the relative frequency of positive sample outcomes over time

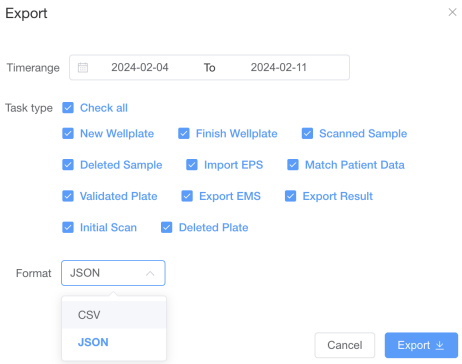


Figure 6.5: Log export view

Task List

The task list section shows waiting tasks for running process instances from the CPEE. These tasks are presently awaiting a response from the correlator. The tabulated data includes the following attributes:

- **ID:** Identifier of the task type.
- **Name:** Corresponding name of the task type.
- **Activity:** Specifies the precise task identifier from the CPEE, facilitating the identification of the awaiting task.
- **Instance:** Displays the CPEE process instance ID, encapsulated by a hyperlink directing to the relevant process instance within the CPEE.

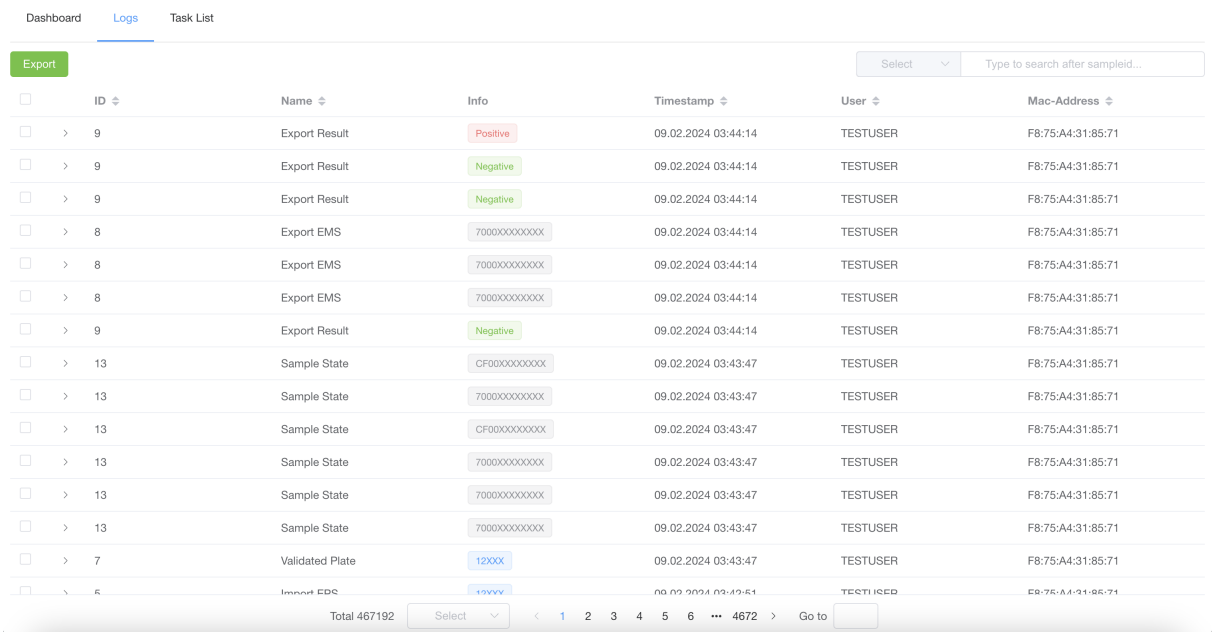


Figure 6.6: Visualization of collected logs

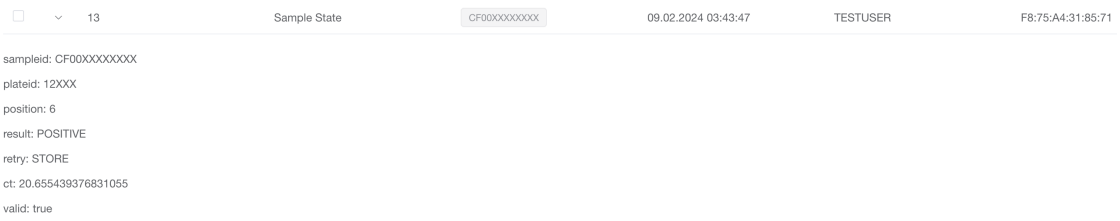


Figure 6.7: Detailed view of a log entry

- **Timestamp:** Indicates the time when the callback for the process instance was recorded in the database.

Again, all columns can be sorted in ascending or descending order, and the columns *ID* and *Task Name* can be filtered.

The task list serves the primary purpose of providing a quick overview of the waiting tasks in the corresponding process instances and quickly identifying potential issues, for instance, if certain tasks do not get called back.

Real-Time Updates

Both Logs and Task List views offer real-time data updates by utilizing server-sent events (SSE) [130]. This functionality means that if a new event has been stored in the database by the logging service, a push message is automatically sent to the frontend (i.e., client), which is then visualized in the

user interface. The same applies to newly created callbacks from the CPEE. Consequently, the laboratory personnel always have an up-to-date view of the data. The benefit of employing SSE lies in its integration within the HTML5 specification [131], which is supported natively by most modern web browsers. This factor eliminates the need for third-party libraries or plugins to implement real-time communication, thereby streamlining development processes and enhancing compatibility across different platforms and devices.

6.2.5 Logging Service

The logging service facilitates RESTful API interactions through the Express.js framework. This system primarily aggregates unprocessed event logs from the laboratory information management system, storing them in an unaltered state within the MongoDB database. These event logs are transmitted via HTTPS POST requests, encapsulating data in a JSON structure. Additionally, the service redirects all event logs to the backend service, which includes a correlator service. Users can access, filter, and organize the stored log entries utilizing HTTPS GET requests. By default, the system retrieves the most recent 1,000 entries, though this quantity is modifiable through the *limit* query parameter. Furthermore, the service offers options for categorizing the results based on time stamps, allowing for segregation into various temporal units such as years, months, weeks, and hours.

6.2.6 Reverse Proxy

The free and open-source version of Nginx has been utilized for deploying the reverse proxy. The reverse proxy efficiently directs incoming traffic toward the frontend, backend, and logging services. The frontend is conveniently accessible through the root path ("/"), while the backend service can be reached via the sub-path */backend*. It is worth noting that the Discord bot does not require accessibility through the reverse proxy, as its communication solely relies on secure WebSocket connections with Discord, as elaborated in the Discord bot subsection. The exact Nginx configuration file (i.e., nginx.conf) can be found on the Github repository in the nginx subfolder.

6.3 Testing Environment

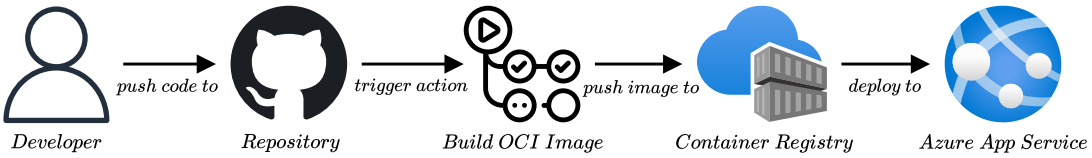


Figure 6.8: CI/CD deployment pipeline

This thesis established a testing environment to validate and examine recent software modifications before their implementation in the production environment. The testing environment exclusively employs test data, separate from production data.

The backend service, frontend, and reverse proxy were deployed on a Microsoft Azure Cloud instance of App Service. The App Service instance was running a Linux OS on a Basic (B2) SKU size with 2 virtual CPU cores, 3.5 GB of RAM, and 10 GB of storage. All three previously mentioned services were started as containers by using Docker Compose.

A MongoDB Atlas Free instance was utilized as the database for the backend service. Given that the code repository resides on GitHub, Continuous Integration/Continuous Delivery (CI/CD) practices were employed for the testing environment using GitHub Actions. The workflow in Figure 6.8 illustrates the configured CI/CD pipeline.

Upon committing and pushing the relevant code changes to the remote repository (i.e., the GitHub repository), the pipeline associated with the corresponding folder is triggered. Subsequently, the pipeline initiates the building of a Docker image encompassing all the necessary components. Once the image is successfully built, it is pushed to a container registry (in this case, Docker Hub was utilized). Docker Hub promptly detects the appearance of a new image and triggers the Azure App Service through a defined webhook. Afterward, the Azure App Service instance duplicates the new image to local storage and initiates the setup of the new container. This approach prevents the need for manual intervention and possesses the advantage of immediately reflecting changes in the testing environment.

6.4 Obtaining Raw Log Data

Given the in-house development of the LIS, modifications to its implementation are more easily achievable compared to an externally developed proprietary system. Consequently, an examination of the BPMN diagram, as presented in Figure 4.3, was conducted to assess traceability within the LIS, in collaboration with the designated LIS developer of the laboratory. The central question was as follows: Which events of the process can be tracked within the LIS? To achieve this, a systematic analysis of the LIS's source code was conducted to identify suitable events amenable to tracking.

The analysis yielded a catalog of 13 total events, which are enumerated and described in Table 6.1.

ID	Event Name	Body		Description
		Name	Datatype	
1	New Plate	PlateID	String	Gets triggered when a new 96-well plate is created by scanning the respective barcode command. Contains the generated plate ID.
2	Finish Plate	PlateID	String	Gets triggered when a 96-well plate is finished with pipetting by scanning the respective barcode command. Contains the plate ID.
3	Scanned Sample	SampleID PlateID	String String	Gets triggered when the barcode attached to a sample is scanned. Contains the corresponding sample and plate IDs including the position on the plate.

ID	Event Name	Body		Description
		Name	Datatype	
		Position	String	
4	Delete Sample	SampleID	String	Gets triggered when the delete sample barcode command is scanned, which cancels the last scanned sample of a current well plate. Contains the corresponding sample ID.
5	Import EPS	PlateID	String	Gets triggered when the .eps file from the thermocycler is imported into the LIS. Contains the corresponding sample ID.
6	Match Patient	SampleID	String	Gets triggered when the sample has been successfully matched with the corresponding patient data. Contains the sample ID.
7	Validated Plate	PlateID	String	Gets triggered when the 96-well plate has been successfully validated by the laboratory personnel. Contains the plate ID.
8	Export EMS	SampleID	String	Gets triggered when the sample has been successfully exported to the Epidemiologisches Meldesystem (EMS). Contains the sample ID.
9	Export Result	SampleID Completed Result	String Boolean Enum	Gets triggered when the export routine is executed. Contains the sample ID; the result, which can be either positive or negative; and the completed tag, which indicates whether the export was successful.
10	Export State	SampleID Success Service	String Boolean Enum	Gets triggered after the execution of the export routine. Returns the export state of each export service indicating whether the export has failed.
11	Initial Scan	SampleID	String	Gets triggered when the sample is initially scanned directly after it arrives at the laboratory but before the process starts. Contains the sample ID.

ID	Event Name	Body		Description
		Name	Datatype	
12	Deleted Plate	PlateID	String	Gets triggered when a 96-well plate is canceled by scanning the respective barcode command. Contains the well plate ID.
13	Sample State	SampleID PlateID Position Result Retry CT Valid	String String String Enum Enum Double Boolean	Gets triggered after validating the full well plate. Returns the full state of a sample containing information regarding plate, position, result, *cycle threshold* (in short C_t) value, retry state, and whether the result is valid.

Table 6.1: Catalog of identified events from the LIS

The initial purpose of the *Export State* event was to gain a more granular view of the *Export Result* event, indicating which export routines were successful and which failed. However, this event type was never implemented by the LIS developer due to the unpredictable complexity of the implementation.

Considering the *Sample State* event, the valid tag determines whether the sample (result) is exportable. If the same sample is tested on multiple wells on the same plate, only one sample result can be exported by setting the valid tag. The retry type will be automatically determined after matching the imported.eps file with the corresponding well plate and can be manually adjusted afterward. The following retry types are possible:

- **DEVIANT:** This type is set when the sample result does not seem to be valid or there is a paradoxical sample history. It may manifest, for instance, in divergent results for identical samples, characterized by markedly distinct C_t values. In such cases, laboratory staff are tasked with determining whether to repeat the testing process or to exclude the sample by manually selecting the retry state. Crucially, any deviant samples within a well plate necessitate manual adjustment before the overall plate can be validated. Consequently, a sample cannot simultaneously be deviant and valid.
- **NONE:** The sample will not be analyzed again, which is the default case.
- **PFIRST:** The sample is first-time positive.
- **PMULT:** The sample is part of a multiwell.
- **RT1:** Retry 1, which is the first retry of a sample.
- **RT2:** Retry 2, which is the second retry of a sample.
- **RT3:** Retry 3, which is the third retry of a sample.

- **SELECTED:** The sample is manually selected for retesting.
- **STORE:** The physical sample is stored for future investigation.

The following result types are possible:

- **NEGATIVE:** The sample tests negative for SARS-CoV-2
- **POSITIVE:** The sample tests positive for SARS-CoV-2
- **IPC:** The positive control sample is invalid.
- **NOTSET:** No result is available, which happens, for instance, if the well is empty. This type is the default case.

An event listener within the LIS was implemented in this thesis. The event listener gets triggered after the events described in Table 6.1. Upon activation, the event listener initiates the generation of an HTTPS POST request, which is sent to the REST API of the logging service. The details of the logging service are described in Section 6.2.5. The payload of this POST request adheres to a JSON-based structure, conforming to the subsequent generic schema (see Listing 6.1). An example of an event massage is shown in Listing 6.2.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "JSON Schema for LIS events",
  "type": "object",
  "properties": {
    "id": {
      "type": "string",
      "description": "Unique event ID"
    },
    "name": {
      "type": "string",
      "description": "Event name"
    },
    "user": {
      "type": "string",
      "description": "Username of the client where the event was
    triggered"
    },
    "macaddress": {
      "type": "string",
      "description": "MAC address of the client where the event was
    triggered"
    },
    "timestamp": {
      "type": "string",
      "format": "date-time",
      "default": "current date",
      "description": "Current timestamp of the event"
    },
    "body": {
```

```
      "type": "object",
      "description": "Body containing the payload of the event"
    }
  },
  "required": [
    "id",
    "name",
    "user",
    "macadress",
    "body"
  ]
}
```

Listing 6.1: JSON schema for events sent from the LIS

The schema in Listing 6.1 defines several properties that the message must or can have:

- **id:** A unique identifier for the event (see ID column in Table 6.1).
- **name:** The name of the event (see Name column in Table 6.1).
- **user:** The username of the LIS client where the event was triggered.
- **macaddress:** The MAC address of the LIS client where the event was triggered.
- **timestamp:** A string that represents the date and time of the event's occurrence. It is expected to follow the ISO 8601 format [132], and its default value is the current date, which means that if no timestamp is provided, the current date is added by the logging service.
- **body:** An object that contains the payload of the event (see Body column in Table 6.1).

