Time-aware middleware for Blockchain-Based Business Process*

Asma Mâalej[1], Wael Sellami[1,2,3] and Hatem Hadj-Kacem[1,2,*]

[1]ReDCAD Laboratory, University of Sfax, Tunisia
[2]FSEG, University of Sfax, Tunisia
[3]Unit of Scientific Research, Applied College, Qassim University, Saudi Arabia

Abstract

Blockchain-Based Inter-organizational business processes often face significant challenges in managing temporal constraints, which are important for ensuring timely and efficient execution due to the decentralized nature of blockchain and the complexities of coordinating multiple organizations. To address these challenges, this paper introduces an approach focusing on two solutions: a Retry Mechanism and a Controlled Extension of Allowed Time. The Retry Mechanism is designed to automatically re-execute tasks that surpass their time limits, providing a robust solution for maintaining process continuity in the face of delays or failures. The Controlled Extension of Allowed Time offers a flexible yet disciplined approach to extending task deadlines under specific conditions, balancing the need for adaptability with the importance of maintaining process integrity.

Our proposed solutions not only improve the management of temporal constraints but also establish a strong foundation for future advancements. Specifically, we explore the potential of parallel processing, where tasks are divided into subtasks and executed concurrently across multiple processors, thereby accelerating overall process execution and ensuring stricter adherence to time constraints. This paper contributes to the advancement of temporal constraint management, offering practical strategies for improving the performance and reliability of complex business processes across organizational boundaries.

Keywords

Inter-organizational business process,, Blockchain, middleware, Temporal Constraints

1. Introduction

In today's interconnected world, Inter-Organizational Business Processes (IOBPs) [1] play an important role in enabling collaboration between different organizations to achieve common business goals [2]. These processes involve the coordination of tasks and data across various entities, often operating under distinct policies, regulations, and systems. As these processes span multiple organizations, ensuring consistency, transparency, and security becomes a significant challenge. In this context, Blockchain technology has emerged as a powerful solution due to its decentralized nature, immutability, and ability to provide transparent, tamper-proof records of

TACC'2024, The 4th Tunisian-Algerian Conference on applied Computing, December 17-18, 2024, Constantine, Algeria
*Corresponding author.
maalejasma01@gmail.com (A. Mâalej); wael.sellami@gmail.com (W. Sellami); hatem.hadjkacem@redcad.org (H. Hadj-Kacem)

transactions. By leveraging blockchain, organizations can streamline IOBPs, ensuring trust and accountability without relying on a central authority.

The combination of IOBPs and blockchain offers new avenues for enhancing business process management. Blockchain's decentralized ledger ensures the integrity and traceability of inter-organizational transactions [3], while its smart contract capabilities can automate tasks and enforce business rules across organizational boundaries. However, integrating blockchain into IOBPs also introduces several challenges, particularly when it comes to managing temporal constraints [4]. This becomes especially problematic in time-sensitive business processes, where exceeding time constraints can result in process violations or costly failures. Therefore, addressing these temporal challenges is crucial to fully realizing the potential of blockchain-based IOBPs.

This study presents a time-sensitive approach to address the temporal challenges in blockchain-based IOBPs, ensuring the execution of tasks within defined time limits. To implement this approach, we developed a middleware solution, which was then applied to a real-world case study to demonstrate its effectiveness.

The paper is organized as follows: In the next section we present Temporal Inter-Organizational Business Process in Blockchain. In section 3, we give a review of the related works. Next, we describe, in section 4 our proposed approach. Section 5 is dedicated to the implementation and evaluation. We will apply a case study focused on an insurance company. This case study will serve to demonstrate the practical application and effectiveness of our proposed approach within the context of insurance management, allowing us to evaluate its real-world viability and impact. Finally, section 6 contains conclusion and plans for future work.

2. Temporal Inter-Organizational Business Process in Blockchain

An **Inter-Organizational Business Process (IOBP)** refers to a set of coordinated activities carried out by two or more independent organizations to achieve a common business objective. Unlike intra-organizational processes, which occur within a single entity, IOBPs require seamless collaboration across organizational boundaries, involving the exchange of information, resources, and services, as well as coordination through mechanisms such as shared platforms, communication tools, or contracts. These processes are critical for managing complex supply chains [5], partnerships, and collaborative enterprises, but they face significant challenges such as trust, data sharing, and governance due to differing organizational priorities and systems. Blockchain technology has emerged as a potential solution to some of these challenges. As a data structure, **blockchain** [6] consists of a linked list of blocks, each containing a set of transactions that are cryptographically linked to the previous one, ensuring immutability and security. This structure is replicated across a decentralized network of full nodes, promoting trust by eliminating the need for a central authority. While blockchain offers benefits [7] such as transparency and enhanced security, it also introduces complexities when managing time-sensitive processes in IOBPs. Temporal constraints, which require tasks to be completed within specific time limits, are difficult to manage in a blockchain context due to factors such as consensus delays, network latency, and limited transaction throughput.

