Block-Timer: A Layer-Optimized Approach for BPMN Process Execution with Time Constraints on Blockchain

Matteo Vaccargiu[1,3,*,†], Endri Xhina[2,†] and Roberto Tonelli[3,†]

[1]International School of Advanced Studies, University of Camerino, Italy
[2]Department of Mathematics and Computer Science, University of Tirana, Albania
[3]Department of Mathematics and Computer Science, University of Cagliari, Italy

Abstract

This paper presents Block-Timer, a layer-optimized approach for executing BPMN processes with time constraints on blockchain networks. Time-bound activities are core components of business processes, representing regulatory deadlines and service level agreements. While blockchain technology offers transparency and trustless coordination for process execution, implementing time constraints on blockchain presents significant technical challenges. Traditional blockchain platforms rely on vulnerable block timestamps, while executing complex processes entirely on Layer 1 faces economic and throughput limitations. We propose a conceptual hybrid-solution where a block-number-based implementation of timer events ensures secure, manipulation-resistant, time tracking without external dependencies. Our solution automatically translates BPMN processes with timer events into executable smart contracts, with a comparative analysis of execution costs between Ethereum (Layer 1) and Arbitrum (Layer 2) across three diverse case studies. The analysis reveals that Layer 2 implementation reduces operational costs compared to Layer 1 while maintaining functional equivalence. Based on these findings, we propose an hybrid architecture that strategically distributes process operations between layers according to security requirements and cost considerations. This approach optimizes the balance between security guarantees and economic feasibility for blockchain-based business process execution, particularly for multi-party workflows requiring both trust and efficiency. Our security analysis confirms the integrity of the block-based timer implementation with no critical vulnerabilities, demonstrating a conceptual solution for time-constrained process execution on blockchain networks.

Keywords

BPMN, Timer Events, Blockchain, Layer 1, Layer 2

CEUR-WS.org/Vol-4105/paper06.pdf

1. Introduction

Business Process Model and Notation (BPMN) has emerged as the standard for modeling organizational workflows, providing a graphical representation that bridges the gap between process design and implementation [1]. In today's business environment, time constraints are fundamental components of these processes, often manifested as regulatory deadlines, service level agreements, or operational timeframes. These time-bound activities are represented in BPMN as timer events, which trigger specific actions after predetermined periods or at scheduled timestamps.

Meanwhile, blockchain technology offers important capabilities for business process execution through immutable record-keeping, transparent operations, and trustless coordination between participants [2]. The integration of BPMN with blockchain technology creates new opportunities for automated, transparent, and verifiable process execution, particularly for multi-party workflows where trust and accountability are essential.

However, implementing time constraints on blockchain presents significant technical challenges [3]. Traditional blockchain platforms like Ethereum rely on block timestamps, which are vulnerable to manipulation by miners within certain bounds. Additionally, executing complex business processes entirely on Layer 1 blockchains faces practical limitations due to high transaction fees that make

DLT2025: 7th Distributed Ledger Technology Workshop, June, 12-14 2025 - Pizzo, Italy
*Corresponding author.
†These authors contributed equally.
matteo.vaccargiu@unica.it (M. Vaccargiu); endri.xhina@fshn.edu.al (E. Xhina); roberto.tonelli@unica.it (R. Tonelli)
0009-0009-9461-6183 (M. Vaccargiu); 0000-0002-7487-1795 (E. Xhina); 0000-0002-9090-7698 (R. Tonelli)
© 2025 2025 © Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2025-11-24

frequent operations economically unfeasible, especially during network congestion periods. Layer 1 blockchains also have inherent throughput limitations that restrict their ability to handle high-volume process execution, while block confirmation times can impact the timing precision required for critical business processes [4].

While Layer 2 scaling solutions like Arbitrum significantly reduce transaction costs through optimistic rollups, they introduce different security assumptions such as delayed finality through fraud-proof mechanisms rather than immediate consensus, and potential vulnerability to validator collusion [5]. These differences may not be appropriate for all operations within a business process, especially those requiring immediate finality or maximum decentralization. This creates a contrast between cost efficiency and security guarantees that must be balanced according to process requirements.

This paper presents an **approach** that addresses these challenges through a **block-number-based implementation of timer events** that ensures secure, manipulation-resistant time tracking without relying on external time services. We **automatically translate BPMN processes with timer events** into secure, executable Solidity smart contracts, along with a comparative analysis of execution costs between Ethereum (Layer 1) and Arbitrum (Layer 2) across three diverse case studies. From this analysis, we propose a **conceptual hybrid architecture** that strategically distributes process operations between Layer 1 and Layer 2 based on security requirements and cost considerations.

Our approach was validated through the implementation of three different BPMN case studies: a municipal document request process, an insurance policy application workflow, and an academic examination process. Each case demonstrates different patterns of timer event usage and inter-participant coordination. We provide a complete replication package including all the case study BPMN codes, the Java code of the automatic translator and the translated three smart contracts at **this link** to support reproducibility and verification.

The remainder of this paper is organized as follows: Section 2 provides background on blockchain time mechanisms and BPMN integration approaches. Section 3 details our methodology for block-based timer implementation and smart contract generation. Section 4 presents the implementation results and cost analysis across different blockchain layers. Section 5 introduces our proposed hybrid architecture. Section 6 discusses assumptions and threats of our study and section 7 implications, limitations, and concludes with future research directions.