```
{
  "id": "1",
  "name": "New Wellplate",
  "user": "user1",
  "macaddress": "00:00:00:00:00:1F",
  "body": {
    "plateid": "1234"
  }
}
```

Listing 6.2: Example event sent from the LIS. Note: The timestamp property is missing since it is optional

6.5 Data Visualization and Reporting

Since all relevant event data is being acquired and archived by the logging service, the subsequent phase involves the representation of the accumulated data. It is important to present the collected data coherently and beneficially for the personnel. Data visualization and reporting methodologies facilitate the conversion of complex and large data sets into comprehensible visual formats.

Leveraging graphical representations such as charts and graphs allows easier identification of trends in data, such as fluctuations in positive sample numbers. The visualization allows for the comparison of data across different times and can be used to monitor the testing process in the laboratory. These insights can highlight issues in sample collection, testing throughput, and testing accuracy.

Consequently, a web-based application has been developed to offer detailed insights into the collected event data through interactive charts, graphs, and tabular presentations. The data can be dynamically manipulated through filtering and sorting functionalities. For a comprehensive understanding of the implementation, refer to Section 6.2.4.

6.6 Digital Workflow Representation

Considering the identified events in Section 6.4 in comparison with the initial testing process in Figure 4.3 and the discussed evaluation regarding tracking and automation capabilities in Section 5.4, it is apparent that the available event data covers only parts of the process.

The import flow is outside this thesis's purview due to data protection reasons (see Section 5.4.6). The automatic pipetting robot and thermocycler (i.e., 384 well plate subprocess) cannot be directly tracked due to missing network connectivity or API access availability. Potential tracking approaches are examined in Section 5.4; however, a concrete implementation would go beyond the scope of this work. Therefore, a simplified process model based on Figure 4.3 is derived.

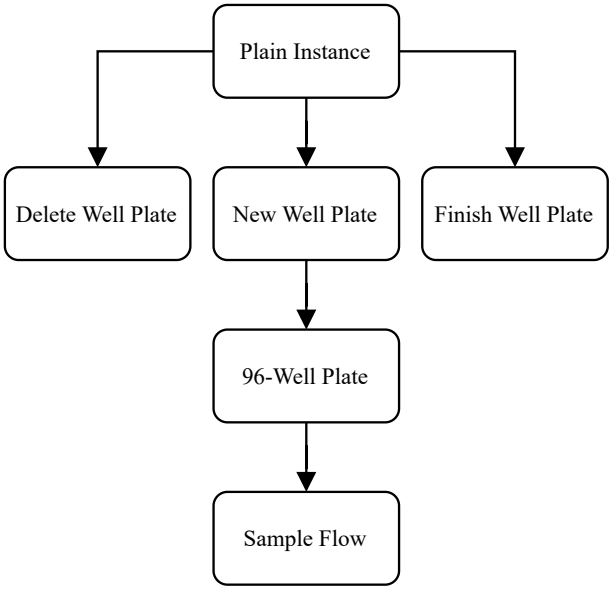


Figure 6.9: Process instance spawn hierarchy

On a high-level view, the process model is divided into six different hierarchically organized processes, each with distinct functions and objectives. The hierarchy is based on the initial process model in Figure 4.3. All well plate instances are spawned originating from the plain instance, which,

in turn, spawn the respective sample instances. A graphical representation of the process instance spawn hierarchy is depicted in Figure 6.9, illustrating the subprocesses as follows:

1. **Plain Instance:** The plain instance embodies the core mechanism for governing the behavior of the process instances, meaning it acts as the managing process. This initial instance is responsible for supervising subsequent process instances and establishes the framework for administering and coordinating complex processes. The plain instance spawns the following subprocesses: New Well Plate, Delete Well Plate and Finish Well Plate.

2. **New Well Plate:** The new well plate generates new instances of well plates, specifically referring to instances of the 96-well plates.

3. **Delete Well Plate:** This process gets triggered when a well plate is canceled, leading to the termination of the corresponding 96-well plate instance along with all associated sample instances.

4. **Finish Well Plate:** This process dispatches a message event to the corresponding 96-well plate instance upon completion of pipetting.

5. **96-Well Plate:** This process represents the physical well plate and is responsible for initiating the sample flow subprocess for each contained sample.

6. **Sample Flow:** Sample flow represents an individual sample, encompassing all relevant data such as results, C_t value, position, and retry status.

The subsequent sections further examine the process models. The compliance checking relevant process tasks are separately discussed in Section 6.7.

6.6.1 Plain Instance

ID	Name	Executor	Description
A1, A2, A3	Spawn Delete/Finish/New Well Plate Flow	Automatic	Spawns the Delete/Finish/New Well Plate subprocesses and initializes a data element containing the plain instances UUID.
A4, A5, A6	Receive finishedids, createdids, deletedids	Automatic	Listens on the notify plain instance message event (Task IDs=B5,C5,D5) from the Delete/Finish/New Well Plate subprocess instances by utilizing the UUID plus a character as an identifier. The received plate ID from the message is stored in an array.

Table 6.2: Workflow task description for Plain Instance

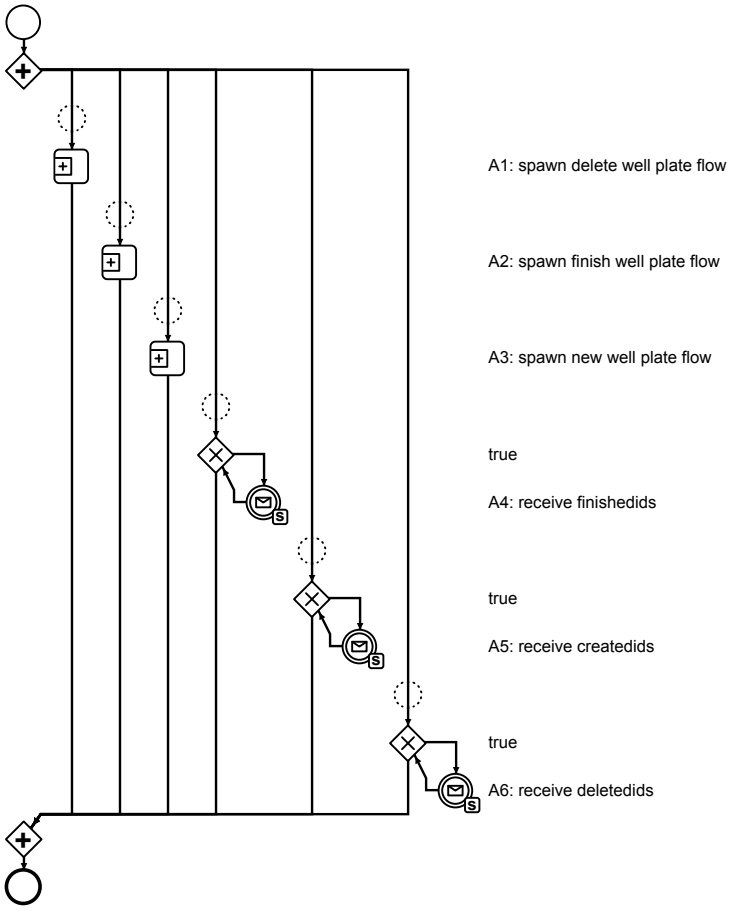


Figure 6.10: Plain Instance

The process model consists of a single parallel gateway encompassing six branches. The first three branches contain a single subprocess task, which spawns the Delete, Finish, and New Well Plate processes respectively. These subprocesses are initiated with a data element containing the UUID of the plain instance process. The three succeeding branches consist of a loop incorporating a receive message event that listens on the plain instances of UUID, intending to capture the created, finished, and deleted well plates from the subprocesses. To distinguish between these subprocesses during message transmission, the UUID is expanded by a distinct character specific to each subprocess. The outputs from the creation, completion, and deletion subprocesses are collected in separate array data elements. The data can be used for compliance-checking purposes. Compared to the initial process model in Figure 4.3, the plain instance represents an additional abstracted higher-level process dimension that aims to control the discrete well plate creation, completion, and deletion processes.

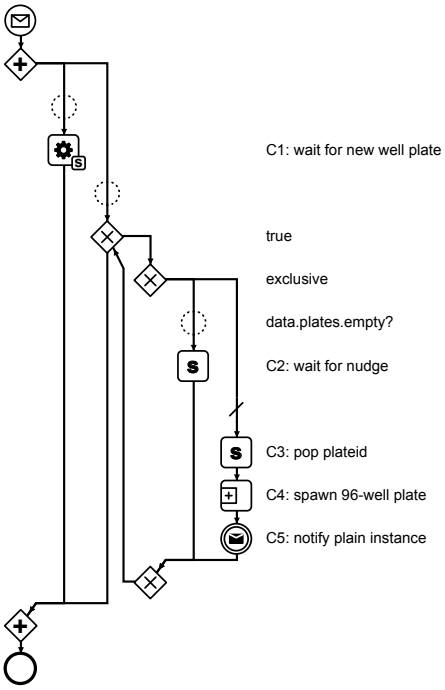


Figure 6.11: New Well Plate Workflow

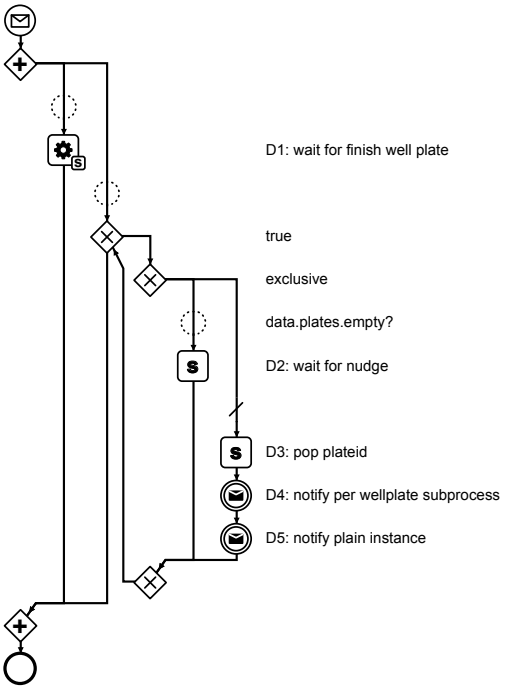


Figure 6.12: Finish Well Plate Workflow

6.6.2 Delete/Finish/New Well Plate

All three subprocesses share a similar process structure and are, therefore, summarized in one section. Notably, these process models entail tasks B1, C1, and D1, which wait for an update callback, meaning that these tasks never finish. The plate is stored in an array upon receiving an update callback, prompting the activation of a nudge signal, subsequently triggering the concurrent execution of awaiting tasks B2, C2, and D2. An exclusive gateway validates the array's contents, extracting the corresponding plate into a distinct variable if it is not empty. This design strategy facilitates the non-blocking handling of multiple well plates simultaneously. However, the CPEE restricts each process instance to a single nudge signal, necessitating the division of the Delete/Finish/New Well Plate processes into three distinct process models. Table 6.3 provides a detailed description of each workflow task.

6.6.3 96-Well Plate

The process initiates with a parallel gateway consisting of three branches (see Figure 6.14). The spawning of new samples occurs as a subprocess within the first branch, facilitated by two exclusive gateways. The outer gateway establishes a loop, iterating until the well plate reaches completion, signaled by the "data.finished" variable transitioning to true upon receipt of the "finished plate" message event from the Finish Well Plate process instance (referenced as task E7) in the third parallel branch. The inner gateway verifies the "data.lastsample" variable's non-nil status to distinguish the callback return value of task E1: wait for sample. This design ensures that even after

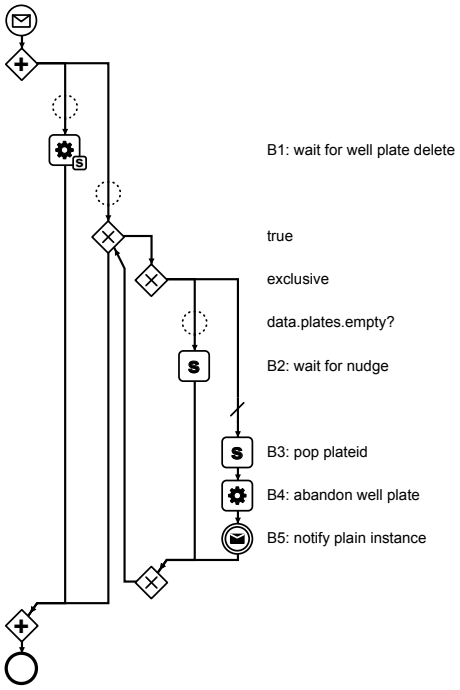


Figure 6.13: Delete Well Plate Workflow

the well plate's completion, tasks E1 and E3: wait for sample remain in a callback state, preventing the parallel gateway from merging and synchronizing the workflow. Consequently, tasks E1 and E3 necessitate manual callback execution, initiated by task E8 for E1 and E9 for E3 in the third branch.

The second branch addresses scenarios where a previously spawned sample is canceled (i.e., deleted), mirroring the structure of the first branch. The inner gateway integrates a tail-controlled loop (i.e., the condition for the loop is evaluated after execution), employing a script task to search the sample array for the deleted sample. The loop terminates upon successfully identifying the sample planned for deletion within the array. This approach considers process instances where the delete callback may precede the successful spawning of the sample due to the asynchronous processing mechanism of the correlator. Subsequently, the corresponding running sample process instance is stopped and abandoned upon loop termination.

Following the parallel gateway, the process instance awaits .eps file import and plate validation. Upon successful validation, a message event is broadcasted to all associated child sample process instances, indicating that the well plate has been verified and is now ready for subsequent processing on a sample-level basis.

6.6.4 Sample Flow

The process in Figure 6.15 commences with matching the sample to the corresponding patient data (F1). If the well plate position equals 1, containing the positive control sample, no patient

ID	Name	Executor	Description
B1, C1, D1	Wait for Well Plate Delete, Wait for New Well Plate, Wait for Finish Well Plate	Human	Waits for an update callback for the events New Plate (ID=1), Finish Plate (ID=2), and Deleted Plate (ID=12). Stores the plate ID in an array and triggers a nudge.
B2, C2, D2	Wait for Nudge	Automatic	Waits for the nudge signal.
B3, C3, D3	Pop Plate ID	Automatic	Removes the last plate ID from the array and stores it in a separate variable.
B4	Abandon Well Plate	Automatic	Sends the plate ID to the backend endpoint which stops and abandons the corresponding well plate process instance, including all respective sample instances.
C4	Spawn 96-Well Plate	Automatic	Spawns a new 96-Well Plate subprocess based on the plate ID previously saved in the variable.
D4	Notify per Well Plate Subprocess	Automatic	Sends the finished plate message to the corresponding well plate process instance by utilizing the plate ID as an identifier.
B5, C5, D5	Notify Plain Instance	Automatic	Sends the plate ID to the plain instance by utilizing the plain instances UUID and a distinct character as an identifier.

Table 6.3: Workflow task description for Delete/Finish/New Well Plate

matching is necessary. Consequently, task F1 is only executed when the sample position is greater than 1, as indicated by the exclusive gateway surrounding F1. The sample process awaits validation of the parent well plate by listening for a message event (F2). Following successful plate validation (E12), the sample receives all relevant data from the sample state event, including the result, C_t value, retry status, and validity. The motivation behind the parallel separation of matching patient data and plate validation with sample state is to handle the possibility of patient data arriving after sample analysis. However, the corresponding patient data must be available for a sample to be exportable, as enforced by the closing parallel gateway. The valid property of the sample state is checked once both patient and sample data are accessible. No export routine is initiated if the same is deemed invalid. Conversely, if the sample is valid, the process instance awaits the export result task (F5) and, if it is not, it awaits the positive control at position 1, the EMS export task (F4). Both tasks are executed concurrently, as they operate independently from each other.

6.7 Automatic Compliance Checking

Following the implementation of the testing process through the use of the CPEE, the concluding phase involves the analysis and expansion of the process with compliance-checking capabilities. As described in Section 2.2, compliance checking can be classified into two categories, namely *backward* and *forward compliance checking*. In both categories, compliance can be checked either *inline* or *externally*. Inline verification means that the compliance rules are embedded directly into the process model. Conversely, external verification is conducted using an independent rule engine that operates according to a predefined process-separated set of rules known as the rule base. The segregation of rules from the process model encourages the reusability and manageability of business rules. This approach not only streamlines the application of these rules across various processes but also facilitates their maintenance and adaptation in dynamic business environments.

A rule refers to a formalized regulation or requirement to which a specific process or activity within an organization must adhere. This formalization is often accomplished through the creation of a domain-specific language (DSL), characterized by a unique syntax and semantics. Once developed, the DSL is processed by parsing it into an abstract syntax tree (AST), which the rule engine subsequently evaluates. This procedure is typically implemented by using a high-level programming language such as Python or Java, leveraging libraries such as ANTLR [133], or employing custom parser code to facilitate the translation. The rule base can be implemented in two ways, namely as a universal rule base that operates concurrently with the process instances or as annotations that are embedded directly within the process model. The latter approach defines the expected behaviors of specific process instances, thereby providing a more tailored application of the rules. Such annotations can be applied at the process or task levels (see Figure 6.16).

6.7.1 Example Rule Schema