While blockchain provides significant benefits for managing inter-organizational business processes, it also presents challenges when handling temporal constraints—the scheduling and order of events that are important for the successful execution of these processes. In many IOBPs, tasks must be performed within strict timeframes or follow a specific sequence to ensure smooth operations and prevent disruptions.

The aim of our research is to propose solutions that can improve the efficiency and reliability of these processes under strict time requirements.

3. Related work

This section provides a comprehensive overview of related work in the stream of Temporal Constraints based Business Processes research. The verification of temporal constraints in business processes have been a long-researched topic (see [8] for an updated survey). But, there has been limited research on implementing these constraints in blockchain-based process execution. Notable frameworks like Caterpillar [9, 10], and others do not address them at all. However, there are some exceptions.

The approach proposed in [11] extends the Caterpillar tool to enable the automatic transformation of temporal constraints for business process models into smart contract code . Specifically, it adds support for transforming temporal constraints such as duration, temporal constraints over cardinality, temporal dependency, and start/end temporal constraints into Solidity code that can be executed on the Ethereum blockchain. This approach integrates time-guards inside the functions implementing activities with temporal constraints, eliminating the need for external timer services. This allows business processes with temporal requirements to be executed in a trustless blockchain environment while enforcing time constraints and avoiding potential violations or delays. The work of [12] propose a holistic framework for implementing temporal constraints in blockchain-based business process execution. The authors identify challenges with enforing temporal contraints on blockchain and propose several alternative measures to facilitate them. Specifically, they introduce five time measures available in blockchain systems: block number, parameter approach, storage oracle and request response oracle. The paper systematically compares these measures on properties like accuracy, trust and cost. It then evaluates how suitable the measures are for implementing different types of temporal constraints commonly found in process models, such as absolute timers, relative timers, cycles, and deferred choice. The goal is to provide guidelines on choosing appropriate measures for specific scenarios when developing blockchain-based process execution platforms. In [13], the authors propose TimeAwareBPMN-js, a web-based graphical editor that enables the modeling and verification of time-aware BPMN processes. The tool allows users to create and edit BPMN models enriched with temporal constraints, such as contingent durations and conditions. It supports a subset of BPMN elements enhanced with temporal attributes and allows the specification of relative constraints between flow objects. The application features a modular architecture that facilitates the integration of different verification plug-ins. As a proof-of-concept, it includes a CSTNU plug-in that verifies the dynamic controllability property by converting the time-aware BPMN model into an equivalent CSTNU instance. This approach enables researchers and practitioners to model, analyze, and verify temporal aspects of business processes in an integrated environment.

The approach proposed in [14] involves implementing a blockchain-based case study of organ transfer by healthcare drone delivery. The authors extended the Caterpillar blockchain-based process execution engine to support temporal constraints in smart contracts. Key steps include modeling the organ transfer process with temporal constraints, generating smart contracts from the process model, and executing the process on the Ethereum blockchain. This approach aims to validate the feasibility of incorporating temporal constraints like deadlines and durations into blockchain-based business process execution. The case study demonstrates how the extended Caterpillar system can handle time-sensitive processes with multiple actors, while leveraging blockchain to secure data sharing and traceability.

In [15], the proposed approach involves a Blockchain-based system for monitoring business processes. It utilizes a three layered architecture consisting of the Process Level, Monitoring Level, and Smart Contract in Blockchain Level. It aims to leverage blockchain's capabilities to enhance trust, security, and automation in business process monitoring. While this approach effectively monitors workflows, it does not consider the temporal aspect of business processes, which is crucial for managing time-sensitive operations and ensuring that tasks are completed within defined time limits.

Haarmann, in [16] aims to analyze the duration of blockchain-based business processes using simulation techniques. The method involves creating a timed Petri net representation of the inter-organizational process, translating each interaction task from a choreography model into corresponding transitions in the Petri net. The approach incorporates blockchain-specific configurations, including diffusion time, inter-block time, and confirmation count, to simulate the blockchain's transaction processing. Using CPN Tools to model and simulate the Petri net, the method allows for quantitative analysis and manual step-through simulation techniques. This simulation-based approach enables stakeholders to estimate the process execution time for various blockchain implementations by simply adjusting the blockchain configuration parameters.