2. Background and Related Work

2.1. Blockchain-Based Business Process Execution

Blockchain technology has emerged as a support for collaborative business processes among mutually untrusting parties in decentralized environments. Recent research has developed several model-driven approaches to implement business processes on blockchain platforms using standardized process modeling notations, particularly BPMN. Di Ciccio et al. [6] present the foundational principles of blockchain support for collaborative business processes, highlighting how blockchain's tamper-proof ledger can capture process history and current state while smart contracts can encode business logic and enforce process rules. The authors note that blockchain enables trustworthy process execution among partially trustable parties without requiring centralized coordinators, making it ideal for inter-organizational processes. They introduce two primary architectural approaches: using blockchain as a message exchange mechanism for choreographies, and deploying entire collaborative processes on-chain. Nousias et al. [7] address the challenges of developing and deploying Decentralized Applications (DApps) on blockchain networks by standardizing and modeling these processes through BPMN. Their approach begins with a Decision Model and Notation (DMN) decision model to help developers determine if a DApp is justified for their use case, followed by two detailed BPMN models: one for DApp development and another for DApp deployment. These models serve as a roadmap to enhance decision-making, facilitate developer communication, and reduce implementation time and cost. The authors demonstrate the practical application of their models by implementing a DApp for registering and verifying academic qualifications, illustrating how BPMN can be a powerful tool for systematic DApp

development. Other two key implementation frameworks for blockchain-based processes are: Lorikeet and Caterpillar. Lorikeet [8], developed by Data61, is a model-driven engineering tool for blockchain application development focusing on business processes and asset control. It generates Solidity smart contracts from BPMN models and asset data schemas, allowing code review and adaptation before deployment. Lorikeet extends BPMN with registry references and action invocations to interact with on-chain asset data stores, making it particularly suitable for processes involving asset management. Caterpillar, presented by López-Pintado et al.[9], takes a different approach by implementing the process execution logic entirely on-chain. It translates BPMN models into smart contracts that enforce business process rules and maintain process state on the blockchain. Caterpillar follows three design principles: modeling collaborative processes as if all parties shared the same execution infrastructure, storing complete process state and metadata on-chain, and enabling process execution to proceed even if off-chain components are unavailable. This approach provides stronger guarantees of process compliance but with higher on-chain storage costs. Corradini et al. [10] present an approach for modeling and executing BPMN choreographies with multiple instances on blockchain technology. The authors address the limitation of existing blockchain-based choreography implementations that typically focus on single-instance scenarios, which restricts their application in complex systems like IoT where multiple devices perform similar operations. Their solution introduces support for multi-instance participants and tasks in BPMN choreography diagrams, with clear semantics and implementation in smart contracts. The framework includes modeling capabilities for specifying multiplicity attributes, automatic translation of the models into Solidity code, and execution on blockchain platforms. To demonstrate feasibility, the authors implemented a Smart Thermostat case study on the Polygon blockchain, where a thermostat coordinates with multiple sensors and radiators. Their cost analysis showed the approach is practical in terms of blockchain transaction fees, particularly when using Layer 2 solutions like Polygon rather than Ethereum mainnet. The same authors propose in [11] a flexible approach that decouples process logic (stored off-chain in IPFS as Drools rules) from execution state (stored on-chain), enabling run-time modifications without compromising blockchain's trustworthiness properties. Their FlexChain framework supports voting mechanisms to ensure agreement on process changes, addressing the challenge of reconciling blockchain's immutability with the need for business process flexibility. A common challenge across these approaches is balancing on-chain and off-chain data and computation. On-chain execution ensures integrity and transparency but incurs higher costs and scalability limitations, while off-chain processing is more efficient but potentially less secure. Researchers are actively addressing these trade-offs through hybrid architectures that optimize process execution while maintaining essential trust guarantees. These implementations demonstrate the feasibility of using blockchain as a foundation for trustworthy, transparent execution of collaborative business processes, while continuing research focuses on improving flexibility, scalability, and cost-effectiveness of blockchain-based process execution environments.

2.2. Time Constraints in Blockchain Systems

Time management is a challenge in blockchain-based systems due to the decentralized and asynchronous nature of blockchain networks. Several approaches have been proposed to address these points. Eichinger and Ebermann [12] investigate the effectiveness of smart contracts in stipulating time constraints. They define "execution accuracy" as the probability that a smart contract execution yields the same result based on world time and injected time. Their experiments on local blockchain implementations (Ganache and Quorum) and public Ethereum test networks (Görli and Rinkeby) show that execution accuracy decreases near interval bounds. They identify two primary methods for implementing time constraints: the "block timestamp method", which uses timestamps set by miners, and the "parameter method", where transaction senders attach timestamps to transactions. Ladleif and Weske [3] present an analysis of temporal constraints in blockchain-based process execution. They identify five distinct time measures available in blockchain environments: block timestamp, block number, parameter, storage oracle, and request-response oracle. Each measure has different characteristics regarding accuracy, trust assumptions, immediacy, cost, and reliability. Their work indicates

that while absolute and relative temporal constraints can be implemented in blockchain, they require careful consideration of the underlying time measure used. For more complex time-based applications, Chae et al. [13] propose a "time-release blockchain" that integrates time-release cryptography with blockchain's proof-of-work mechanism. Their approach automatically adjusts for changes in network computing power over time and eliminates the need for trusted third parties in time-release applications. This enables practical implementation of systems where information must remain encrypted until a predetermined future time, with applications in areas such as electronic voting and sealed-bid auctions. These approaches demonstrate that while implementing time constraints in blockchain systems remains challenging, multiple viable methods exist with different trade-offs in terms of accuracy, security, and efficiency. The choice of approach depends on the specific requirements of the application and the characteristics of the underlying blockchain platform.