```
(* A Compliance Rule consists of a match clause, followed by a condition,
     an action, and an optional else action *)
compliance_rule = 'RULE', rule_id, match_clause, condition_clause,
    action_clause, [ else_action_clause ] ;

rule_id = identifier ;

(* Match Clause - where process elements are specified *)
match_clause = 'MATCH', match_expressions ;

match_expressions = match_expression , { ',' , match_expression } ;
match_expression = process_element, [ match_condition ] ;
process_element = element_path, { '->', element_path } ;
element_path = identifier, [ array_index ] ;  (* Allow array access like
    'Data[0]' *)
array_index = '[', number, ']' ;  (* Defines the syntax for array
    indexing *)

(* Match Conditions are comparisons against values or other elements *)
```

```
match_condition = comparison_operator, ( value | process_element ) ;

(* Condition Clause - logical expressions involving process elements *)
condition_clause = 'CONDITION', logical_expression ;

(* Logical Expressions support AND, OR, NOT operations *)
logical_expression = logical_term , { ('AND' | 'OR' | 'XOR'), logical_term } ;
logical_term = [ 'NOT' ], comparison_expression ;

(* Comparison Expressions for relational operations *)
comparison_expression = expression , { comparison_operator, expression } ;
comparison_operator = '<' | '<=' | '>' | '>=' | '=' | '!=' | 'LIKE' | 'IN' ;

(* Arithmetic expressions *)
expression = term , { ('+' | '-' | '*' | '/' | '%' | '^'), term } ;
term = factor , { comparison_operator, factor } ;
factor = this_access | value | '(', logical_expression, ')' ;

(* This Access - access to the current context of the Match Clause *)
this_access = 'THIS' , { '->', process_element } ;

(* Action Clause - specifies what action to take when conditions are met *)
action_clause = 'ACTION', action_spec ;

(* Optional Else Action Clause - specifies what action to take when conditions are not met *)
else_action_clause = 'ELSE ACTION', action_spec ;

(* Actions can be either commands or function calls *)
action_spec = command | function_call ;

command = ('ACK' | 'SKIP' | 'START' | 'STOP' | 'LOG' '(' string_literal ')' ) ;

(* Function Call - includes function name and optionally arguments *)
function_call = identifier, '(', [ arguments ], ')' ;
arguments = expression , { ',', expression } ; (* Arguments can be any expression, including identifiers or values *)

(* Utility definitions for basic elements like identifiers and numbers *)
identifier = letter , { letter | digit | '_' } ;
value = string_literal | number ;
string_literal = '"', { all_characters_except_quote }, '"' ;
number = [ '-' ], digit , { digit } ;
letter = 'A' | 'B' | ... | 'Z' | 'a' | 'b' | ... | 'z' ;
digit = '0' | '1' | ... | '9' ;
all_characters_except_quote = ? all characters except '"' ? ;
```

Listing 6.3: Compliance rule schema

The rule schema provided in this work is based on the match–condition–action principle, which was introduced by Koenig et al. [134]. In general, a business rule consists of three parts, namely *match*, *condition*, and *action*. The *match* component defines the criteria for associating events with the process activities specified in the rule, that is, identifying the attribute set upon which a successful matching can be established. After successful alignment with the *match* pattern, a determination regarding the instance's compliance is made in the *condition* part. Based on the outcome of the *condition*, corresponding *actions* are determined, and a distinction can be made between the fulfillment and non-fulfillment of the condition.

Structure of a Compliance Rule

The DSL provided in Listing 6.3 describes the syntax for defining compliance rules within a process-oriented framework. Expressed by using the extended Backus–Naur form (EBNF) [135], the DSL supports arithmetic and comparison operations within expressions, which are essential for performing checks and transformations on data elements within the process. This capability is fundamental for rules that dynamically evaluate or manipulate numerical values or other data types.

Each compliance rule ($compliance\_rule$) is defined with a series of structured clauses, starting with a unique identifier ($rule\_id$), followed by a $match\_clause$, a $condition\_clause$, an $action\_clause$, and an optionally defined $else\_action\_clause$. This hierarchical structure allows the rule to specify conditions and actions linked to particular process elements, enhancing rule clarity and enforceability.

Match Clause

The $match\_clause$ serves as the basis for defining the activities of a process for which the rule applies. It comprises one or more $match\_expressions$, where each $match\_expression$ can optionally include a $match\_condition$. These conditions are critical for targeting specific attributes or states of process elements, facilitating precise rule application. The process and activity matching are based on the CPEE activity lifecycle model [136]. This thesis uses a hierarchical path structure similar to the work of Koenig et al. [134] to identify specific process elements, such as data elements or activity labels. The element path is constructed by using identifiers that can represent fields, properties, or other relevant data points. The identifiers are concatenated using the $\rightarrow$ symbol to navigate through nested structures or hierarchies within the event. For example, a path might be expressed as $dataelements \rightarrow change \rightarrow content \rightarrow values \rightarrow plateid$ to access the value from the data element $plateid$. It is also possible to access data elements when they are part of an array by using the array index. This approach allows matching against not only label equivalence but also attribute equivalence [137]. Upon successful matching the corresponding activity or process instance is mapped to the *this* object, which enables the reference to the current context within the rule. It is important to mention that the event stream (runtime) and the logs (ex-post) from the CPEE have a distinct data structure, and, therefore, the matching path needs to be adjusted depending on the approach employed.

Condition Clause

This clause is checked after the match clause is fulfilled and contains logical expressions that determine the conditions under which the specified actions should be triggered. Using logical operators (*AND*, *OR*, *XOR*, *NOT*) in $logical\_expressions$ enables complex, multi-faceted decision-making criteria to be established, reflecting real-world decision scenarios.

Action and Else Action Clauses

Action and else action clauses define the actions to be executed when the conditions are met or not met, respectively. Actions can be straightforward commands or more complex function calls, allowing for dynamic responses based on the process state and the outcomes of the condition evaluations. The default action commands *ACK*, *SKIP*, *START* and *STOP* refer to the CPEEs votes concept [136] which allows to influence the execution of process instances during runtime. The *LOG* action command enables the logging of specific information when the conditions of the compliance rule are met, which might be useful for maintaining an audit trail, debugging rule implementations, or providing runtime feedback on the state of process compliance.

Example Compliance Rules

```
RULE ValidateSampleState
MATCH dataelements->change->changed[0] = "state"
CONDITION
    (THIS->dataelements->change->content->values->state->retry = "DEVIANT
    ") AND (THIS->dataelements->change->content->values->state->valid = "
    TRUE") AND (THIS->dataelements->change->content->values->state->
    position > 1)
ACTION NOTIFY("Invalid sample state detected!")
ELSE ACTION LOG("Sample state OK")
```

Listing 6.4: Example Compliance Rule 1

Compliance Rule 1 in Listing 6.4 is triggered when the sample state data element from a sample flow instance is modified. In the condition clause, the rule evaluates whether the retry type equals *DEVIANT* and is considered valid at the same time. The rule highlights a scenario where an anomalous sample state is still considered valid by the system. Upon satisfying these conditions, the action clause is activated, which involves executing a notification function with the message "Invalid sample state detected!". This notification alerts stakeholders or systems that the sample state represents an anomaly that requires attention, despite the validity assigned to it. Alternatively, if the conditions are not met, the else action clause is executed, and the system logs the message "Sample state OK". This action records that the state is within acceptable limits, confirming the absence of any critical discrepancies in the data.

```
RULE ProcessTimeLimit
MATCH state->change->state = "finished"
CONDITION DURATION(THIS) >= 24
ACTION NOTIFY("Process duration exceeded 24 hours")
```

Listing 6.5: Example Compliance Rule 2

Compliance Rule 2 in Listing 6.5 is activated when the process instance execution lifecycle transitions to the state *finished*. This transition is essential as it denotes the completion of a process instance. The condition clause introduces a temporal constraint, utilizing a function, *DURATION*, to calculate the total time elapsed since the initiation of the process. The *THIS* object represents the matched process instance in the current context. If the duration of the process meets or surpasses 24 hours, the subsequent action involves issuing a notification to the relevant stakeholders. This compliance rule enforces time-related constraints, ensuring that process instances do not exceed an allowable time frame without appropriate notifications being triggered.

```
RULE SkipMatchPatientData
MATCH activity->calling->content->activity = "a4"
CONDITION THIS->data->sample->position > 1
ACTION ACK
ELSE ACTION SKIP
```

Listing 6.6: Example Compliance Rule 3

Compliance Rule 3 in Listing 6.6 matches against the calling activity *a4*, which refers to the *F1: match patient data* task in the sample flow. The condition checks whether the sample position is greater than 1, indicating that the sample does not serve as a positive control. If the condition is satisfied, the action clause sends an acknowledgment (*ACK*) vote to resume with process execution. Conversely, the else action clause is configured to execute the *SKIP* vote to bypass the matched activity. This rule can be used to omit an explicit representation of the condition in the process model by skipping tasks thus reducing complexity.

6.7.2 Forward Compliance Checking

Forward compliance checking can be divided into two approaches:

Design-Time Compliance Checking

As stated by Ly et al. [138] a process model is considered compliant during design time when it permits only the execution of instances that do not breach certain rules or standards (semantic constraints). Therefore, by securing compliance at the model level, it is implicitly assured that the resulting process instances will also be compliant. Design time is preventive, aiming to build compliance into the process from the start.

The process models designed in this work can be considered structurally compliant since they are built with the XML-based domain-specific language of the CPEE, which adheres to the BPMN standard. If the structure is violated, then no process execution is possible on the CPEE.

Run-Time Compliance Checking

According to the work of Kharbili et al. [48] the compliance of the model is enforced during process execution by integrating atomic compliance assertions in process models or specific tasks. These assertions are then utilized by compliance checking engines for verification or are employed during subsequent stages of execution. From this perspective, regulatory requirements can be integrated directly into the models, as exemplified by control flow attributes such as anti-patterns [139] that

aim to enhance the quality of processes. Alternatively, these constraints may necessitate real-time data for enforcing quality assertions during execution. Runtime compliance checking is more reactive. It involves monitoring ongoing operations to detect and address violations as they occur.

Inline

Considering the process models defined in Section 6.6, several inline runtime compliance checking opportunities can be identified.

The compliance rule mandates that specific process instances or tasks are completed within a predefined duration. In the realm of laboratory testing, the well plate and samples must be analyzed within a designated timeframe to prevent the expiration of reagents. Failing to do so compromises the integrity of the results, necessitating the repetition of the entire testing process, including sample extraction. This redundancy not only generates additional costs but also consumes more resources. To mitigate these issues, the process models for the well plate and samples incorporate a parallel branch that tracks the elapsed time. If the designated interval elapses without task completion, then laboratory personnel receive a notification to take the necessary corrective actions.

Figure 6.15 illustrates the implementation of this compliance rule in the process models. Upon initiating a new process instance, Task G2 is activated in the parallel branch is activated to set a timeout, accomplished through an HTTP POST request to the internally developed */timeout* endpoint. This endpoint responds with a callback flag, initiating a suspended state for the timeout task G2, which can resume under two conditions:

1. If the timeout exceeds its predefined interval, then the endpoint returns a positive response, triggering Task G3: Send notification. This task triggers the discord bot, which sends a formatted message incorporating parameters such as log level and content, potentially enhanced with markdown (e.g. including hyperlinks), thereby informing laboratory staff of the time exceedance.

2. If the process concludes before surpassing the time interval, it is imperative to first address the waiting timeout task before reaching the process's end event. This is managed by executing Task G1: callback timeout, which sends an HTTP POST request to the */timeout* endpoint with a termination flag, instructing the endpoint to resume the timeout task. The callback message carries a nil flag to prevent the activation of the send notification task unless the time interval is exceeded.

Figure 6.17 depicts a sample notification issued by the bot, with the sample and well plate IDs formatted as hyperlinks directing to their respective process instances. Figure 6.18 demonstrates the configuration of the CPEE task responsible for sending notifications.

The primary benefit of adopting a process model-centric approach lies in the enhanced customizability of business logic compared to traditional programmatic strategies. Such models offer superior flexibility, efficiency, and accessibility. Facilitating adjustments at the process model level notably improves communication between business stakeholders and information technology

(IT) teams. Process models typically employ visual representations that are more comprehensible to non-technical stakeholders, simplifying discussions and enhancing collaborative engagement. Consequently, this approach empowers business professionals to actively participate in both the development and modification of business logic, ensuring that the systems developed are better aligned with strategic business needs and objectives.

Moreover, business environments are inherently dynamic, necessitating the ability to swiftly adapt processes and workflows in response to evolving market conditions or regulatory changes. Process models are instrumental in enabling rapid modifications, thereby obviating the need for extensive software redevelopment. Moreover, adjustments made through process models generally require less technical expertise than direct code alterations, thus reducing reliance on highly specialized and, often, expensive software engineering resources. Consequently, maintaining and updating business logic via process models can be less financially burdensome, as the process models decrease the volume of code that needs to be rewritten and retested.

For instance, consider the compliance rule depicted in Figure 6.19, which incorporates a loop to send notifications at defined intervals. This method is designed to consistently remind laboratory personnel to resolve the identified compliance rule violation in each process instance, thereby ensuring continuous adherence to regulatory standards.

External

External runtime compliance checking can be achieved by subscribing to the data stream interface of the corresponding process instance from the CPEE. The data stream interface offers a publish/-subscribe (PUB/SUB) mechanism that enables targeted subscriptions to particular subsets of events. Each subscription must include an optional endpoint for event push and a defined set of topics and events. In subscriptions where the endpoint is not specified, it is presumed that events are sent via SSE. The events are formatted according to the JSON schema. Subsequently, these events are collected and filtered by the endpoint. The rules are then analyzed by the rule engine by comparing the incoming events against a pre-established rule base. The detection of any discrepancies or violations against set compliance rules triggers the initiation of corrective measures. These measures may include notifying relevant personnel or influencing the process execution (i.e. execution shaping) by utilizing the CPEE voting mechanism [37, 136]. For these so-called votes, the process engine actively awaits responses from each engaged external service before proceeding with further actions. Potential interventions might involve postponing the implicated activity (callback) or immediately starting and stopping the process instance.

Considering the sample flow, possible compliance rules are presented in Listings 6.4, 6.5, and 6.6, which can be verified during runtime by monitoring subscribed topics in the event stream.

6.7.3 Backward Compliance Checking

Inline

Inline backward compliance checking can be effectively implemented by integrating a dedicated task within the process model. This task is responsible for ensuring that preceding tasks conform to established compliance rules. In the context of laboratory testing, a task can be inserted to

verify that the duration between the import of the .eps file and plate validation does not surpass a three-hour limit. This rule is seamlessly incorporated into the well plate process model by adding a task after the "E11: Validate Plate" task. Before execution of the "E10: Import .eps File" task and following the successful completion of the "E11: Validate Plate" task, the start and end timestamps are recorded in a distinct variable. The inserted task calculates the time elapsed between these timestamps, which is then compared against the three-hour compliance threshold. If this threshold is exceeded, a notification is sent to inform the laboratory personnel. Figure 6.20 illustrates the modified well plate process model that includes the previously mentioned inline compliance mechanism. The start and stop logic is executed within the service task output handling functions *Prepare* and *Finalize*.

External

The concept of external backward compliance checking is realized through the post-process analysis of execution logs. The CPEE provides an integral logging service that subscribes to designated topics and ensures their persistent storage. These logs are structured following the XES format and adhere to the YAML schema. Retrieval of these logs is facilitated through an HTTP GET request targeted at */logs/{instance-uuid}*. Subsequently, these logs undergo a systematic parsing and evaluation process based on a predefined set of rules. Appropriate remedial actions can be implemented in instances where discrepancies are identified. Utilizing the same scenario from external runtime compliance checking, the sample state can be determined by extracting the data element from the lifecycle transition activity *Receiving* associated with task *F3: receive sample state*. Additionally, by comparing the provided timestamps, it is possible to identify violations regarding the duration of time intervals, either at the process or task level. Such example rules have already been discussed in Section 6.7.1.

6.8 Challenges

In the early stages of development, the testing environment was running on the Free T1 SKU in Azure, which contained 1 GB of RAM. One of the first problems that occurred during testing the first prototype was that the services randomly stopped. Examining the log files revealed that it appeared that the containers were running out of RAM. Consequently, the SKU was upgraded to Basic B2, which contains 3.5 GB of RAM.

During testing with the CPEE, some spawned sample instances randomly stopped when they called the backend service running on Azure App Service. By investigating the CPEE logs, the following output was provided:

```
event:
  concept:instance: 11262
  concept:name: Match patient data
  concept:endpoint: https://greschner.azurewebsites.net/backend/corr
  id:id: a4
  cpee:activity: a4
  cpee:activity_uuid: cea171d9b312d1381f7ffdcb72581589
  cpee:instance: b6a9199a-5a96-455e-b25b-c595749ca3b0
```

```
9  lifecycle:transition: unknown
10 cpee:lifecycle:transition: activity/receiving
11 raw:
12 - name: error
13   mimetype: application/json
14   data: '{"status":0,"error":null}'
15 time:timestamp: '2023-04-09T13:37:15.763+02:00'
```

Listing 6.7: Process instance log pulled from CPEE

The backend service provided a response to the CPEE, containing a JSON-formatted text indicating a status of 0 and a null error (see Line 14 in Listing 6.7), resulting in the termination of the spawned sample instance. However, an examination of the log files in the Azure App Service revealed the absence of any error messages, both in the reverse proxy and the backend service. The messages sent by the CPEE did not appear to reach the backend service, as indicated by the reverse proxy logs. This finding ruled out the hosted services as the underlying cause, further complicated by the inability to manually reproduce the behavior. Internet research showed that Azure App Service encounters difficulties with DNS resolution when using the default domain name "*.azurewebsites.net." The resolution to this problem involved obtaining a custom domain name and registering it with the corresponding app service.

When the "Delete Sample" task was called back with the same sample (i.e., same position) multiple times, the CPEE well plate instance was stopped. The reason for this behavior was that the instance sends "stop" and "abandon" to the same sample process instance multiple times, causing an error since the instance had already been stopped and abandoned. To address this issue, the sample designated for abandonment was eliminated from the samples array, resulting in a nil return value within the "E4: check for sample" script task. This solution was implemented in the "E6: abandon spawned sample" task.

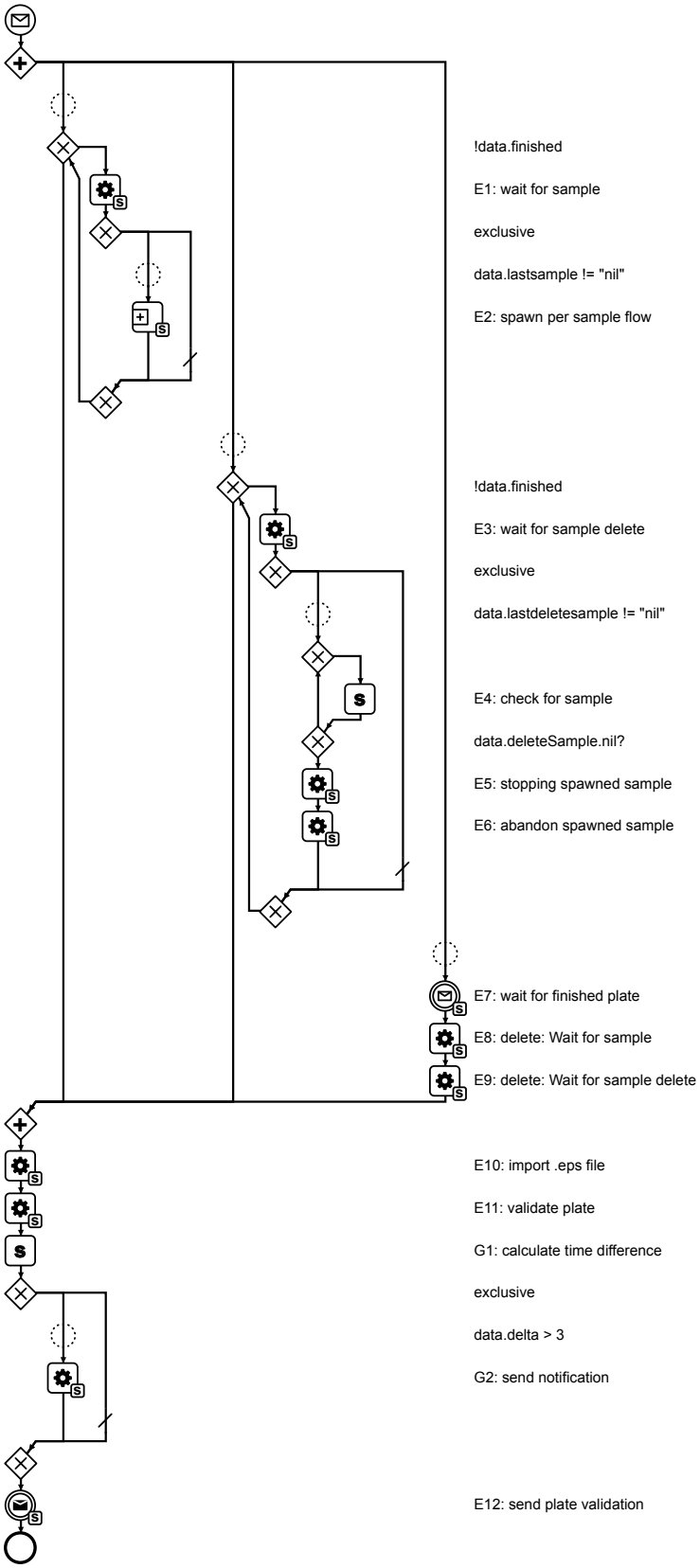


Figure 6.14: 96-Well Plate Workflow

ID	Name	Executor	Description
E1	Wait for Sample	Human	Waits for a Scanned Sample event (ID=3) containing the sample ID, plate ID, and position.
E2	Spawn per Sample Flow	Automatic	Spawns a new Sample Flow subprocess based on the sample properties from E1 and stores the spawned sample in an array.
E3	Wait for Sample Delete	Human	Waits for a Delete Sample event (ID=4) containing the sample ID.
E4	Check for Sample	Automatic	Checks whether the sample to be deleted is contained in the array.
E5	Stopping Spawned Sample	Automatic	Stopps the sample process instance by sending an HTTP-PUT request on the corresponding */state* endpoint.
E6	Abandon Spawned Sample	Automatic	Abandons the sample process instance by sending an HTTP-PUT request on the corresponding */state* endpoint. Afterward, the sample is removed from the array.
E7	Wait for Finished Plate	Automatic	Listens to the finished plate message event (Task ID=D4) from the parent instance by utilizing the plate ID as an identifier.
E8	Delete: Wait for Sample	Automatic	Calls back the task E1: Wait for Sample.
E9	Delete: Wait for Sample Delete	Automatic	Calls back the task E3: Wait for Sample Delete.
E10	Import .eps File	Human	Waits for an Import EPS event (ID=5) containing the plate ID.
E11	Validate Plate	Human	Waits for a Validate Plate event (ID=7) containing plate ID.
E12	Send Plate Validation	Automatic	Sends a plate validation message event to all associated sample process instances by utilizing the plate ID as an identifier.

Table 6.4: Workflow task description for 96-Well Plate

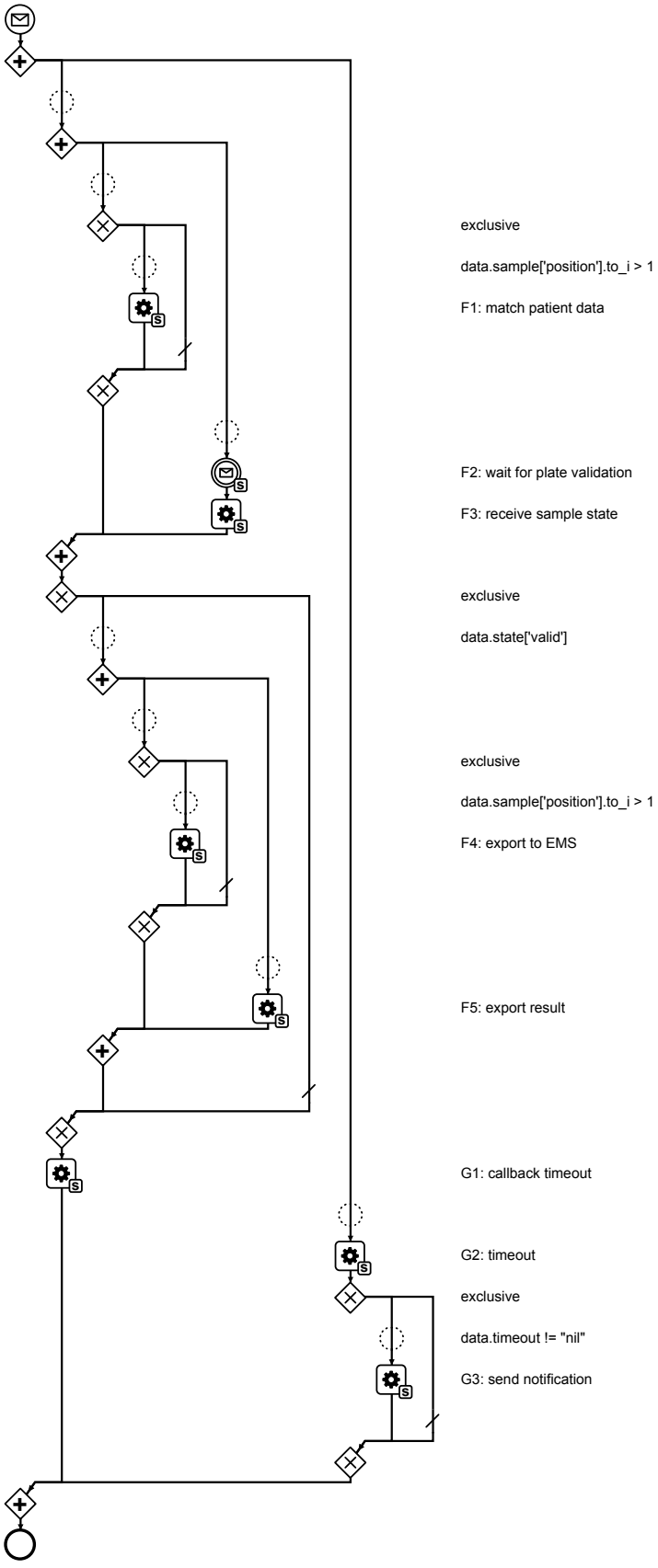


Figure 6.15: Sample Workflow

ID	Name	Executor	Description
F1	Match Patient Data	Human	Waits for a Match Patient event (ID=6) containing the sample ID.
F2	Wait for Plate Validation	Automatic	Listening on the plate validation message event (Task ID=E12) from the parent plate instance by utilizing the plate ID as an identifier.
F3	Receive Sample State	Human	Waits for a Sample State event (ID=13) containing the sample ID, plate ID, position, result, retry state, C_t value, and valid state.
F4	Export to EMS	Human	Waits for an Export EMS event (ID=8) containing the sample ID.
F5	Export Result	Human	Waits for an Export Result event (ID=9) containing the sample ID, completed boolean, and the result.

Table 6.5: Workflow task description for Sample Flow

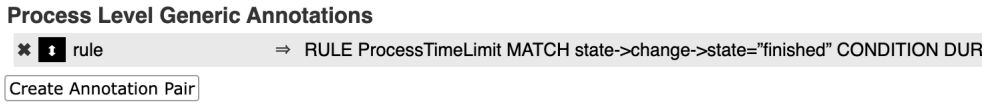


Figure 6.16: Process-level annotation of Listing 6.5 applied to Sample Flow

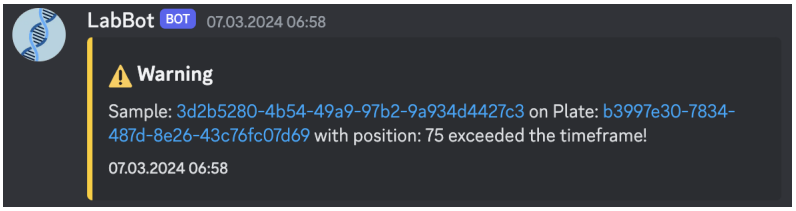


Figure 6.17: Example notification message sent from the bot

Figure 6.18: Task implementation for sending bot notifications from the CPEE

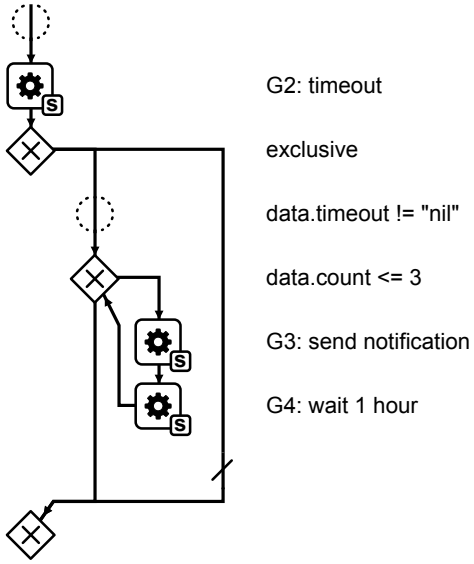


Figure 6.19: Excerpt of the adapted compliance rule from Sample Flow

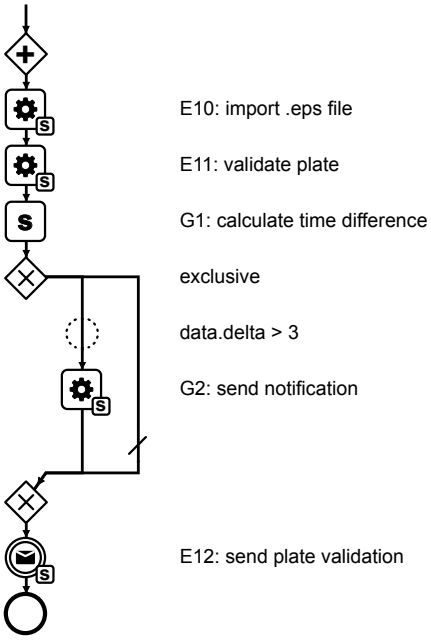


Figure 6.20: Inline backward compliance checking example

7 Evaluation

This section focuses on two subject areas. First, the process-based approach in this work is compared and classified to the four-step integration strategy outlined in the work of Mangler et al. [120]. Second, the process compliance findings are examined and discussed by utilizing empirical data derived from real-world laboratory SARS-CoV-2 test sets.

7.1 Assessment Regarding Integration Capability

The work of Mangler et al. [120] a process-based framework designed to facilitate the transition of small to medium enterprises (SMEs) towards advanced digitalization and automation within the manufacturing sector by addressing common barriers, such as resource constraints and the need for domain-specific knowledge requirements. The practical application of this framework is demonstrated in a pilot factory setting, detailing the interactions between a lathe, a robot, and a loading station to exemplify the integration of cyber-physical systems (CBS) in manufacturing. The case study particularly emphasizes the framework's ability to adapt and scale according to the specific needs and capacities of SMEs. The paper contributes to academic and practical discussions on Industry 4.0 by offering a scalable model that can be adapted to various manufacturing contexts, particularly benefiting SMEs that may struggle with the transition to fully automated systems. This framework not only enhances transparency and efficiency but also supports the incremental and reversible integration of digital technologies into existing manufacturing ecosystems.

The paper outlines a four-step integration approach that promotes a structured, gradual shift towards enhanced operational flexibility and reliability. These steps include the following:

1. **Soft Integration:** The initial phase focuses on minimally invasive interventions that aim to map and monitor existing operations without disrupting ongoing production processes. This step involves the creation of simple, executable process models that capture data from various sources on the shop floor, such as enterprise resource planning (ERP) and manufacturing execution systems (MES), and the machinery itself. The primary goal during this phase is to establish a foundational layer of digital oversight that allows for better visibility and preliminary data-driven insights.

2. **Process Modeling:** Building on the infrastructure established during the soft integration, this phase concentrates on enhancing the interaction between different production resources. By developing detailed BPMN models, this step systematically captures the dynamics of material sourcing and scheduling and the operational sequences of machines. The enhanced process models developed during this stage help orchestrate the manufacturing processes more effectively, paving the way for more integrated and optimized operations.

3. **Augmentation:** The third step addresses the digitalization gaps identified in the earlier phases, particularly those involving human machine interactions. This phase introduces digital tools that augment or replace manual processes with automated systems, thereby reducing the reliance on human intervention. Augmentation may include the deployment of user interfaces at workstations to facilitate direct data entry or the use of connected devices that streamline data capture and transmission processes. This phase significantly enhances the operational efficiency and accuracy of data handling within the manufacturing setup

4. **Control:** In the final phase of the integration approach, the framework assumes full control over the manufacturing operations. This stage is characterized by a shift from human- to system-driven processes, where the majority of operational decisions and actions are automated. The process models are now capable of executing complex tasks independently, including the management of production schedules and the direct control of machinery. This advanced level of automation ensures a high degree of precision and efficiency, facilitating a shift toward truly smart manufacturing environments.

Each step of this integration process is designed to be implemented progressively, allowing SMEs to manage the pace of digitalization according to their specific capabilities and needs. Implementing the four-step integration model previously outlined in the context of the laboratory domain confirms that the solution proposed in this work fully supports the initial two levels. Soft integration is accomplished through the seamless incorporation of data collection services into the LIS, which is designed to capture all relevant data related to SARS-CoV-2 testing through standardized interfaces in real-time. Data organization is facilitated by employing the logging schema presented in Listing 6.1. This data collection operates unobtrusively in the background, ensuring that laboratory operations remain undisturbed and unaltered. The collected data is aggregated and visualized in a dedicated frontend service (see Section 6.2.4).

The implementation of the initial phase establishes the foundation for the subsequent phase, process modeling. This phase involves mapping BPMN-based process models to the data gathered and integrating these models with the CPEE to orchestrate laboratory workflows. To facilitate this, a correlator has been developed to align the executed process instances with the acquired data in near real-time. The laboratory processes, which are explicitly defined, adhere to a hierarchical framework, as illustrated in Figure 6.9. Moreover, these processes foster collaboration between developers and domain experts. This cooperative strategy not only facilitates the creation of precise models that mirror actual processes but also improves the models' applicability across various scenarios within the laboratory. Such enhancements save time and resources in process design. This developmental stage also incorporates a validation phase that assesses effectiveness and efficiency by pinpointing potential bottlenecks, thereby enabling process refinement before their full implementation. Nonetheless, this phase does not actively interfere with ongoing laboratory testing processes, which explains why subsequent evolutionary steps have not been realized.