In [17], authors presents a constraint satisfaction method for modeling and analyzing temporal process constraints with the ability to handle controlled violations. The method can handle various temporal patterns including basic duration and gap constraints.

Although these studies introduce innovative perspectives, they all have some limitations. Some of them, while they focuses on integrating Business processes with Blockchain, they may neglect the temporal perspective. Others, However our research aims to address these gaps.

4. Proposed approach

In this section, we present our proposed approach for managing inter-organizational business processes using a middleware solution based on temporal constraints. Our approach is designed to address the complexities and dynamic nature of collaborative workflows across multiple organizations. By integrating blockchain technology and linear programming, our architecture provides a robust framework that ensures both transparency and efficiency in process execution. The design schema of our approach, depicted in Figure 1, illustrates the core components and their interactions within the system.

Process level: The system incorporates temporal constraints that ensure all operations respect

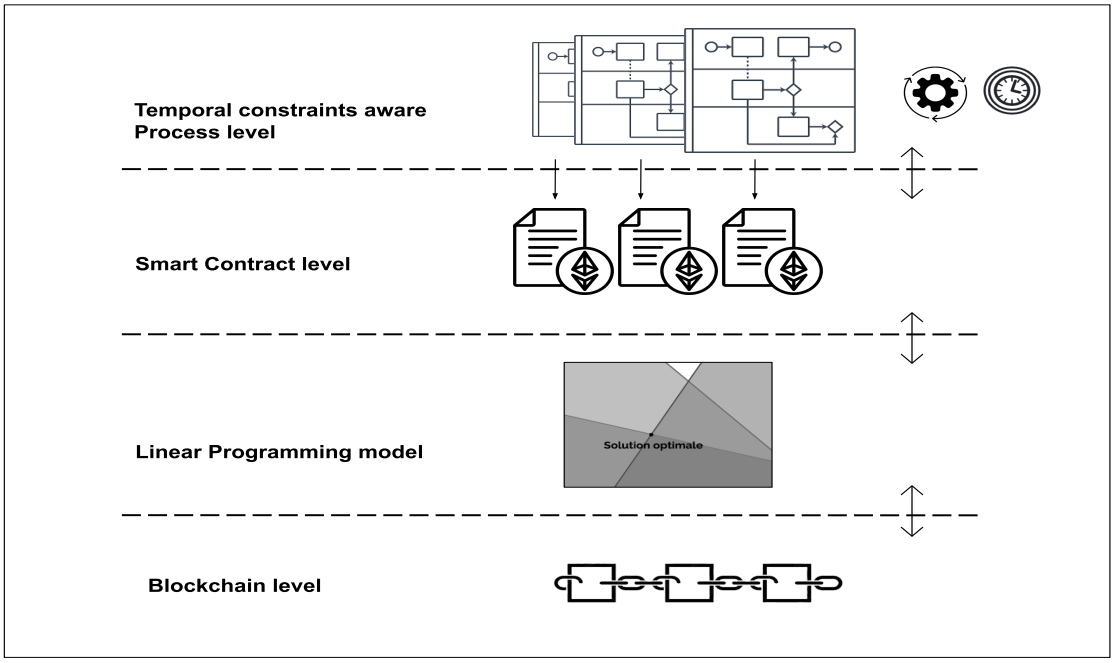


Figure 1: Overview of the Temporal Constraints Based approach architecture

predefined time windows. This is important for business processes where certain actions must be completed within specified deadlines. These constraints are modeled in process diagrams and later translated into smart contracts.

Smart contract level: Once the process logic is defined with temporal constraints, it is translated into smart contracts [18]. Each business process is associated with a smart contract, which functions as a self-executing script residing on the blockchain.

Linear Programming level: Linear programming provides a framework to optimize a given objective, such as minimizing costs or maximizing efficiency, subject to a set of linear constraints. This method is especially valuable when dealing with limited resources while satisfying demand, balancing workloads, or adhering to capacity limits. By expressing the problem mathematically, linear programming ensures that decisions are made systematically, with the best possible outcome based on the established parameters.

In the context of processing activities, an optimization model can be formulated to minimize the total cost of processing, while accounting for budget restrictions, capacity limits, and penalties associated with unprocessed or delayed activities. The following optimization model utilizes linear programming to determine the optimal number of accepted, rejected, and unprocessed activities during specific time slots, ensuring that the system operates within its capacity and budget while minimizing overall costs.