2.3. Operation Classification for Multi-Layer Execution

Several research efforts have explored the use of blockchain in multi-party processes, each contributing complementary insights into how operations can be classified and distributed across architectural layers for decentralized execution.

Argento et al. [14] introduce ID-Service, a blockchain-based accountability platform for cross-organizational workflows (COWs). The architecture relies on a two-module division: the Accountability Certification layer integrates identity management via eIDAS-compliant digital identities, while the Accountability Workflow layer handles the orchestration of inter-organizational processes using smart contracts. Each operational action is anchored to an authenticated identity and recorded immutably, enforcing non-repudiation. Notably, the synchronization points in workflows are modeled as critical transitions between actors, preserving causal ordering and traceability. This layered architecture effectively separates the tasks of identity anchoring and process execution, laying the groundwork for classifying blockchain operations according to system responsibility and trust boundaries.

A complementary perspective is proposed by Alfarisy [15], which addresses data collection and validation in a resource governance scenario. Here, a service-oriented architecture is used to develop an IoT-integrated blockchain platform for fishing quota enforcement. The system architecture spans four layers—device, network, service, and application—each abstracting a specific operational domain. Transactional data such as timestamps and fish weights are recorded via IoT devices and pushed through the service layers to the blockchain. The study provides quantitative assessments of cohesion, coupling, and reusability, showing that well-separated services with low interdependency are preferable for repeatability and maintainability. These principles validate the need for classifying operations based on abstraction and coordination layer, particularly in high-volume data acquisition scenarios.

Further abstraction is explored by Pirani et al. [16], where blockchain is embedded into a cyber-systemic methodology for governing hybrid reality (HyR) phenomena. Rather than focusing on domain-specific implementations, the paper proposes a formal framework that positions blockchain as a systemic trust infrastructure. The authors present a cybernetic modeling approach grounded in system dynamics and Ashby's Law of Requisite Variety, in which blockchain acts as a regulatory mechanism enforcing control loops between human and machine agents. The operations here are not merely transaction-level interactions, but control-oriented interventions across dynamically defined roles and states. This systemic framing introduces a new axis of operation classification—governance and adaptation—which complements traditional task-level distinctions.

Finally, the work in [17] presents a visual, reuse-oriented code generator for smart contracts, based on BPMN models. Their generator employs a multi-layer library that separates generic logic, domain-specific functions, and composable workflow templates. Each layer corresponds to different granularity and specialization of contract logic, enabling domain experts to reuse tested modules without extensive programming knowledge. Code generation, compilation, and deployment are integrated in a single toolchain, streamlining smart contract creation for multi-party interactions. This approach highlights the value of classifying blockchain operations by reusability and modularity, with particular emphasis on how different layers facilitate efficient development and clearer separation of concerns.

Together, these studies affirm the importance of categorizing blockchain-enabled operations not only by their functional scope, but also by their positioning across trust, orchestration, and abstraction layers. This classification is important for enabling scalable and maintainable process execution in decentralized environments.

3. Methodology

Our research methodology combines BPMN process modeling with blockchain implementation on both Layer 1 and Layer 2 networks, focusing on secure and efficient timer event handling. This section details our approach to process modeling, smart contract generation, security verification, and cost analysis.

3.1. BPMN Process Modeling

We created three representative BPMN collaboration diagrams using Camunda Modeler, each featuring different timer event patterns and multi-participant interactions. The first case study is a citizen-municipality interaction for a document request process with 30-day and 15-day processing deadlines. The second one is a policyholder-insurer application workflow with 5-day and 7-day decision deadlines. The third application is a three-party (professor-student-secretary) workflow with 15-day evaluation and 7-day decision deadlines. Each process was designed to capture common time-constrained business scenarios while providing sufficient complexity to validate our approach across various timer patterns and participant configurations.

3.2. Smart Contract Generation

We developed an automated BPMN-to-Solidity transformation tool that generates blockchain-executable smart contracts from BPMN diagrams. Our transformation approach employs a block-based timer implementation. Rather than using timestamp-based mechanisms, our approach converts ISO duration formats (e.g., "P30D" for 30 days) into block numbers. According to Ethereum network statistics [1], since October 2022, the daily average block time has maintained a consistent 12-second interval, providing a stable foundation for our time calculations. This stability allows us to reliably estimate that one day corresponds to approximately 7200 blocks (24 hours * 3600 seconds / 12 seconds per block). This conversion uses the formula:

$$\text{blockEstimate} = \text{days} * 7200 \frac{\text{blocks}}{\text{day}}$$

Arbitrum does not maintain a fixed block time like Ethereum. However, according to Offchain Labs, the average latency of Arbitrum One blocks is around 250 milliseconds [2] [3].

This approach provides resistance to miner manipulation of timestamps, a deterministic execution without relying on external time oracles, and a consistent timing across network conditions.

The generated contracts implement a state machine that tracks each BPMN element's lifecycle through three states: DISABLED, ENABLED, and DONE. Element dependencies are enforced through state checks, ensuring process integrity.

Moreover, each function includes participant validation based on Ethereum addresses mapped to BPMN participants, ensuring that only authorized participants can execute their designated tasks.

Finally the generated contracts incorporate *OpenZeppelin*'s security patterns, including *Reentrancy-Guard*, *Ownable*, and *Pausable*, providing protection against common attack vectors and administrative control mechanisms.