Steps 3 and 4 are not fulfilled due to the absence of direct interactions with the executed process instances. Nevertheless, it could be argued that Step 3 is partially satisfied, as compliance violations are proactively communicated by the lab bot, enhancing support for human operators and increasing operational efficiency beyond the passive monitoring capabilities characteristic of Level 2. Achieving full Level 3 capability would necessitate, for instance, the establishment

of a direct connection through standard interfaces from the PCR machine, specifically enabling the automatic loading of the .eps file into the LIS or the process model to eliminate the need for manual file handling. Alternatively, an enhanced version of the lab bot could offer actionable commands that directly modify process execution. For example, if a sample exceeds a predefined time interval, the lab bot could issue a notification that allows human operators to respond promptly by executing actions such as dispatching an automated email to the affected patient and informing them of a delay or the need for retesting. Furthermore, implementing a decision support system to interpret PCR results based on set criteria could contribute to maintaining high testing accuracy and reliability standards.

Step 4 focuses on the shift from manual to automated operations. During this phase, the PCR testing procedure is governed by a centralized management system that coordinates activities following the previously established BPM models. Automation at this stage includes the use of advanced robotics to automatically handle all physical tasks and software systems to manage workflow, data analysis, and communications. The system ensures that all operations are performed according to compliance standards and quality controls. It can also adapt workflows in real-time based on immediate feedback and analytical data, thereby optimizing the entire testing process dynamically. This final step can be considered equivalent to total lab automation (TLA).

Implementing these evolution steps requires collaboration between domain experts, IT specialists, and laboratory management to ensure that each phase is executed correctly and integrates seamlessly into the existing laboratory environment. The goal is to create a PCR testing process that is not only efficient and reliable but also adaptable to changing requirements and capable of handling high volumes of tests with minimal manual intervention.

7.2 Dataset

In this work, a dataset comprising log events from the laboratory testing process was collected and preserved over eight months, from December 2022 to July 2023. This dataset, which includes 461934 log entries initiated by 16 distinct users from 32 devices, forms the foundation for subsequent analysis. Over this interval, 59014 samples were scanned, yielding 746 well plates. Of these, seven well plates and 661 samples were discontinued (i.e., canceled). In total, data was exported to the EMS 84078 times. The number of exports exceeded the scanned samples because each sample might have been tested multiple times, potentially triggering an export with each test. A comprehensive analysis of the dataset was conducted through the integration with a semantic model in PowerBI via the MongoDB BI Connector. The process logs generated via the CPEE were sanitized, filtered, and anonymized by substituting sample IDs, plate IDs, and values with randomized identifiers and shifted/rounded values, ensuring that the original data could not be traced back. Subsequently, the dataset was made publicly available on Zenodo (refer to Section 1.3 for the link). The findings are detailed below.

The line graph in Figure 7.1 illustrates the variations in COVID-19 test positivity rates over a series of months from the captured time interval. After a low in February 2023 at 15.98%, the rate peaked in March 2023 at 35.69%, then fluctuated below and above the period's average rate of 27.39%, culminating in another peak in July 2023 at 36.84%. Throughout this period, the graph delineates a pattern characterized by significant fluctuations, demonstrating periods of increase

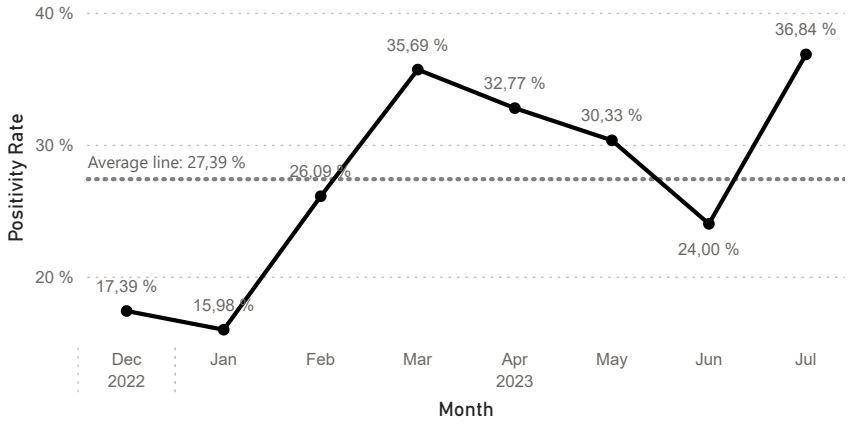


Figure 7.1: Percentage of positive samples over time

and decrease in the percentage of positive COVID-19 tests. The average positivity rate, illustrated by a dashed line, provides a reference against which these monthly variations can be gauged, offering insight into the relative magnitude of each month's rate against the overall average for the timeframe. The analysis of the CT values from positive samples revealed that the mean value is 26.59 with a standard deviation of 6.80.

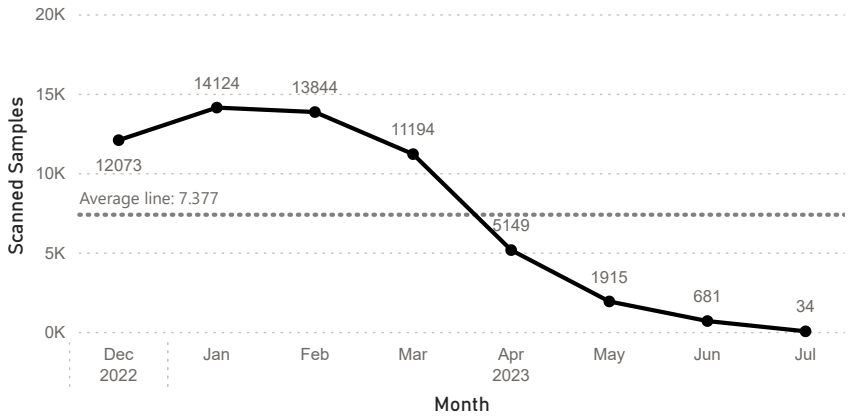


Figure 7.2: Number of scanned samples over time

Figure 7.2 displays the monthly volume of scanned samples. In December 2022, the count reached 12073 samples, followed by a modest increase in January 2023. Subsequently, there was a steady reduction in the number of samples, reaching a minimum of 34 samples by July 2023. An average of 7377 samples were recorded during this period, underscoring a significant decline in testing throughput. The reduction in sample testing is attributed to increased vaccination rates

and the availability of self-testing kits. Furthermore, beginning in February 2022, the Austrian government initiated the easing of COVID-19 restrictions [140], which decreased the necessity for mandatory testing.

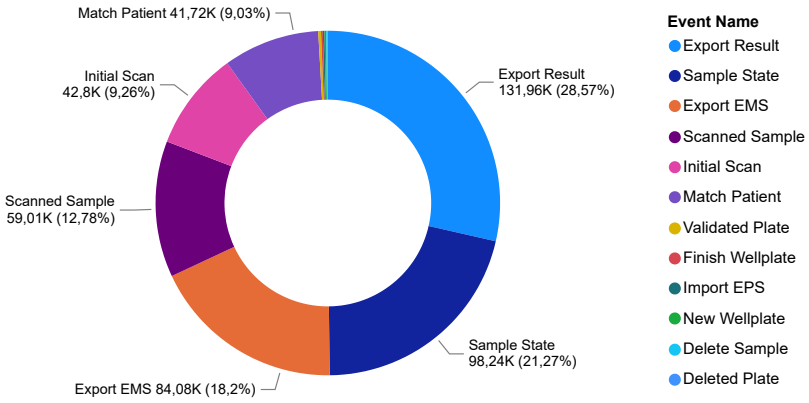


Figure 7.3: Distribution of captured event logs

The chart in Figure 7.3 presents a classification of laboratory events and provides their percentages concerning the total recorded activities. The Export Result category constitutes the predominant segment, comprising 28.57% of the events, highlighting that a substantial number of activities concern the exportation of test results. In contrast, the Export EMS occurs less frequently as positive controls (i.e., Position 1 samples) are not exported. The data shows a greater number of exports and sample states than scanned samples, attributed to the possibility of testing a sample multiple times, either on the same or a different well plate. There are fewer initial scans than scanned samples due to initial scan events being triggered at specific sample IDs only. Minor portions of the logs are allocated to actions such as validating plates, completing well plates, importing data, and the creation of new well plates, each depicted by smaller segments in the chart.

Figure 7.4 presents a quantitative analysis of the validity of the evaluated test samples. Of the total samples processed, 87.59%, equating to 86065 samples, were classified as *False*. This classification indicates non-compliance with the established criteria or standards necessary for further processing or accurate result determination. When a sample was deemed invalid, it did not trigger the export process. Conversely, a smaller proportion, 12.41% or 12190 samples, was categorized as *True*, indicating that these samples have met the validation requirements and are, therefore, suitable for reliable testing outcomes. These samples were subsequently exported. The high number of invalid samples is attributed to several factors. One significant reason involves the use of multiwells, where multiple samples are placed in the same well. If a test on a multiwell yields a positive result, all samples in the well are automatically marked as invalid until it is determined which sample(s) caused the positive result. Additionally, samples that test positive for the first

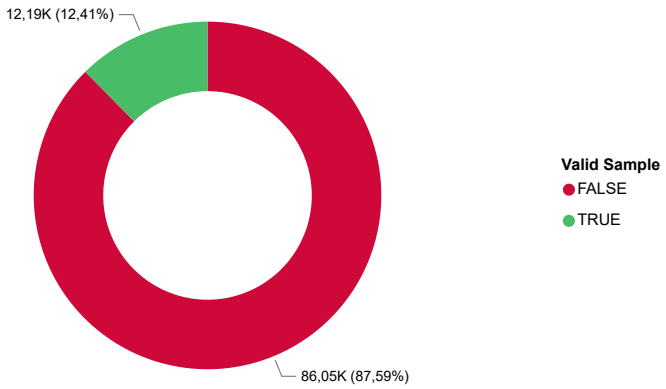


Figure 7.4: Distribution of valid samples

time are always subject to retesting for verification purposes, resulting in the initial test being classified as invalid.

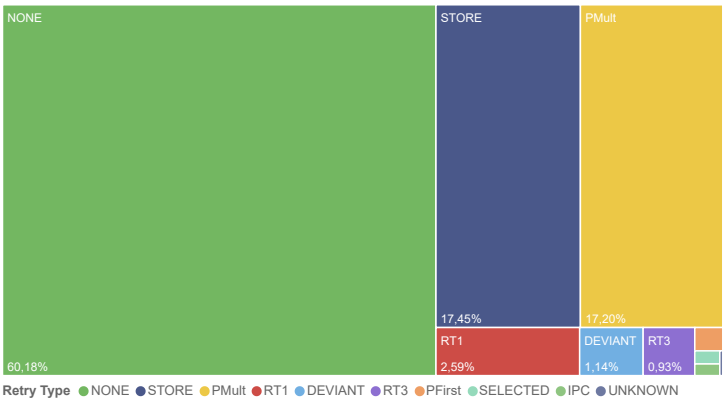


Figure 7.5: Tree map showing the distribution of retry types

The tree map displayed in Figure 7.5 illustrates the distribution of different retry types of samples. The largest segment, colored green and labeled *NONE*, comprises 60.18% of the total, indicating that the majority of samples did not require retesting. Adjacent to this, the *STORE* ssegment, shown in blue, represents 17.45% of samples, signifying those that were stored for potential reevaluation or further analysis. Smaller segments include *RT1*, *RT3*, and *DEVIANT*, which indicate specialized retesting procedures due to unusual or deviating initial results. *RT3* indicates retesting the sample three times.

7.3 Findings

External backward compliance checking was conducted through a direct evaluation of the dataset stored in a local MongoDB instance. Compliance Rule 1, as delineated in Listing 6.4, was implemented. This rule assesses the data from the “Sample State” event for anomalies, meaning that a sample cannot concurrently be categorized as DEVIANT and valid. The postmortem application of this rule to the collected event data revealed several violations. Specifically, 18 samples across five well plates were identified as non-compliant with this rule. Analysis of these samples showed diverse outcomes: three samples were positive; three were negative; and 12 were categorized as IPC, indicating an anomaly in the positive control. Notably, all IPC results were from samples located on the same well plate. Further examination of these 18 samples revealed that although they were marked as valid, no corresponding export routine was initiated. Consultation with laboratory personnel confirmed that these samples had been incorrectly classified as valid. This error has implications for the workflow, leading to process instances getting stuck as no export events were generated by the LIS. The LIS appears to have inherent logic to manage such discrepancies. However, laboratory staff remain uninformed; at present, sample data continues to be mislabeled as valid within the LIS. To address this issue, it is worthwhile to enhance compliance monitoring by implementing notifications for laboratory staff when such deviations are detected. This would enable timely corrections by resetting the sample status to invalid, thereby aligning the sample processing with compliance requirements. Without the implementation of the compliance rule, the laboratory staff will not be able to identify the violation since no warning is implemented in the LIS. Therefore, the time between the occurrence and detection of the violation by the lab personnel is infinite.