Objective function :

$$\text{Min } z = \sum_{t \in T} C_t^A \cdot I_t^A + \sum_{t \in T} C_t^R \cdot I_t^R + \text{Penalty} \cdot \sum_{w \in \Omega} UFR_t(w) \tag{1}$$

Table 1
Input Data

Notation	Description
C_t^A	Cost of processing an accepted activity during time slot t
C_t^R	Cost of processing a rejected activity during time slot t
MaxBudget	Maximum allocated budget for processing
$\max_t$	Maximum processing capacity during time slot t
$f_q^A(w)$	Number of accepted activities processed in scenario w
$f_q^R(w)$	Number of rejected activities processed in scenario w
$r_r^t(w)$	Number of received activities during time slot t for scenario w
Penalty	Penalty for unprocessed or incorrectly handled activities

Table 2
Decision Variables

Notation	Description
I_t^A	Number of accepted activities processed during time slot t
I_t^R	Number of rejected activities processed during time slot t
$UFR_t(w)$	Number of unprocessed activities in scenario w

Subject to:

- $z \leq \text{MaxBudget}$ (2)
- $I_t^A + I_t^R \leq \max_t, \forall t \in T$ (3)
- $rr_t(w) = I_t^A + I_t^R + UFR_t(w), \forall t \in T, \forall w \in \Omega$ (4)
- $I_t^A \geq 0, I_t^R \geq 0, UFR_t(w) \geq 0, \forall t \in T, \forall w \in \Omega$ (5)

Where, the objective function (1) consists in minimizing the total cost of processing activities, including penalties for unprocessed activities. Constraint (2) states that the total processing cost cannot exceed the maximum allocated budget. Constraint (3) insists that the total number of processed activities must be less than or equal to the processing capacity for each time slot. Constraint (4) ensures that all received activities are either processed (accepted or rejected) or

left unprocessed. Constraint (5) defines the non-negativity constraints for decision variables.

Blockchain level : blockchain layer offers a decentralized and secure infrastructure for executing smart contracts, ensuring the enforcement of each process and its associated temporal constraints. By making these operations immutable and tamper-resistant, the blockchain guarantees that all actions are transparent, reliable, and protected from external interference or manipulation. This layer plays a key role in maintaining the integrity of the entire system, providing a trusted environment where time-sensitive operations can be carried out with full transparency and security.

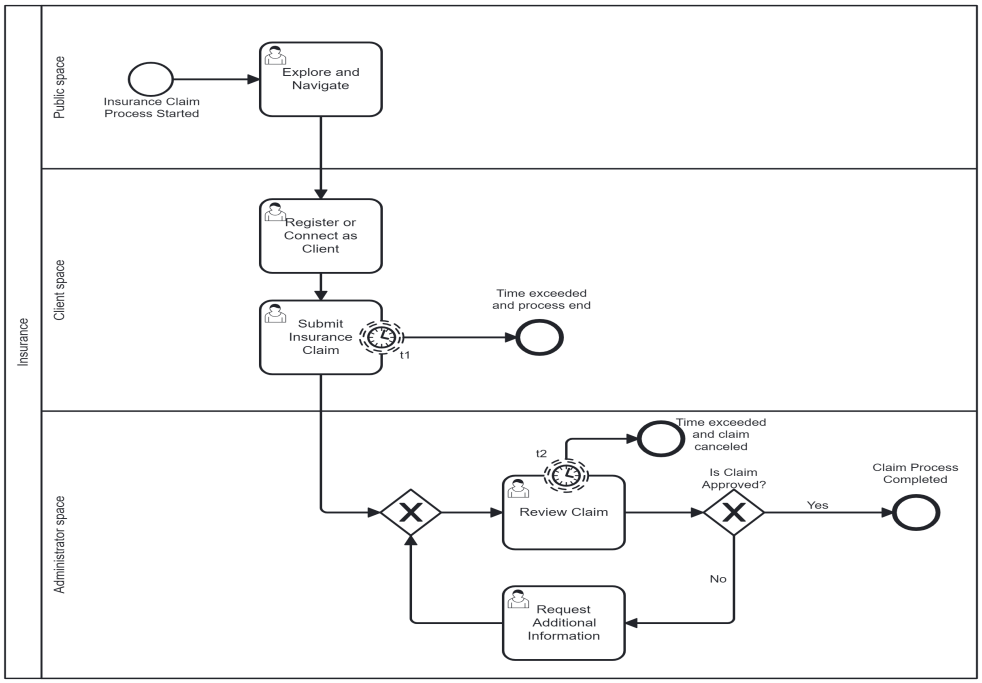


Figure 2: BPMN Diagram for Insurance process

5. Implementation and evaluation

5.1. Case study

In this case study, we present a BPMN diagram (see Figure 2) developed by Camunda [19] that maps the core processes within an insurance system. The diagram highlights three distinct spaces: the Public Space, the Client Space, and the Administrative Space, each representing different stages of interaction between the client and the insurance company. This BPMN model demonstrates how insurance companies can optimize these processes for greater efficiency and client satisfaction.