[1] https://etherscan.io/chart/blocktime
[2] https://research.arbitrum.io/t/the-power-of-faster-blocks/9609
[3] https://arbiscan.io/blocks

3.3. Security Analysis

We implemented a two-stage security verification process. First, each generated contract underwent automated vulnerability detection using *Slither* [18], a static analysis framework that identifies common smart contract vulnerabilities including reentrancy, integer overflow/underflow, and access control issues. Then we supplemented the Slither analysis with custom checks specifically designed for timer-based operations, including verification of proper access controls for timer functions, analysis of block number manipulation risks, assessment of dependency enforcement mechanisms and validation of participant address handling.

These security measures ensured that the generated contracts maintained process integrity while addressing blockchain-specific vulnerabilities. All contracts passed both the Slither analysis and our custom security checks with zero critical vulnerabilities detected.

3.4. Deployment and Cost Analysis

We deployed each smart contract on both *Ethereum Sepolia Testnet* (Layer 1) and *Arbitrum Sepolia Testnet* (Layer 2) using Remix IDE and MetaMask. For each contract, we executed the complete process flow, recording gas consumption for every operation. To provide a realistic cost assessment, we collected daily gas price data from Etherscan[4] and Arbiscan[5], along with cryptocurrency price data from CoinGecko[6][7], over a one-month period (March 14 - April 13, 2025). We then calculated median gas price and the median cryptocurrency price and the standard deviation. We found for Ethereum 0.6350 ± 1.1438 Gwei and $\$1{,}887.76 \pm 167.64$, instead for Arbitrum 0.0174 ± 0.1042 Gwei and $\$0.33397 \pm 0.03893$.

This methodology provides a robust cost estimation that accounts for market fluctuations while avoiding outlier-driven distortions. By using median values rather than means, we established a realistic operational cost baseline for blockchain-based process execution across different layers.

4. Implementation and Results

This section presents our implementation of three case studies and the comparative cost analysis between Layer 1 and Layer 2 blockchain networks. The BPMN codes of the case studies, the Java code of the automatic translator and the translated three smart contracts are available at **this link**.

4.1. Case Study Implementation

We implemented three distinct BPMN processes, each selected to represent different timer event patterns and multi-participant scenarios common in real-world applications.

The first case study in Figure 1 implements a municipal document request process between a citizen and municipality. This process includes a 30-day primary processing window and a 15-day extension period when additional processing time is needed. The workflow begins with a completeness check that creates an initial decision point, followed by document processing with enforced regulatory deadlines. This case study demonstrates how timer events ensure compliance with administrative procedures and service level agreements.

The second case study in Figure 2 models an insurance policy application process between a policyholder and an insurance company. This workflow includes two timer events: a 7-day proposal deadline for the insurance company to prepare a policy proposal and a 5-day decision deadline for the policyholder to sign the policy. The process includes decision points that create alternative paths depending on the completeness of the application and the policyholder's acceptance decision. This case study demonstrates timer events that govern both organizational and customer-facing deadlines.

[4] https://etherscan.io/gasTracker#chart_gasprice
[5] https://arbiscan.io/chart/gasprice
[6] https://www.coingecko.com/it/monete/ethereum/historical_data?start=2025-03-13&end=2025-04-12
[7] https://www.coingecko.com/en/coins/arbitrum/historical_data?start=2025-03-12&end=2025-04-14