```
1 RULE ValidateSampleState
2 MATCH dataelements->change->changed[0] = "state"
3 CONDITION
4     (THIS->dataelements->change->content->values->state->retry = "PMult")
      AND (THIS->dataelements->change->content->values->state->valid = "
     TRUE") AND
5     (THIS->dataelements->change->content->values->state->result = "
     POSITIVE") AND
6     (THIS->dataelements->change->content->values->state->position > 1)
7 ACTION NOTIFY("Invalid sample state detected!")
8 ELSE ACTION LOG("Sample state OK")
```

Listing 7.1: Example Compliance Rule 4

A sample that is tested as part of a multiwell cannot be considered valid if the multiwell is tested positive because it is unclear which samples are the positive ones. This constraint is encapsulated in a compliance rule derived from Listing 6.3, resulting in Listing 7.1. An evaluation of this rule on the dataset indicated that 49 samples from 10 multiwells breached this regulation. These multiwells contained between two and six samples, averaging 4.9 samples per well. Similarly to Compliance Rule 1, no export routines were initiated, despite the samples being erroneously marked as valid. This oversight led to running sample process instances getting stuck, as the expected export events for samples flagged as valid would not occur. Consultations with laboratory personnel confirmed the misclassification of these samples as valid, which led to data inconsistencies within

Task	Without Compl. Checking Time passed (in h)	With Compl. Checking Time passed (in h)
Sample Scanned	0	0
Sample Processed	$\approx 2,32$	$\approx 2,32$
Patient Data Matched	$\approx 49,18$	24
Δ Delta	$\approx 25,18$ hours	

Table 7.1: Sample flow comparison with and without compliance checking enabled

the LIS. Without the application of the compliance rule, these discrepancies would have remained undetected, as the LIS lacks proactive error handling mechanisms for this issue. Consequently, the error detection time extends indefinitely. The compliance rule outlined in Listing 7.1 can be adapted for inline compliance monitoring by incorporating an exclusive gateway immediately after receiving the sample status (i.e., Task F3). An alarm, such as a notification dispatched through the laboratory bot to the staff that references the affected sample and well plate, could be triggered when an invalid sample state is detected (i.e. the condition is met).

Various use cases have been identified considering the compliance rule outlined in Listing 6.5, which specifies that sample flow processes should be completed within 24 hours to ensure the timely delivery of patient test results. These use cases include adherence to SLA completion times to circumvent penalties associated with non-compliance. During the monitoring period, more than 20 sample flow executions were detected where the sample export was delayed beyond 24 hours due to incomplete patient data. Such delays prevented the completion of the "F1: match patient data" task, consequently activating the compliance alert of Task G2 from Figure 6.15, which prompted an automated notification from the laboratory's bot system (see example message in Figure 6.17).

A specific sample was analyzed to compare the timelines between the manual detection of compliance issues and those identified automatically by the system. Table 7.1 presents these findings, illustrating the lifecycle of a particular sample both, with and without the application of the compliance rule. In this example, a sample logged on 15.01.2023 at 20:06 was matched with patient data on 17.01.2023 at 21:17, which is approximately 2.05 days later. The sample was processed and ready by 22:20 on 15.01.2023, which was 134 minutes after the process started (i.e., the sample was scanned), but the absence of necessary patient data prevented the exportation of the result. This issue was manually resolved two days later by laboratory personnel.

The proactive use of the compliance rule facilitates earlier notifications to laboratory staff, allowing for faster corrective actions compared to traditional methods that rely on manual oversight. For instance, the proactive detection by the compliance rule of missing patient data resulted in identifying the issue 1.04 days sooner (based on the example from 16.01.2023 at 20:06, when the compliance rule was breached) compared to the manual intervention on 17.01.2023 at 21:17. Consequently, in this instance, the process deviation was recognized 1.04 days sooner through the process-based approach than through conventional passive monitoring methods.

The compliance rule could be further tightened by ensuring patient data is available right after the sample state has been received. For instance, in the scenario previously mentioned, an alert is activated immediately after the sample state has been received, as evidenced on 15.01.2023 at

22:20. This approach facilitates a significant enhancement in timeliness by approximately 1.96 days.

Further analysis of the lab event logs for this sample revealed that, despite valid test outcomes, the sample state was flagged as invalid due to the missing data at that point. Interestingly, the "Export Result" event was still initiated. Discussions with laboratory staff confirmed that the export process was executed but failed due to missing patient data. Consequently, a "complete" flag has now been added to the "Export Result" event message body to signify the success or failure of the export.

The recurrent growth of the producer queue from the correlator due to unmatched laboratory events prompted an investigation into its underlying causes. The analysis identified that certain laboratory events, including export routines, were triggered multiple times. For instance, some samples were redundantly exported to the EMS as a result of multiple triggers across various clients, as indicated by unique MAC addresses and user entries. These findings were reported to the laboratory's IT administrator, leading to a review of the LIS. The impact of exporting the same sample multiple times to the EMS remains unclear, although it is assumed that the EMS possesses inherent mechanisms to handle such redundancies. Implementing a Level 4 process-aware system could be the solution to prevent redundant operations, where process instances trigger export routines. Alternatively, establishing a compliance rule to monitor the correlator's producer queue could help in detecting and alerting to any violations, though this method does not proactively prevent redundancy.

A comprehensive analysis of the dataset indicated that a significant proportion of lab events originated from the root user. Specifically, the root user accounted for 19.57% of all events, ranking as the second most frequent user. Further examination of the event types revealed that the root user exclusively initiated two actions, namely Export EMS (53.85%) and Match Patient Data (46.15%). Using the root user account is not recommended due to the elevated risks associated with its extensive access rights. The root account, by design, can make unrestricted modifications to system settings and files. This unrestricted access also poses a security threat, as it can be exploited by malware or unauthorized users to gain control over the system. To enhance security, it is preferable to operate under accounts with more limited permissions that are sufficient for specific tasks. This practice of assigning the minimum necessary permissions, known as the principle of least privilege, helps minimize the risk of catastrophic errors and security breaches, thereby maintaining a more secure and stable system environment. The findings, together with the corresponding recommendation, were communicated to the responsible IT administrator.

8 Discussion

This work successfully applied a process-based framework aimed at enhancing digitalization and automation within laboratory settings, specifically focusing on SARS-CoV-2 testing processes. The evaluation results demonstrate the strengths and limitations of the framework as applied in a real-world environment, providing crucial insights into the process compliance and integration capabilities necessary for efficient laboratory operations.

8.1 Integration and Digitalization

The framework was benchmarked against the four-step integration strategy of Mangler et al. [120], revealing substantial alignment, particularly in the initial two steps, namely soft integration and process modeling. Soft integration was effectively achieved through the integration of data collection services into the LIS, which handled data without disrupting laboratory operations. The subsequent process modeling phase allowed for an effective mapping of BPMN-based process models to actual data flows within the laboratory. This modeling facilitated a more orchestrated workflow, which proved essential for enhancing operational transparency and efficiency.

Comparing the laboratory's digitalization journey with the broader framework of the four-step integration model offers several insights. First, it underscores the need for a flexible integration strategy that can be adapted based on the specific needs and capacities of the entity, a principle that is central to the framework discussed by Mangler et al. [120]. Second, the challenges encountered in advancing beyond the process modeling phase due to the lack of direct interaction with process instances provide empirical support advocating for more robust frameworks that can handle such complexities, especially in environments with high compliance requirements, such as laboratories.

However, the framework did not extend into the Augmentation and Control steps, which would involve more advanced digital interventions such as automating manual processes and achieving full operational autonomy through smart systems. The absence of these steps indicates a gap in achieving a fully integrated cyber-physical system, highlighting an area for further development and exploration in future research.

8.2 Process Compliance and Operational Efficiency

The evaluation revealed significant insights into process compliance and anomaly detection within the laboratory processes. Compliance rules implemented in the LIS helped in identifying and addressing deviations in real-time and ex-post, significantly reducing potential errors in sample handling and data processing. For instance, the proactive communication of compliance violations

via the lab bot enhanced operational efficiency and supported human operators more effectively than traditional passive monitoring systems.

The detailed analysis of log data, including the distribution of valid and invalid samples, underscores the critical role of accurate sample handling and the need for stringent compliance to maintain high testing standards. The discovery of compliance violations, such as the misclassification of sample validity, highlights the necessity for dynamic compliance rules integrated within LIS. This aspect underscores a gap in the real world, where compliance is often assumed rather than actively managed and monitored.

8.3 Limitations

The solution lacks direct interaction with executed process instances, particularly in the automation and control of laboratory processes. The absence of direct connectivity for automation, such as automated data loading and process adjustments, suggests a limitation in evaluating the practicality of real-time, automated decision-making within the lab setting. This limitation could impact the generalizability of the results to environments where high levels of automation are critical for operational efficiency and accuracy.

Despite advancements in process modeling and data collection, the provided solution still relies significantly on manual interventions, especially in handling compliance violations and operational discrepancies. The evaluation identified several instances where samples were misclassified or anomalies occurred without automatic correction.

Additionally, the study's reliance on the process compliance findings using the laboratory's SARS-CoV-2 test sets introduces limitations due to the specific nature of these tests. The generalizability of the findings may be affected by the unique circumstances and requirements associated with COVID-19 testing, such as the high variability in test demand and the evolving nature of testing protocols. This context-specific focus might not fully translate to other laboratory environments or testing requirements, potentially limiting the broader applicability of the results.

8.4 Future Work

Several areas for future research emerge based on the limitations identified in this study. Expanding the implementation of the four-step integration model to include the full range of steps will provide a more comprehensive understanding of the model's benefits and challenges. This could involve developing additional capabilities for real-time interaction with process instances and exploring the integration of advanced robotics and automated decision-making systems in the laboratory environment. Another approach consists of establishing direct connections and interfaces with key laboratory equipment to facilitate real-time data integration and control mechanisms. This will support the advancement to higher levels of the integration strategy, particularly the "Augmentation" and "Control" phases.

Research could also focus on addressing the scalability of the model beyond the context of COVID-19 testing. Applying the framework to a variety of laboratory processes such as blood tests, including those with different scales of operations and types of tests, would help validate and

refine the model's effectiveness across diverse settings. This expansion would not only enhance the framework's robustness but also its adaptability to different technological and operational challenges.

Further, the study highlights a need for improved compliance monitoring and error detection systems within the LIS. Future research could explore the development of more sophisticated monitoring tools that can proactively detect and resolve process deviations and compliance violations. This could include the integration of artificial intelligence and machine learning techniques to predict potential errors and automate corrective actions, thereby enhancing the reliability and efficiency of laboratory operations (i.e., predictive maintenance).

Finally, considering the significant role of data quality and management highlighted in the study, there is an opportunity to delve deeper into the impact of data integrity on process outcomes. Research could focus on developing advanced data validation and cleaning techniques to maintain high-quality data inputs, which is crucial for the successful automation and digitalization of laboratory processes.

9 Conclusion

This work explored the integration of process-aware information systems in laboratory settings, primarily focusing on the SARS-CoV-2 testing processes during the COVID-19 pandemic. The research delved into how these systems enhance operational efficiency, flexibility, and compliance within medical diagnostic laboratories. Process-aware information systems significantly enhance operational efficiencies across healthcare and non-healthcare industries.

The pandemic underscored the critical role of agile and compliant processes, as laboratories were compelled to rapidly adapt to evolving health guidelines and testing demands. Contrastingly, non-healthcare industries typically harness process-aware information systems to streamline operations and bolster efficiency with a strong focus on cost-effectiveness and scalability. These sectors often face less stringent compliance demands compared to healthcare, allowing more flexibility in the application of process technologies (RQ1).