- **Public Space:** This section represents the initial touchpoint for users exploring insurance

options. Users can explore and navigate the insurance offerings on the platform but have not yet committed to any formal actions like creating an account or submitting claims.

- **Client Space:** This space represents the part of the process where the user becomes an active participant by registering or logging in as a client. Once authenticated, the client has access to the system's claim submission features.
- **Administrator Space:** Once a claim is submitted by the client, the administrative process begins. The administrator reviews it to determine its validity and decide whether additional information is needed from the client. If all conditions are met and the claim is approved, the process is concluded. If the claim review takes too long, it can result in automatic cancellation.

5.2. Implementation

To clarify our methodology and validate our approach, we detail the implementation of an application, as it is illustrated in Figure 3. It illustrates the architecture of a time-aware middleware that integrates BPMN with blockchain technology. The process begins with a BPMN model that defines the workflow of business processes. This model is fed into Caterpillar which converts it into smarts contracts.

Caterpillar serves as an innovative bridge, linking Camunda with the Ethereum blockchain. This functionality is important for our project, as it enables business processes, meticulously modeled in Camunda, to be transformed into executable smart contracts on the Ethereum blockchain. These smart contracts are developed using the Solidity programming language [20]. Rather than being a simple conversion, this transformation involves an intelligent reinterpretation of business logic into blockchain programming instructions, ensuring that the original process's consistency and integrity are maintained. This approach facilitates seamless integration and streamlines the transition from BPM theory to blockchain implementation.

The middleware plays an important role in facilitating essential functionalities, such as data management, which ensures the availability and organization of relevant information; claim management, which handles claims within the smart contract context; and notifications management, which manages alerts related to the contract's lifecycle. This integration provides a secure and automated approach to managing business processes, with the added benefit of time-awareness, ensuring timely execution and accountability within blockchain environments.

In the case of this insurance process, several temporal and operational constraints are essential for maintaining the workflow's integrity and efficiency. First of all, the process requires the customer to register or connect to the system before accessing insurance options.

In the case of claims submission, the process becomes more intricate. The validity of the claim must be checked within 48 hours after submission, and if additional information is required, the customer must provide it. Any delay beyond this period results in automatic claim rejection.

These constraints ensure that the entire process remains streamlined and secure, minimizing the risk of delays or fraudulent activities. By defining such temporal dependencies and operational rules, the system is better equipped to handle multiple customer interactions efficiently, without the need for intermediaries, while maintaining strict adherence to the business rules.

Now, we present our contribution by comparing it with existing works. We observe that in cases of temporal constraint violations, the *notifyDesigner()* function alone is not sufficient. To