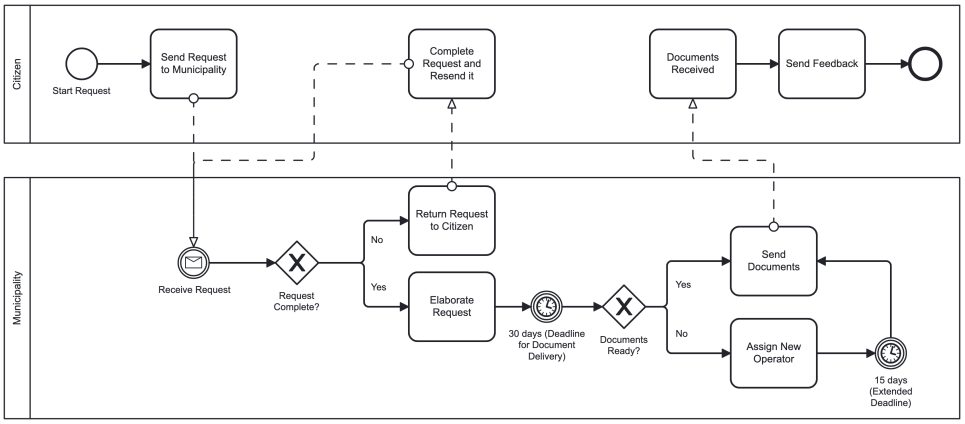

Figure 1: BPMN Diagram Citizen and Municipality case study

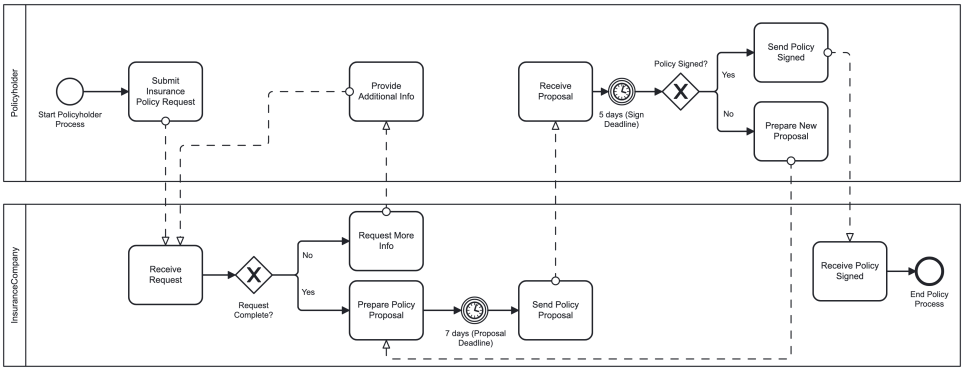

Figure 2: BPMN Diagram Policyholder and Insurance Company case study

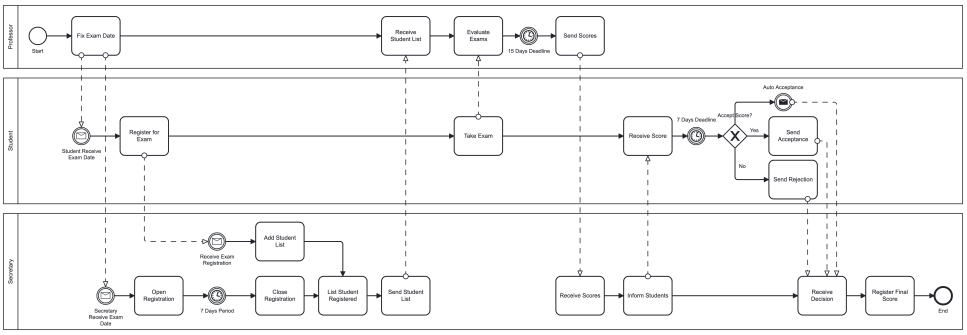

Figure 3: BPMN Diagram Student, Professor and Secretary case study

Finally, the third case study in Figure 3 represents an academic examination process involving three participants: professor, student, and secretary. This complex workflow features multiple timer events, including a 7-day registration period, a 15-day deadline for professors to evaluate exams, and a 7-day deadline for students to accept or reject their grades. The process also includes an auto-acceptance mechanism if no decision is made within the specified timeframe. This case study demonstrates how timer events can coordinate activities among multiple participants with interdependent responsibilities.

Each BPMN model was transformed into a Solidity smart contract using our automated conversion tool. The transformation process preserved the timer event semantics by implementing them as block-number-based mechanisms rather than traditional timestamp approaches. The generated smart contracts included security features from OpenZeppelin, participant role management, and comprehensive audit logging to track process execution.

4.2. Deployment and Gas Analysis

We deployed each smart contract on both Ethereum Sepolia Testnet (Layer 1) and Arbitrum Sepolia Testnet (Layer 2) to analyze performance and cost differences.

#	Function/Action	Gas Used	ETH Cost (USD)	ARB Cost (USD)
1	*Contract Creation*	4,918,636	$5.90	$0.02860
2	`startEvent()`	147,921	$0.18	$0.00086
3	`sendRequestToMunicipality()`	155,448	$0.19	$0.00090
4	`receiveRequest()`	130,843	$0.16	$0.00076
5	`gatewayAction("CheckRequest")`	149,728	$0.18	$0.00087
6	`elaborateRequest()`	133,311	$0.16	$0.00078
7	`f30DaysDeadlineForDocumentDelivery()`	133,288	$0.16	$0.00078
8	`gatewayAction("CheckDocuments")`	149,579	$0.18	$0.00087
9	`sendDocuments()`	135,784	$0.16	$0.00079
10	`documentsReceived()`	130,800	$0.16	$0.00076
11	`sendFeedback()`	133,291	$0.16	$0.00078

Table 1
Approx. USD cost per transaction on Ethereum vs Arbitrum (14 Mar–13 Apr 2025). **Ethereum:** median gas price 0.635 Gwei, ETH $1,887.76. **Arbitrum:** median gas price 0.0174 Gwei, ARB $0.33397.

Table 1 presents the gas consumption and approximate USD cost for each operation in the municipal document request process. The contract deployment required approximately 4.9 million gas units, translating to $5.90 on Ethereum and just $0.0286 on Arbitrum. Individual operations like submitting a request consumed between 130,000-185,000 gas units, corresponding to $0.16-$0.19 on Ethereum but only $0.00076-$0.00090 on Arbitrum. Notably, timer event functions that implement the 30-day and 15-day deadlines consumed similar gas amounts to standard operations (approximately 133,000 gas units).

#	Function/Action	Gas Used	ETH Cost (USD)	ARB Cost (USD)
1	*Contract Creation*	5,197,620	$6.24	$0.03020
2	`startEvent()`	147,965	$0.18	$0.00086
3	`submitInsurancePolicyRequest()`	133,289	$0.16	$0.00078
4	`receiveRequest()`	130,843	$0.16	$0.00076
5	`gatewayAction("Gateway_CheckRequest", true)`	149,646	$0.18	$0.00087
6	`preparePolicyProposal()`	133,291	$0.16	$0.00078
7	`f7DaysProposalDeadline()`	133,334	$0.16	$0.00078
8	`sendPolicyProposal()`	133,288	$0.16	$0.00078
9	`receiveProposal()`	130,822	$0.16	$0.00076
10	`f5DaysSignDeadline()`	133,290	$0.16	$0.00078
11	`gatewayAction("Gateway_CheckSign", true)`	149,670	$0.18	$0.00087
12	`sendPolicySigned()`	133,289	$0.16	$0.00078
13	`receivePolicySigned()`	130,821	$0.16	$0.00076

Table 2
Approx. USD cost per transaction on Ethereum vs Arbitrum (14 Mar–13 Apr 2025). **Ethereum:** median gas price 0.635 Gwei, ETH $1,887.76. **Arbitrum:** median gas price 0.0174 Gwei, ARB $0.33397.