Through a detailed examination of the existing testing process, the provided implementation has real-time data capture capabilities to ensure that laboratory activities are immediately logged and monitored, facilitating a dynamic response to process deviations and anomalies. With comprehensive real-time data insights offered through the frontend and LabBot, laboratory personnel could identify and address potential issues swiftly, allocate resources more effectively, and predict future trends based on historical data analysis, enhancing decision-making and operational efficiency (RQ2). The source code and process models developed during this thesis are available for review and reuse on GitHub[1].

The findings confirmed that implementing process-aware information systems can lead to substantial improvements in several areas, such as data collection, traceability, and compliance within laboratories. The empirical evaluation of the provided solutions' real-time data capture and compliance checking capabilities is crucial in reducing manual errors and improving laboratories' responsiveness to process deviations such as incorrect sample states induced by human errors. This evaluation helps enhance the reliability and accuracy of the test results (RQ3). For detailed process logs supporting these evaluations, refer to the dataset on Zenodo[2].

The implications of these findings are far-reaching. They offer a blueprint for other laboratories seeking to enhance their operational capabilities. The move toward a more automated and process-aware system is indicative of the broader trends in healthcare toward digitalization and precision management. These advancements are crucial in the context of increasing demands on healthcare systems and the need for rapid and accurate diagnostics, as evidenced during the pandemic. Moreover, this thesis makes a critical contribution to the field of laboratory automation by demonstrating the practical application of BPM by utilizing the CPEE in a real-world laboratory environment.

[1] `https://github.com/greschner/cpee-worklist`, last accessed: 2024-06-12
[2] `https://doi.org/10.5281/zenodo.11617408`, last accessed: 2024-06-12

In conclusion, this work underscores the significant impact of process-aware information systems for laboratory automation, particularly in enhancing the efficiency, accuracy, and compliance of medical diagnostic processes. The insights gained and the methodologies developed are valuable for future advancements in the field of laboratory process management.

Bibliography

[1] Wil MP van der Aalst, Vladimir Rubin, Boudewijn F van Dongen, Ekkart Kindler, and Christian W Günther. Process mining: A two-step approach using transition systems and regions. *BPM Center Report BPM-06-30, BPMcenter. org*, 6, 2006.

[2] Wei-jie Guan, Zheng-yi Ni, Yu Hu, Wen-hua Liang, Chun-quan Ou, Jian-xing He, Lei Liu, Hong Shan, Chun-liang Lei, David SC Hui, et al. Clinical characteristics of coronavirus disease 2019 in china. *New England journal of medicine*, 382(18):1708–1720, 2020.

[3] Xiaoyi Huang, Fengxiang Wei, Liang Hu, Lijuan Wen, and Ken Chen. Epidemiology and clinical characteristics of covid-19. *Archives of Iranian medicine*, 23(4):268–271, 2020.

[4] Jonathan M Tsai, Nicole V Tolan, Athena K Petrides, Sanjat Kanjilal, Manfred Brigl, Neal I Lindeman, Yen-Der Li, Milenko J Tanasijevic, Sankha S Basu, and Stacy EF Melanson. How sars-cov-2 transformed the clinical laboratory: challenges and lessons learned. *The journal of applied laboratory medicine*, 6(5):1338–1354, 2021.

[5] Antonio La Marca, Martina Capuzzo, Tiziana Paglia, Laura Roli, Tommaso Trenti, and Scott M Nelson. Testing for sars-cov-2 (covid-19): a systematic review and clinical guide to molecular and serological in-vitro diagnostic assays. *Reproductive biomedicine online*, 41(3):483–499, 2020.

[6] Caitlin R Ondracek, Jonathan R Genzen, Christina M Lockwood, Saswati Das, Phillip Kang, and Stacy E F Melanson. Robust response of the clinical laboratory to the covid-19 pandemic despite significant challenges. *The journal of applied laboratory medicine*, 8(6):1160–1172, 2023.

[7] Victor M Corman, Olfert Landt, Marco Kaiser, Richard Molenkamp, Adam Meijer, Daniel KW Chu, Tobias Bleicker, Sebastian Brünink, Julia Schneider, Marie Luisa Schmidt, et al. Detection of 2019 novel coronavirus (2019-ncov) by real-time rt-pcr. *Eurosurveillance*, 25(3):2000045, 2020.

[8] Supply shortages impacting covid-19 and non-covid testing. `https://asm.org/Articles/2020/September/Clinical-Microbiology-Supply-Shortage-Collecti-1`, Nov 2020. [Last accessed: 2023-11-05].

[9] Eva Krause, Janine Michel, Andreas Puyskens, Natalie Hofmann, Thomas Rinner, Barbara Biere, Brigitte G Dorner, Martin Skiba, Lars Schaade, and Andreas Nitsche. Flexible upscaling of laboratory pcr testing capacity at the robert koch institute during the sars-cov-2 pandemic. *Virology Journal*, 20(1):139, 2023.

[10] Eric Q. Konnick, Jordan Laser, and Karen E. Weck. The Role of Clinical Laboratories in Emerging Pathogens—Insights From the COVID-19 Pandemic. *JAMA Health Forum*, 2(10):e213154–e213154, 10 2021.

[11] Tze Ping Loh, Andrea Rita Horvath, Cheng-Bin Wang, David Koch, Khosrow Adeli, Nicasio Mancini, Maurizio Ferrari, Robert Hawkins, Sunil Sethi, Giuseppe Lippi, The International Federation of Clinical Chemistry, and Laboratory Medicine Taskforce on COVID-19. Operational considerations and challenges of biochemistry laboratories during the covid-19 outbreak: an ifcc global survey. *Clinical Chemistry and Laboratory Medicine (CCLM)*, 58(9):1441–1449, 2020.

[12] Florian Pauker, Juergen Mangler, Stefanie Rinderle-Ma, and Matthias Ehrendorfer. Industry 4.0 integration assessment and evolution at evva gmbh: Process-driven automation through centurio. work. In *Business Process Management Cases Vol. 2: Digital Transformation-Strategy, Processes and Execution*, pages 81–91. Springer, 2021.

[13] Olena Filchakova, Dina Dossym, Aisha Ilyas, Tamila Kuanysheva, Altynay Abdizhamil, and Rostislav Bukasov. Review of covid-19 testing and diagnostic methods. *Talanta*, 244:123409, 2022.

[14] Beatriz Böger, Mariana M Fachi, Raquel O Vilhena, Alexandre F Cobre, Fernanda S Tonin, and Roberto Pontarolo. Systematic review with meta-analysis of the accuracy of diagnostic tests for covid-19. *American journal of infection control*, 49(1):21–29, 2021.

[15] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *J. Manage. Inf. Syst.*, 24(3):45–77, December 2007.

[16] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, mar 2004.

[17] Roel Wieringa and Ayşe Moralı. Technical action research as a validation method in information systems design science. In Ken Peffers, Marcus Rothenberger, and Bill Kuechler, editors, *Design Science Research in Information Systems. Advances in Theory and Practice*, pages 220–238, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[18] Giuseppe Lippi and Giorgio Da Rin. Advantages and limitations of total laboratory automation: a personal overview. *Clinical Chemistry and Laboratory Medicine (CCLM)*, 57(6):802–811, 2019.

[19] Charles D Hawker. Nonanalytic laboratory automation: a quarter century of progress. *Clinical chemistry*, 63(6):1074–1082, 2017.

[20] Chris H Demiris and Peter R Ciment. Task targeted automation,(tta) from roche diagnostics provides a cost effective and easily justified approach to automation for improving the efficiency of the clinical laboratory. *JALA: Journal of the Association for Laboratory Automation*, 3(5):64–66, 1998.

[21] Adam L Bailey, Nathan Ledeboer, and Carey-Ann D Burnham. Clinical Microbiology Is Growing Up: The Total Laboratory Automation Revolution. *Clinical Chemistry*, 65(5):634–643, 05 2019.

[22] Charles D Hawker. Laboratory automation: total and subtotal. *Clinics in laboratory medicine*, 27(4):749–770, 2007.

[23] David A Armbruster, David R Overcash, and Jaime Reyes. Clinical chemistry laboratory automation in the 21st century-amat victoria curam (victory loves careful preparation). *The Clinical Biochemist Reviews*, 35(3):143, 2014.

[24] CLSI. Clsi auto03: Laboratory automation: Communications with automated clinical laboratory systems, instruments, devices, and information systems, 2nd edition. Technical report, Clinical and Laboratory Standards Institute, 2009. [Last accessed 06-02-2024].

[25] Melvin P Weinstein and James S Lewis. The clinical and laboratory standards institute subcommittee on antimicrobial susceptibility testing: background, organization, functions, and processes. *Journal of clinical microbiology*, 58(3):10–1128, 2020.

[26] Sadamu Takasaka. Standardization of hl7 clinical laboratory system. *JALA: Journal of the Association for Laboratory Automation*, 7(5):72–74, 2002.

[27] CLSI. Clsi lis01: Specification for low-level protocol to transfer messages between clinical laboratory instruments and computer systems, 2nd edition. Technical report, Clinical and Laboratory Standards Institute, 2008. [Last accessed 06-02-2024].

[28] CLSI. Clsi lis02: Specification for transferring information between clinical laboratory instruments and information systems, 2nd edition. Technical report, Clinical and Laboratory Standards Institute, 2004. [Last accessed 06-02-2024].

[29] Arthur ter Hofstede, Wil MP van der Aalst, Arthur HM ter Hofstede, and Mathias Weske. Business process management: A survey. In *Business Process Management: International Conference, BPM 2003 Eindhoven, The Netherlands, June 26–27, 2003 Proceedings 1*, pages 1–12. Springer, 2003.

[30] Mathias Weske et al. Concepts, languages, architectures. *Business Process Management*, 2007.

[31] Marlon Dumas, Wil M Van der Aalst, and Arthur H Ter Hofstede. *Process-aware information systems: bridging people and software through process technology*. John Wiley & Sons, 2005.

[32] Michael Georgeff and Jon Pyke. Dynamic process orchestration. *White paper, Staffware PLC http://is. tm. tue. nl/bpm2003/download/WP% 20Dynamic% 20Proce% ss% 20Orchestration% 20v1. pdf*, 2003.

[33] Peter Lawrence. *Workflow handbook 1997*. John Wiley & Sons, Inc., 1997.

[34] RS Mans. Workflow support for the healthcare domain. 2011.

[35] Wil MP van der Aalst. Business process management: a personal view. *Business Process Management Journal*, 10(2), 2004.

[36] Florian Stertz, Juergen Mangler, and Stefanie Rinderle-Ma. Temporal conformance checking at runtime based on time-infused process models, 2020.

[37] Jürgen Mangler and Stefanie Rinderle-Ma. Cpee - cloud process exection engine. In *Int'l Conference on Business Process Management*, CEUR-WS.org, September 2014.

[38] Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50, 2016.

[39] H. M. W. Verbeek, Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Xes, xesame, and prom 6. In Pnina Soffer and Erik Proper, editors, *Information Systems Evolution*, pages 60–75, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[40] W.M.P. Aalst, van der. *Process mining : discovery, conformance and enhancement of business processes*. Springer, Germany, 2011.

[41] Wil MP Van Der Aalst, Hajo A Reijers, and Minseok Song. Discovering social networks from event logs. *Computer Supported Cooperative Work (CSCW)*, 14:549–593, 2005.

[42] Asef Pourmasoumi and Ebrahim Bagheri. Business process mining. *Encyclopedia with Semantic Computing and Robotic Intelligence*, 1(01):1630004, 2017.

[43] Elham Ramezani, Dirk Fahland, and Wil M. P. van der Aalst. Where did i misbehave? diagnostic information in compliance checking. In Alistair Barros, Avigdor Gal, and Ekkart Kindler, editors, *Business Process Management*, pages 262–278, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[44] Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil M.P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information Systems*, 54:209–234, 2015.

[45] Elham Ramezani, Dirk Fahland, Jan Martijn van der Werf, and Peter Mattheis. Separating compliance management and business process management. In *Business Process Management Workshops: BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part II 9*, pages 459–464. Springer, 2012.

[46] Heba Aamer, Marco Montali, and Jan Van den Bussche. What can database query processing do for instance-spanning constraints? In *International Conference on Business Process Management*, pages 132–144. Springer, 2022.

[47] Alexander Seeliger, Timo Nolle, Benedikt Schmidt, and Max Mühlhäuser. Process compliance checking using taint flow analysis. 2016.

[48] Marwane el Kharbili, Ana Karla Alves de Medeiros, Sebastian Stein, and Wil MP van der Aalst. Business process compliance checking: Current state and future challenges. 2008.

[49] Alberto Dolci, Davide Giavarina, Sara Pasqualetti, Dominika Szőke, and Mauro Panteghini. Total laboratory automation: Do stat tests still matter? *Clinical Biochemistry*, 50(10):605–611, 2017. Improving laboratory Performance Through Quality Indicators.

[50] Johanna I Westbrook, Andrew Georgiou, and Marilyn I Rob. Computerised order entry systems: sustained impact on laboratory efficiency and mortality rates? *Studies in Health Technology and Informatics*, 136:345, 2008.

[51] Andrew Georgiou, Margaret Williamson, Johanna I Westbrook, and Sangeeta Ray. The impact of computerised physician order entry systems on pathology services: a systematic review. *International journal of medical informatics*, 76(7):514–529, 2007.

[52] Bernhard Breil, Fleur Fritz, Volker Thiemann, and Martin Dugas. Mapping turnaround times (tat) to a generic timeline: a systematic review of tat definitions in clinical domains. *BMC medical informatics and decision making*, 11:1–12, 2011.

[53] Hara P Pati and Gurmeet Singh. Turnaround time (tat): difference in concept for laboratory and clinician. *Indian Journal of Hematology and Blood Transfusion*, 30:81–84, 2014.

[54] Amy H Lou, Manal O Elnenaei, Irene Sadek, Shauna Thompson, Bryan D Crocker, and Bassam Nassar. Evaluation of the impact of a total automation system in a large core laboratory on turnaround time. *Clinical Biochemistry*, 49(16-17):1254–1258, 2016.

[55] Binita Goswami, Bhawna Singh, Ranjna Chawla, VK Gupta, and V Mallika. Turn around time (tat) as a benchmark of laboratory performance. *Indian Journal of Clinical Biochemistry*, 25:376–379, 2010.

[56] Rodney S Markin and Scott A Whalen. Laboratory automation: trajectory, technology, and tactics. *Clinical chemistry*, 46(5):764–771, 2000.

[57] Marijana Miler, Nora Nikolac Gabaj, Lora Dukic, and Ana-Maria Simundic. Key performance indicators to measure improvement after implementation of total laboratory automation abbott accelerator a3600. *Journal of medical systems*, 42:1–12, 2018.

[58] Shamayleh Abdulrahim, Awad Mahmoud, and Farhat Jumana. Iot based predictive maintenance management of medical equipment. *Journal of Medical Systems*, 44(4), 2020.

[59] Michael P Cornes and Chandrashekar Shetty. Mechanisms of measuring key performance indicators in the pre-analytical phase. *J Lab Precision Med*, 2018.

[60] Janne Cadamuro, Martin Gaksch, Cornelia Mrazek, Elisabeth Haschke-Becher, and Mario Plebani. How do we use the data from pre-analytical quality indicators and how should we. *J Lab Precis Med*, 3(6):40–40, 2018.

[61] Peijun ZHAI, Dongmei HU, Yue FU, and Yingshu ZOU. Major changes will take place on iso15189" medical laboratories: requirements for quality and competence". *Chinese Journal of Laboratory Medicine*, pages 677–680, 2022.

[62] Maria Salinas, Maite López-Garrigós, Mercedes Gutiérrez, Javier Lugo, Jose Vicente Sirvent, and Joaquin Uris. Achieving continuous improvement in laboratory organization through performance measurements: a seven-year experience. *Clinical chemistry and laboratory medicine*, 48(1):57–61, 2010.

[63] Mostafa M Rizk, Adel Zaki, Nermine Hossam, and Yasmin Aboul-Ela. Evaluating laboratory key performance using quality indicators in alexandria university hospital clinical chemistry laboratories. *The Journal Of The Egyptian Public Health Association*, 89(3):105–113, 2014.

[64] Vivek Patkar, Matthew South, and Richard Thomson. From guidelines to careflows: modelling and supporting complex clinical processes. *Computer-based medical guidelines and protocols: a primer and current trends*, 139:44, 2008.