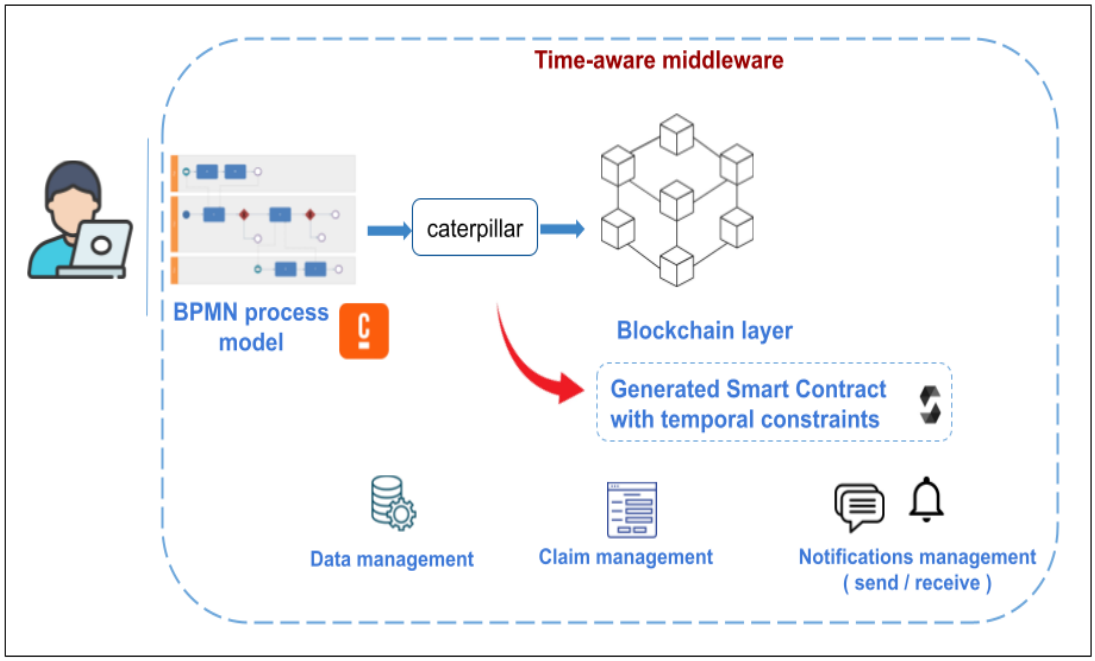


Figure 3: Overview of the Time-aware middleware architecture

address this, we propose two approaches based on the task at hand. The first is **the Retry Mechanism**, where the task is automatically retried if it exceeds the time limit. This approach is useful for processes that need to be reexecuted in case of failure or time overruns. The second approach is to **Extend the Allowed Time (with limits)**, where limited flexibility is introduced by extending the maximum allowed time if the task is near completion. However, this should only be done under specific conditions to prevent temporal constraint violations from becoming a recurring issue.

```
if (duration > minTime || duration < maxTime) {
    notifyDesigner();
    retryTask(); }
```

Listing 1: Solidity code in case of the "Retry" function in smart contract

```
function retryTask() internal {
    require(retryCount < maxRetries, "Max retries exceeded, aborting task.");
    retryCount += 1;
    // Adding a delay (1 min) before retrying the task
    uint retryDelay = 1 minutes;
    uint retryStartTime = block.timestamp;

    if (block.timestamp >= retryStartTime + retryDelay) {
        taskA(minTime, maxTime); }}
```

Listing 2: Solidty code of the RetryTask() function in smart contract

```
if (duration > minTime || duration < maxTime) {
    notifyDesigner();
    if (canExtendTime(duration)) {extendTime();}
    else {revert("Time constraint violated, process aborted."); }}
```

Listing 3: Solidity Code in case of the "extendTime" function in smart contract

Optimization model

The aim is to minimize the overall claim processing costs within the constraints of a given budget and processing capacity. Specifically, the model considers both accepted and rejected claims, assigning distinct costs to each for a given time slot. The uncertain parameters, including the number of claims received and the scenario-specific outcomes for claims processed or rejected, are encapsulated in our model to account for the variability in demand.

Table 3
Input Data

Notation	Description
C_t^A	Cost of processing an accepted claim for the time slot
C_t^R	Cost of processing a rejected claim for the time slot
MaxBudget	Maximum budget allocated for claim processing
max_t	Maximum processing capacity for claims during the time slot t
$fq^A(w)$	Number of claims successfully processed for scenario w
$fq^R(w)$	Number of claims rejected for scenario w
$rr_t(w)$	Number of claims received for the time slot t for scenario w
Penalty	Penalty for unprocessed or incorrectly handled claims

Decision variables, such as the number of accepted and rejected claims processed during each time slot, are used to determine the optimal allocation of resources. Moreover, the model incorporates penalties for unprocessed or incorrectly handled claims. Table 3 presents the notations of the parameters and Table 4 presents the decision variables of the optimization model.

Objective function :

$$\text{Min } z = \sum_{t \in T} C_t^A \cdot I_t^A + \sum_{t \in T} C_t^R \cdot I_t^R + \text{Penalty} \cdot \sum_{w \in \Omega} UFR_t(w) \tag{1}$$

Subject to:

Table 4
Decision variables

Notation	Description
I_t^A	Number of accepted claims processed for the time slot t
I_t^R	Number of rejected claims processed for the time slot t
$UFR_t(w)$	Number of unprocessed claims for scenario w

- $z \leq \text{MaxBudget}$ (2)
- $I_t^A + I_t^R \leq \max_t, \forall t \in T$ (3)
- $rr_t(w) = I_t^A + I_t^R + UFR_t(w), \forall t \in T, \forall w \in \Omega$ (4)
- $I_t^A \geq 0, I_t^R \geq 0, UFR_t(w) \geq 0, \forall t \in T, \forall w \in \Omega$ (5)

where, objective function (1) consists in minimizing the total cost of processing claims, including penalties for unprocessed claims. Constraint (2) states that the total processing cost cannot exceed the maximum allocated budget. Constraint (3) insists that the total number of processed claims (accepted or rejected) must be less than or equal to the processing capacity for each time slot. Constraint (4) ensures that all received claims are either processed (accepted or rejected) or left unprocessed. Constraint (5) is the non-negativity constraints for decision variables.