Similar patterns emerged in the insurance policy application process (Table 2), where contract deployment required 5.2 million gas units ($6.24 on Ethereum, $0.0302 on Arbitrum). The timer functions for the 7-day proposal deadline and 5-day sign deadline consumed approximately 133,300 gas units each, equivalent to $0.16 on Ethereum and $0.00078 on Arbitrum.

The academic examination process, with its increased complexity and participant count, showed comparable per-operation gas consumption (Table 3) but higher cumulative costs due to the greater

number of process steps. Contract deployment required 5.0 million gas units ($6.04 on Ethereum, $0.029 on Arbitrum), while individual operations ranged from 111,000 to 176,000 gas units.

#	Function/Action	Gas Used	ETH Cost (USD)	ARB Cost (USD)
1	*Contract Creation*	5,030,195	$6.04	$0.02900
2	`startEvent()`	146,589	$0.18	$0.00085
3	`fixExamDate()`	176,404	$0.21	$0.00103
4	`secretaryReceiveExamDate()`	129,595	$0.16	$0.00075
5	`studentReceiveExamDate()`	129,638	$0.16	$0.00075
6	`openRegistration()`	131,849	$0.16	$0.00077
7	`f7DaysPeriod()`	131,904	$0.16	$0.00077
8	`closeRegistration()`	112,119	$0.13	$0.00065
9	`listStudentRegistered()`	131,922	$0.16	$0.00077
10	`sendStudentList()`	111,927	$0.13	$0.00065
11	`receiveStudentList()`	131,880	$0.16	$0.00077
12	`registerForExam()`	154,087	$0.18	$0.00090
13	`receiveExamRegistration()`	129,604	$0.16	$0.00075
14	`addStudentList()`	131,971	$0.16	$0.00077
15	`takeExam()`	134,204	$0.16	$0.00078
16	`evaluateExams()`	131,825	$0.16	$0.00077
17	`f15DaysDeadline()`	131,843	$0.16	$0.00077
18	`sendScores()`	131,776	$0.16	$0.00077
19	`receiveScores()`	129,578	$0.16	$0.00075
20	`informStudents()`	134,206	$0.16	$0.00078
21	`receiveScore()`	131,796	$0.16	$0.00077
22	`f7DaysDeadline()`	131,753	$0.16	$0.00077
23	`gatewayAction("Gateway_ScoreDecision")`	147,271	$0.18	$0.00086
24	`sendAcceptance()`	111,919	$0.13	$0.00065
25	`receiveDecision()`	131,769	$0.16	$0.00077
26	`registerFinalScore()`	131,834	$0.16	$0.00077

Table 3
Approx. USD cost per function on Ethereum vs Arbitrum (**Ethereum:** median gas price 0.635 Gwei, ETH=$1,887.76; **Arbitrum:** median gas price 0.0174 Gwei, ARB=$0.33397).

4.3. Layer 1 vs Layer 2 Cost Comparison

Our analysis reveals that Arbitrum's Layer 2 implementation reduces operational costs compared to Ethereum's Layer 1. This dramatic cost reduction maintains the same functional capabilities and security properties while providing significantly better economic feasibility for complex business processes. The cost difference is primarily attributable to two factors: lower gas prices on Arbitrum (median 0.0174 Gwei vs. 0.635 Gwei on Ethereum) and the lower native token value ($0.33397 for ARB vs. $1,887.76 for ETH). While both factors contribute to cost reduction, the gas price difference represents the architectural efficiency of Layer 2 solutions, while token value differences reflect market conditions. The median operation cost on Ethereum ($0.16-$0.18) would become prohibitive for complex processes with many participants and operations, potentially costing hundreds of dollars for complete process execution. In contrast, the same operations on Arbitrum cost fractions of a cent ($0.00076-$0.00090), making even complex processes economically viable.

4.4. Timer Event Performance

Our block-number-based timer implementation demonstrated consistent and predictable behavior across both networks. By converting time durations to block numbers (7,200 blocks per day, assuming 12-second block times), the system created a reliable execution framework that avoided timestamp manipulation vulnerabilities. The gas consumption for timer-related functions remained consistent

with other operations, indicating no significant performance penalty for implementing time constraints. This consistency suggests that complex time-dependent business processes can be implemented without disproportionate cost increases for timer management. Security analysis confirmed that our block-based approach successfully prevented common time-based vulnerabilities. Both automated Slither analysis and custom security checks verified the integrity of the timer implementation, with no critical vulnerabilities detected.

5. Proposed Hybrid Layer 1-Layer 2 Architecture

Our comparative analysis of Layer 1 and Layer 2 execution costs reveals a compelling economic case for implementing BPMN processes on Layer 2 networks. However, certain operations may benefit from the stronger security guarantees of Layer 1. This section introduces a systematic framework for distributing process operations between Layer 1 and Layer 2 based on security requirements, economic considerations, and operational characteristics.

5.1. Operation Classification Framework

We propose a classification framework that categorizes BPMN operations based on three dimensions: *security criticality*, *execution frequency*, and *data persistence* requirements. This framework provides a structured approach to determining the most suitable execution layer for each operation.
Security Criticality Assessment: Each operation is evaluated on a security criticality scale based on Table 4.

Financial impact	Operations that directly influence financial transactions or asset transfers require the highest security guarantees
Legal significance	Operations that create legally binding commitments or satisfy regulatory requirements demand strong immutability guarantees
Trust requirements	Operations involving multiple participants with limited trust relationships benefit from Layer 1's stronger consensus mechanisms

Table 4
Operations Security Criticality Assessment Framework

Execution Frequency Analysis: Operations are classified by their expected execution frequency according with Table 5.