[65] Michael Binder, Wolfgang Dorda, Georg Duftschmid, Reinhold Dunkl, Karl Anton Fröschl, Walter Gall, Wilfried Grossmann, Kaan Harmankaya, Milan Hronsky, Stefanie Rinderle-Ma, et al. On analyzing process compliance in skin cancer treatment: An experience report from the evidence-based medical compliance cluster (ebmc 2). In *Advanced Information Systems Engineering: 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings 24*, pages 398–413. Springer, 2012.

[66] Wil M. P. van der Aalst. *Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management*, pages 1–65. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[67] Richard Lenz and Manfred Reichert. It support for healthcare processes–premises, challenges, perspectives. *Data & Knowledge Engineering*, 61(1):39–58, 2007.

[68] Meg Murray. Strategies for the successful implementation of workflow systems within healthcare: a cross case comparison. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 10–pp. IEEE, 2003.

[69] Silvana Quaglini, Mario Stefanelli, Giordano Lanzola, Vincenzo Caporusso, and Silvia Panzarasa. Flexible guideline-based patient careflow systems. *Artificial intelligence in medicine*, 22(1):65–80, 2001.

[70] Mark J Halsted and Craig M Froehle. Design, implementation, and assessment of a radiology workflow management system. *American Journal of Roentgenology*, 191(2):321–327, 2008.

[71] Minmin Han, Thomas Thiery, and Xiping Song. Managing exceptions in the medical workflow systems. In *Proceedings of the 28th international conference on Software engineering*, pages 741–750, 2006.

[72] R Hess. The chester county hospital: Case study. *2007 BPM and Workflow Handbook*, 1, 2007.

[73] Krishna Juluru and John Eng. Internet-based radiology order-entry, reporting, and workflow management system for coordinating urgent study requests during off-hours. *American Journal of Roentgenology*, 184(3):1017–1020, 2005.

[74] Silvia Panzarasa, Silvana Quaglini, Giuseppe Micieli, Simona Marcheselli, Mauro Pessina, Corrado Pernice, Anna Cavallini, Mario Stefanelli, et al. Improving compliance to guidelines through workflow technology: implementation and results in a stroke unit. *Medinfo*, 839(2):834–839, 2007.

[75] GD Laet, J Naudts, and J Vandevivere. Workflow in nuclear medicine. *Computerized medical imaging and graphics*, 25(2):195–199, 2001.

[76] Martin Sedlmayr, Thomas Rose, Torben Greiser, Rainer Röhrig, Markus Meister, and Achim Michel-Backofen. Automating standard operating procedures in intensive care. In John Krogstie, Andreas Opdahl, and Guttorm Sindre, editors, *Advanced Information Systems Engineering*, pages 516–530, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[77] Peter Dadam and Manfred Reichert. The adept project: a decade of research and development for robust and flexible process support: Challenges and achievements. *Computer Science-Research and Development*, 23:81–97, 2009.

[78] Manfred Reichert and Peter Dadam. Adept flex—supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10:93–129, 1998.

[79] Peter Dadam, Manfred Reichert, Stefanie Rinderle, Martin Jurisch, Hilmar Acker, Kevin Göser, Ulrich Kreher, and Markus Lauer. Towards truly flexible and adaptive process-aware information systems. In *Information Systems and e-Business Technologies: 2nd International United Information Systems Conference UNISCON 2008 Klagenfurt, Austria, April 22–25, 2008 Proceedings 2*, pages 72–83. Springer, 2008.

[80] Manfred Reichert, Stefanie Rinderle, Ulrich Kreher, and Peter Dadam. Adaptive process management with adept2. In *21st International Conference on Data Engineering (ICDE'05)*, pages 1113–1114. IEEE, 2005.

[81] Nathan Goodman, Steve Rozen, and Lincoln Stein. The labflow system for workflow management in large scale biology research laboratories. In *ISMB*, pages 69–77, 1998.

[82] James L Peterson. Petri nets. *ACM Computing Surveys (CSUR)*, 9(3):223–252, 1977.

[83] Margareth Timóteo, Emanuelle Lourenço, Ana Carolina Brochado, Luciana Domenico, Joice da Silva, Bruna Oliveira, Renata Barbosa, Pietro Montemezzi, Carlos Fernando de Almeida Barros Mourão, Beni Olej, et al. Digital management systems in academic health sciences laboratories: A scoping review. In *Healthcare*, volume 9, page 739. MDPI, 2021.

[84] Josep Carmona, Boudewijn van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking*. Springer, 2018.

[85] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst. Towards robust conformance checking. In Michael zur Muehlen and Jianwen Su, editors, *Business Process Management Workshops*, pages 122–133, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[86] Sebastiaan J van Zelst, Alfredo Bolt, Marwan Hassani, Boudewijn F van Dongen, and Wil MP van der Aalst. Online conformance checking: relating event streams to process models using prefix-alignments. *International Journal of Data Science and Analytics*, 8:269–284, 2019.

[87] Andrea Burattin and Josep Carmona. A framework for online conformance checking. In Ernest Teniente and Matthias Weidlich, editors, *Business Process Management Workshops*, pages 165–177, Cham, 2018. Springer International Publishing.

[88] Andrea Burattin, Sebastiaan J van Zelst, Abel Armas-Cervantes, Boudewijn F van Dongen, and Josep Carmona. Online conformance checking using behavioural patterns. In *Business Process Management: 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9–14, 2018, Proceedings 16*, pages 250–267. Springer, 2018.

[89] Florian Stertz, Juergen Mangler, and Stefanie Rinderle-Ma. Data-driven improvement of online conformance checking. In *2020 IEEE 24th international enterprise distributed object computing conference (EDOC)*, pages 187–196. IEEE, 2020.

[90] Christoph Rinner, Emmanuel Helm, Reinhold Dunkl, Harald Kittler, and Stefanie Rinderle-Ma. Process mining and conformance checking of long running processes in the context of melanoma surveillance. *International journal of environmental research and public health*, 15(12):2809, 2018.

[91] Reinhold Dunkl, Karl Anton Fröschl, Wilfried Grossmann, and Stefanie Rinderle-Ma. Assessing medical treatment compliance based on formal process modeling. In *Information Quality in e-Health: 7th Conference of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2011, Graz, Austria, November 25-26, 2011. Proceedings 7*, pages 533–546. Springer, 2011.

[92] David Knuplesch, Linh Thao Ly, Stefanie Rinderle-Ma, Holger Pfeifer, and Peter Dadam. On enabling data-aware compliance checking of business process models. In *Conceptual Modeling–ER 2010: 29th International Conference on Conceptual Modeling, Vancouver, BC, Canada, November 1-4, 2010. Proceedings 29*, pages 332–346. Springer, 2010.

[93] Linh Thao Ly, Stefanie Rinderle-Ma, David Knuplesch, and Peter Dadam. Monitoring business process compliance using compliance rule graphs. In *On the Move to Meaningful Internet Systems: OTM 2011: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011, Proceedings, Part I*, pages 82–99. Springer, 2011.

[94] Linh Thao Ly, David Knuplesch, Stefanie Rinderle-Ma, Kevin Göser, Holger Pfeifer, Manfred Reichert, and Peter Dadam. Seaflows toolset–compliance verification made easy for process-aware information systems. In *Information Systems Evolution: CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers 22*, pages 76–91. Springer, 2011.

[95] Sebastian Neubert, Bernd Göde, Xiangyu Gu, Norbert Stoll, and Kerstin Thurow. Potential of laboratory execution systems (less) to simplify the application of business process management systems (bpmss) in laboratory automation. *SLAS TECHNOLOGY: Translating Life Sciences Innovation*, 22(2):206–216, 2017.

[96] Silke Holzmüller-Laue, Bernd Göde, and Kerstin Thurow. Model-driven complex workflow automation for laboratories. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 758–763. IEEE, 2013.

[97] Robin A Felder. Automated specimen inspection, quality analysis, and its impact on patient safety: beyond the bar code, 2014.

[98] Hoi-Ying Elsie Yu, Harold Lanzoni, Tracy Steffen, Warren Derr, Kim Cannon, Jeanene Contreras, and Jordan Erik Olson. Improving laboratory processes with total laboratory automation. *Laboratory Medicine*, 50(1):96–102, 2019.

[99] Rainer Heidrich. Automation in small labs. *Journal of Laboratory Medicine*, 45(4-5):237–240, 2021.

[100] Shantani KANNAN, Kannan SUBBARAM, ALI Sheeza, and Hemalatha KANNAN. A systematic review with narrative synthesis on medical robotics and laboratory automation in the control of sars-cov-2, ebola and h1n1 (swine flu) viruses. *Journal of Health and Social Sciences*, 5(2):193–208, 2020.

[101] Robert Hawkins. Managing the pre-and post-analytical phases of the total testing process. *Annals of laboratory medicine*, 32(1):5–16, 2012.

[102] Michael Laposata and Anand Dighe. "pre-pre" and "post-post" analytical error: high-incidence patient safety hazards involving the clinical laboratory. 2007.

[103] AK Stroobants, HMJ Goldschmidt, and M Plebani. Error budget calculations in laboratory medicine: linking the concepts of biological variation and allowable medical errors. *Clinica chimica acta*, 333(2):169–176, 2003.

[104] Gabriel Lima-Oliveira, Waldemar Volanski, Giuseppe Lippi, Geraldo Picheth, and Gian Cesare Guidi. Pre-analytical phase management: a review of the procedures from patient preparation to laboratory analysis. *Scandinavian journal of clinical and laboratory investigation*, 77(3):153–163, 2017.

[105] Annalise E Zemlin. Errors in the extra-analytical phases of clinical chemistry laboratory testing. *Indian Journal of Clinical Biochemistry*, 33:154–162, 2018.

[106] Lirong Zou, Feng Ruan, Mingxing Huang, Lijun Liang, Huitao Huang, Zhongsi Hong, Jianxiang Yu, Min Kang, Yingchao Song, Jinyu Xia, Qianfang Guo, Tie Song, Jianfeng He, Hui-Ling Yen, Malik Peiris, and Jie Wu. Sars-cov-2 viral load in upper respiratory specimens of infected patients. *New England Journal of Medicine*, 382(12):1177–1179, 2020.

[107] David M Goldfarb, Peter Tilley, Ghada N Al-Rawahi, Jocelyn A Srigley, Geoffrey Ford, Heather Pedersen, Abhilasha Pabbi, Stephanie Hannam-Clark, Marthe Charles, Michelle Dittrick, et al. Self-collected saline gargle samples as an alternative to health care worker-collected nasopharyngeal swabs for covid-19 diagnosis in outpatients. *Journal of clinical microbiology*, 59(4):10–1128, 2021.

[108] Centers for disease control and prevention (cdcp). interim laboratory biosafety guidelines for handling and processing specimens associated with coronavirus disease 2019. `https://www.cdc.gov/coronavirus/2019-ncov/lab/lab-biosafety-guidelines.html`, 2019. [Last accessed: 2024-06-01].

[109] World health organization (who). laboratory testing for coronavirus disease (covid-19) in suspected human cases - interim guidance. `https://iris.who.int/bitstream/handle/10665/331501/WHO-COVID-19-laboratory-2020.5-eng.pdf?sequence=1`, 2020. [Last accessed: 2024-06-01].

[110] I Made Artika, Ageng Wiyatno, and Chairin Nisa Ma'roef. Pathogenic viruses: Molecular detection and characterization. *Infection, Genetics and Evolution*, 81:104215, 2020.

[111] Ollie Yiru Yu, Irene Shuping Zhao, May Lei Mei, Edward Chin-Man Lo, and Chun-Hung Chu. Dental biofilm and laboratory microbial culture models for cariology research. *Dentistry journal*, 5(2):21, 2017.

[112] Huseyin Tombuloglu, Hussein Sabit, Hamoud Al-Khallaf, Juma H Kabanja, Moneerah Al-saeed, Najat Al-Saleh, and Ebtesam Al-Suhaimi. Multiplex real-time rt-pcr method for the diagnosis of sars-cov-2 by targeting viral n, rdrp and human rp genes. *Scientific Reports*, 12(1):2853, 2022.

[113] Manit Arya, Iqbal S Shergill, Magali Williamson, Lyndon Gommersall, Neehar Arya, and Hitendra RH Patel. Basic principles of real-time quantitative pcr. *Expert review of molecular diagnostics*, 5(2):209–219, 2005.

[114] Michael R Tom and Michael J Mina. To interpret the sars-cov-2 test, consider the cycle threshold value, 2020.

[115] Soha Al Bayat, Jesha Mundodan, Samina Hasnain, Mohamed Sallam, Hayat Khogali, Dina Ali, Saif Alateeg, Mohamed Osama, Aiman Elberdiny, Hamad Al-Romaihi, and Mohammed Hamad J. Al-Thani. Can the cycle threshold (ct) value of rt-pcr test for sars cov2 predict infectivity among close contacts? *Journal of Infection and Public Health*, 14(9):1201–1205, 2021.

[116] Sonia N Rao, Davide Manissero, Victoria R Steele, and Josep Pareja. A narrative systematic review of the clinical utility of cycle threshold values in the context of covid-19. *Infectious diseases and therapy*, 9:573–586, 2020.

[117] Bgbl. ii nr. 15/2020. `https://www.ris.bka.gv.at/Dokumente/BgblAuth/BGBLA_2020_II_15/BGBLA_2020_II_15.pdfsig`, 2020. [Last accessed: 2024-05-20].

[118] Constantine Evans. Qslib. **https://github.com/cgevans/qslib**, 2021. [Last accessed: 2024-02-12].

[119] Constantine Evans. Qslib documentation. **https://qslib.readthedocs.io/en/latest/index.html**, 2022. [Last accessed: 2024-02-14].

[120] Jürgen Mangler, Florian Pauker, Stefanie Rinderle-Ma, and Matthias Ehrendorfer. centurio.work ? industry 4.0 integration assessment and evolution. In *17th Int'l Conference on Business Process Management*, pages 106–117, 2019.

[121] Dirk Merkel et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, 239(2):2, 2014.

[122] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228, 2017.

[123] Anjali Chauhan. A review on various aspects of mongodb databases. *Int. J. Eng. Res. Sci. Technol*, 8(5):90–92, 2019.

[124] Alexandru Boicea, Florin Radulescu, and Laura Ioana Agapin. Mongodb vs oracle–database comparison. In *2012 third international conference on emerging intelligent data and web technologies*, pages 330–335. IEEE, 2012.

[125] Will Reese. Nginx: the high-performance web server and reverse proxy. *Linux Journal*, 2008(173):2, 2008.

[126] Rahul Soni. *Nginx*. Springer, 2016.

[127] Lambert M Surhone, Mariam T Tennoe, and Susan F Henssonow. Node. js, 2010.

[128] OpenJS Foundation. Node.js. **https://nodejs.org/en** [Last accessed: 2024-02-12].

[129] Brett Nelson. Getting to know vue. js. *Apress Media, August*, 2018.

[130] MDN contributors. Server-sent events. **https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events**, 2024. [Last accessed: 2024-02-25].

[131] WHATWG. 9.2 server-sent events. **https://html.spec.whatwg.org/multipage/server-sent-events.html**, 2024. [Last accessed: 2024-02-25].

[132] Misha Wolf and Charles Wicksteed. Date and time formats. *W3C NOTE-datetime-19980827, August*, 1998.

[133] Terence J. Parr and Russell W. Quong. Antlr: A predicated-ll (k) parser generator. *Software: Practice and Experience*, 25(7):789–810, 1995.

[134] Patrik Koenig, Juergen Mangler, and Stefanie Rinderle-Ma. Compliance monitoring on process event streams from multiple sources. In *2019 International Conference on Process Mining (ICPM)*, pages 113–120, 2019.

[135] Niklaus Wirth. Extended backus-naur form (ebnf). *ISO/IEC*, 14977(2996):2–21, 1996.

[136] Juergen Mangler and Stefanie Rinderle-Ma. Cloud process execution engine: Architecture and interfaces, 2022.

[137] Stefanie Rinderle-Ma, Manfred Reichert, and Martin Jurisch. On utilizing web service equivalence for supporting the composition life cycle. *International Journal of Web Services Research (IJWSR)*, 8(1):41–67, 2011.

[138] Linh Thao Ly, Stefanie Rinderle-Ma, Kevin Göser, and Peter Dadam. On enabling integrated process compliance with semantic constraints in process management systems: Requirements, challenges, solutions. *Information Systems Frontiers*, 14(2):195–219, 2012.

[139] Jana Koehler and Jussi Vanhatalo. Process anti-patterns: How to avoid the common traps of business process modeling. *IBM WebSphere Developer Technical Journal*, 10(2):4, 2007.

[140] Christina Walcherberger, Florian Holl, Markus Pollak, Nikolaus Kowarz, and Julia Partheymüller. Chronologie zur corona-krise in Österreich - teil 7: Der delta-lockdown, die omikron-welle und das “frühlingserwachen”. `https://viecer.univie.ac.at/corona-blog/corona-blog-beitraege/blog-150-chronologie-zur-corona-krise-in-oesterreich-teil-7-der-delta-lockdown-die-omikron-welle-und-das-fruehlingserwachen/`, 2022. [Last accessed: 2024-05-01].