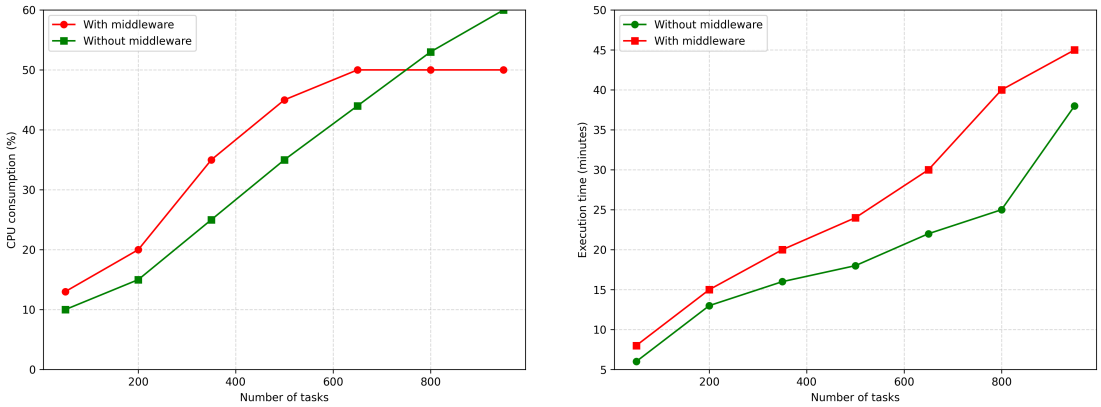
5.3. Evaluation

To evaluate the performance of our proposed time-aware middleware, we executed a number of tests on an insurance process. The goal of the evaluation is to compare the process execution time before and after applying the middleware, focusing on its ability to handle temporal constraints effectively.

We evaluated the process using different sets of tasks, ranging from 50 to 950 tasks. For each set, we measured the total execution time both without the middleware and with the middleware applied. The results are presented in the form of a line graph (Figure 4a), where the X-axis represents the number of tasks in the process and the Y-axis represents the execution time in minutes.

The first evaluation metric is CPU consumption, which reflects how efficiently system resources are used during the execution of the process. Figure 4a presents a comparison between CPU consumption for the process with and without the middleware. The X-axis represents the number of tasks, and the Y-axis represents the percentage of CPU utilization.

As shown in Figure 4a, processes without the middleware consume significantly more CPU than those with the middleware. This result highlights one of the key benefits of the middleware:



(a) CPU consumption curve

(b) Average of process execution time

Figure 4: Evaluation

its ability to optimize resource usage by intelligently managing retries, extensions, and parallel executions, which reduces the overall load on the system. The reduced CPU consumption suggests that the middleware provides more efficient processing, making it suitable for large-scale or resource-constrained environments.

The second metric is execution time, measured to determine how the middleware impacts the overall duration of the process. Figure 4b provides a comparison of execution times with and without the middleware. The X-axis represents the number of tasks in the process, and the Y-axis represents the execution time in minutes.

In contrast to CPU consumption, the results show that processes with the middleware take slightly more time to execute than those without it. This is due to the middleware's additional mechanisms, such as retrying failed tasks and extending the allowed time for near-completion tasks, which introduce a small overhead. However, this slight increase in execution time is a trade-off for ensuring that processes adhere to strict temporal constraints, which is critical for time-sensitive operations like insurance claims processing.

6. Conclusion and future work

In this paper, we have explored novel approaches to handling temporal constraints in inter-organizational business processes. Our proposed solution extends beyond the limitations of existing methods by incorporating a Retry Mechanism and a controlled Extension of Allowed Time. The Retry Mechanism provides a practical solution for handling time overruns by re-executing tasks automatically, which is important for processes where failure recovery is essential. The Extension of Allowed Time introduces a level of adaptability, allowing tasks that are nearing completion to benefit from extended deadlines under predefined conditions, thus mitigating the risk of habitual constraint violations.

Moreover, our proposed future work aims to ensure that tasks are completed within the defined time limits by executing them in parallel across multiple available processors. This

involves breaking down a task into smaller subtasks that can run simultaneously, thereby speeding up the overall process. By utilizing parallel processing, we intend to optimize task execution times and improve the efficiency of the entire system.