Initialization operations	One-time setup operations that establish the process framework
Decision points	Gateway operations that determine process flow direction
Routine operations	Regular tasks performed throughout the process lifecycle
Timeout operations	Timer-triggered operations that occur after specific periods

Table 5
Operations Execution Frequency Analysis Framework

Data Persistence Requirements: Operations are categorized based on their data persistence needs following Table 6.

Critical state transitions	Major state changes that fundamentally alter process status
Intermediate state updates	Incremental changes that track process progression
Audit information	Supportive data primarily used for verification and transparency

Table 6
Operations Data Persistence Requirements Framework

5.2. Layer Assignment Algorithm

Based on the classification framework, Layer 1 primary candidates are process initialization operations that establish core parameters, operations with high financial impact (<$1000 value transfer), final commitment operations with legal significance and major state transitions that fundamentally alter process status. Layer 2 primary candidates are routine operations with moderate security requirements, frequent status update operations, operations with minimal financial impact and intermediate data collection activities. Borderline cases can be timer events that enforce critical deadlines but have no direct financial impact, decision gateways that influence but don't directly control asset transfers and operations with moderate legal significance. For borderline cases, we recommend additional assessment of the relative cost differential between Layer 1 and Layer 2 implementation, the specific security guarantees required by stakeholders and the potential impact of a security compromise.

5.3. Implementation Architecture

Our conceptual architecture proposes a multi-contract approach with Layer 1 serving as the authoritative system of record while Layer 2 handles routine operations.

The Layer 1 contract serves as the process anchor, maintaining core process parameters and participant information, critical state variables that define process status, authorization controls for cross-layer synchronization and final state commitments and financial settlements.

The Layer 2 contract handles daily operations, including routine task execution and status tracking, intermediate state updates, timer event monitoring, gateway decision processing and audit trail maintenance.

To maintain process integrity across layers, our framework would require a secure synchronization mechanism. Even if we do not implement this component in the current work, several established approaches from literature could be adapted for this purpose. The events-based architecture described by Bigiotti et al. [19] provides a promising foundation, where watchtowers monitor events from interface smart contracts to coordinate state changes across chains. Their protocol demonstrates how two transactions (one on each chain) can successfully complete cross-chain operations using standardized JSON message formats for event semantics.

Other potential approaches include Merkle-root state commitments for periodic validation [20], message passing protocols like LayerZero [21], and cryptographic proof validation [22] systems similar to those used in optimistic rollups.

The practical implementation of such synchronization mechanisms represents an important direction for future work, as our current focus is on the operation classification framework and on the economic analysis of hybrid approaches.

5.4. Case Study Application

Applying our framework to the municipal document request process in Figure 1.

Layer 1 Operations	Layer 2 Operations
Process initialization (contract deployment)	Request submission and reception
Final document delivery confirmation	Request elaboration tasks
30-15 days deadline start and end	Intermediate status updates
Gateway decisions (request completeness verification)	Timer event triggers and tracking
	New operator assignment
	Feedback collection

Table 7
Layer 1 vs. Layer 2 operations for the citizen and municipality case study.

This distribution in Table 7 reduces the total process execution cost compared to a pure Layer 1 implementation while maintaining appropriate security guarantees for critical operations. Similar

patterns emerge in the insurance application and academic examination processes, with operations involving financial commitments, legal obligations, or final state determinations assigned to Layer 1, while routine coordination activities migrate to Layer 2.

5.5. Security and Cost Analysis

The proposed solution can balance security and cost considerations through strategic operation placement. By executing only the most critical operations on Layer 1, we maintain strong security guarantees where needed while leveraging Layer 2's cost efficiency for routine operations. For a typical process execution in our municipal document request case study, the hybrid approach reduces costs of a pure Layer 1 implementation while maintaining Layer 1 security for critical operations. This represents a balanced trade-off between the complete security of Layer 1 and the full economy of Layer 2. The architecture's security properties derive from Ethereum's underlying consensus mechanism for critical operations, while accepting Arbitrum's security model for non-critical activities. This tiered approach ensures that operations receive security guarantees proportional to their criticality.

6. Assumptions and Limitations

6.1. Explicit Assumptions

Implementing the three case studies, we made the following assumptions:

Block-time model – Ethereum average of 12 s; Arbitrum latency 250 ms.
Consensus security – honest-majority assumption for both layers.
Gas-price window – median prices 14 Mar–13 Apr 2025 (see 3.4).

We acknowledge these assumptions, which were selected empirically at the time of the study and can be adjusted in future replications.

6.2. Limitations and Threats to Validity

While our approach demonstrates promising results in addressing time-constrained execution in blockchain-based workflows, several limitations remain that warrant further attention. However, some important considerations must be acknowledged when interpreting our findings. Our case studies, while diverse, may not represent the full spectrum of business processes and timer event patterns. Additional validation across a broader range of process types and industry domains would strengthen the generalizability of our findings. The block-number-based time conversion assumes a relatively stable block time (12 seconds for Ethereum); however, actual block times can vary due to network conditions, potentially affecting the precision of timer events. Long-term monitoring of timer accuracy across network conditions would provide additional validation. Our cost analysis used median gas prices over a one-month period to provide a realistic baseline. However, gas prices can be highly volatile, particularly during network congestion periods. Cost projections should account for this potential variability. While our synchronization mechanism was conceptually described, a complete implementation and validation of cross-layer communication remains to be developed. The practical reliability of maintaining consistent state across layers requires further empirical validation. The security guarantees of our hybrid architecture rely on the underlying security models of both Ethereum and Arbitrum. Changes to these platforms or their security properties could impact the overall security of the implementation.