7. Declaration on Generative AI and AI-assisted Technologies

During the preparation of this work, the authors used X-GPT-4 for grammar and spelling checks. After using this tool, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] K. Bouchbout, Z. Alimazighi, Inter-organizational business processes modelling framework, in: ADBIS (2), 2011, pp. 45–54.

[2] Object Management Group, Business Process Model and Notation (BPMN) Version 2.0.2, 2014. URL: https://www.omg.org/spec/BPMN/2.0.2/PDF, [Accessed 7 July 2021].

[3] J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar, et al., Blockchains for business process management-challenges and opportunities, ACM Transactions on Management Information Systems (TMIS) 9 (2018) 1–16.

[4] J. Eder, E. Panagos, M. Rabinovich, Workflow Time Management Revisited, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 207–213. URL: https://doi.org/10.1007/978-3-642-36926-1_16. doi:10.1007/978-3-642-36926-1_16.

[5] H. Stadtler, Supply chain management: An overview, Supply chain management and advanced planning: Concepts, models, software, and case studies (2014) 3–28.

[6] M. Di Pierro, What is the blockchain?, Computing in Science & Engineering 19 (2017) 92–95.

[7] G. Habib, S. Sharma, S. Ibrahim, I. Ahmad, S. Qureshi, M. Ishfaq, Blockchain technology: benefits, challenges, applications, and integration of blockchain technology with cloud computing, Future Internet 14 (2022) 341.

[8] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, The temporal perspective in business process modeling : An evaluative survey and research challenges, Service Oriented Computing and Applications 9 (2014) 75–85. doi:10.1007/s11761-014-0170-x.

[9] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, Caterpillar: A blockchain-based business process management system., BPM (Demos) 172 (2017) 1–5.

[10] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, A. Ponomarev, Caterpillar: A business process execution engine on the ethereum blockchain, Software: Practice and Experience 49 (2018) 1162 – 1193. URL: https://api.semanticscholar.org/CorpusID:51967088.

[11] A. Abid, S. Cheikhrouhou, M. Jmaiel, Modelling and executing time-aware processes in trustless blockchain environment, in: S. Kallel, F. Cuppens, N. Cuppens-Boulahia, A. Hadj Kacem (Eds.), Risks and Security of Internet and Systems, Springer International Publishing, Cham, 2020, pp. 325–341.

[12] J. Ladleif, M. Weske, Time in blockchain-based process execution, in: 2020 IEEE 24th

International Enterprise Distributed Object Computing Conference (EDOC), 2020, pp. 217–226.

[13] M. Ocampo-Pineda, R. Posenato, F. Zerbato, Timeawarebpmn-js: An editor and temporal verification tool for time-aware bpmn processes, SoftwareX 17 (2022) 100939.

[14] A. Abid, S. Cheikhrouhou, S. Kallel, M. Jmaiel, Temporal constraints in smart contract-based process execution: A case study of organ transfer by healthcare delivery drone, in: A. H. Kacem, S. Kallel, F. Belala, M. Belguidoum, M. Jmaiel, I. B. Rodriguez (Eds.), Proceedings of the Tunisian-Algerian Joint Conference on Applied Computing (TACC 2021), Tabarka, Tunisia, December 18-20, 2021, volume 3067 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 1–13. URL: https://ceur-ws.org/Vol-3067/paper1.pdf.

[15] R. Essid, W. Sellami, H. Hadj-Kacem, MW4BPM: A middleware for blockchain-based business process monitoring, in: S. Kallel, Z. Benzadri, A. H. Kacem (Eds.), Proceedings of the Tunisian-Algerian Joint Conference on Applied Computing (TACC 2023), Sousse, Tunisia, November 8-10, 2023, volume 3642 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 170–181. URL: https://ceur-ws.org/Vol-3642/paper15.pdf.

[16] S. Haarmann, Estimating the duration of blockchain-based business processes using simulation, in: Central-European Workshop on Services and their Composition, 2019, pp. 24–31. URL: https://api.semanticscholar.org/CorpusID:96426650.

[17] A. Kumar, R. R. Barton, Controlled violation of temporal process constraints–models, algorithms and results, Information Systems 64 (2017) 410–424.

[18] K. Delmolino, M. Arnett, A. Kosba, A. Miller, E. Shi, Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab, in: Lecture Notes in Computer Science, volume 9604, 2016, pp. 79–94.

[19] Camunda, bpmn-js: BPMN 2.0 Viewer and Editor, 2014. URL: https://bpmn.io/toolkit/bpmn-js/, [Accessed 7 July 2021].

[20] C. Dannen, C. Dannen, Solidity programming, Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners (2017) 69–88.