7. Conclusion and Future Work

This paper presented Block-Timer, an end-to-end evaluation of block-number timers for implementing BPMN process execution with time constraints on blockchain. Our implementation of block-number-based for timer events ensures secure, manipulation-resistant time tracking without relying on external

time services, and the hybrid proposed solution can strategically distributes process operations between Layer 1 and Layer 2 blockchains based on security requirements and cost considerations. Our empirical analysis across three case studies demonstrated that Arbitrum's Layer 2 implementation reduces operational costs compared to Ethereum's Layer 1, while maintaining the same functional capabilities. The gas consumption for timer event operations remained comparable to other process operations (approximately 133,000 gas units), indicating no significant performance penalty for implementing time constraints. Although the hybrid architecture is not yet fully implemented, our preliminary evaluation suggests that it strikes a favorable balance between security and cost-efficiency by executing only the most critical operations (process initialization, major state transitions, and legally significant commitments) on Layer 1, while delegating routine operations (status updates, data collection, and timer tracking) to Layer 2. This approach ensures appropriate security guarantees for operations based on their criticality while significantly reducing overall execution costs. Our security analysis confirmed the integrity of the block-number-based timer implementation, with no critical vulnerabilities detected through automated and custom security checks. By converting time durations to block numbers (7,200 blocks per day), our approach provides resilience against timestamp manipulation attacks while creating a deterministic execution framework.

Several promising directions for future research emerge from this work. Implementing and evaluating a synchronization mechanisms between Layer 1 and Layer 2 deserves particular attention, with emphasis on optimizing the balance between security guarantees and operational costs. Extending the block-number-based approach to support more complex timer patterns represents another important direction. This includes recurring timers, calendrical expressions (e.g., "every Monday"), and business day calculations that account for holidays and working hours. By addressing these challenges, future research can further enhance the efficiency, security, and practicality of blockchain-based business process execution, particularly for time-sensitive multi-party workflows where trust and accountability are essential.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union-NextGenerationEU.

This work was partially funded by Ministero dell'Università e della Ricerca (MUR), issue D.M. 351/2022 "Borse di Dottorato"—Dottorato di Ricerca di Interesse Nazionale in "Blockchain e Distributed Ledger Technology", under the National Recovery and Resilience Plan (NRRP).

We acknowledge financial support under the project HALO (Hazard-Analysis and Anti-Counterfeiting Ledger Oriented Protection) CUP F23C23000310008 in the context of Development Regional Program 2020-2024 Strategy 2-Economics Identity Project 2.1 - Technological Research and Innovation.

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5-Call for tender No. 3277 published on 30 December 2021 by the Italian Ministry of University and Research (MUR) funded by the European Union-NextGenerationEU. Project Code ECS0000038—Project Title eINS Ecosystem of Innovation for Next Generation Sardinia—CUP F53C22000430001-Grant Assignment Decree No. 1056 adopted on 23 June 2022 by the Italian Ministry of University and Research (MUR).

We acknowledge financial support under the project code PE0000021, Concession Decree No. 1561 of 11.10.2022 adopted by Ministero dell'Università e della Ricerca (MUR), CUP - to be indicated by each Beneficiary, according to attachment E of Decree No. 1561/2022, Project title "Network 4 Energy Sustainable Transition – NEST".

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), by the Italian Ministry of University and Research (MUR) funded by the European Union-NextGenerationEU. Project Code ECS0000038—Project eINS Ecosystem of Innovation for Next Generation Sardinia—CUP F53C22000430001-Grant Assignment for the PoC "TraCCCS: Tracciabilità, Certificazione Blockchain e Valorizzazione dei Carbon Credits per PMI Sarde con Impianti di Produzione di Energia Rinnovabile".

Declaration on Generative AI: During the preparation of this work, the author(s) used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] T. Allweyer, BPMN 2.0: introduction to the standard for business process modeling, BoD–Books on Demand, 2016.

[2] W. Viriyasitavat, L. Da Xu, D. Niyato, Z. Bi, D. Hoonsopon, Applications of blockchain in business processes: A comprehensive review, Ieee Access 10 (2022) 118900–118925.

[3] 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. doi:10.1109/EDOC49727.2020.00034.

[4] T. Neudecker, H. Hartenstein, Network layer aspects of permissionless blockchains, IEEE Communications Surveys & Tutorials 21 (2018) 838–857.

[5] A. Gangwal, H. R. Gangavalli, A. Thirupathi, A survey of layer-two blockchain protocols, Journal of Network and Computer Applications 209 (2023) 103539.

[6] C. Di Ciccio, A. Cecconi, M. Dumas, L. García-Bañuelos, O. López-Pintado, Q. Lu, J. Mendling, A. Ponomarev, A. Binh Tran, I. Weber, Blockchain support for collaborative business processes, Informatik Spektrum 42 (2019) 182–190.

[7] N. Nousias, G. Tsakalidis, S. Petridou, K. Vergidis, Modelling the development and deployment of decentralized applications in ethereum blockchain: a bpmn-based approach, in: International Conference on Decision Support System Technology, Springer, 2022, pp. 55–67.

[8] A. B. Tran, Q. Lu, I. Weber, Lorikeet: A model-driven engineering tool for blockchain-based business process execution and asset management., in: BPM (dissertation/demos/industry), 2018, pp. 56–60.

[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] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, F. Tiezzi, Blockchain-based execution of bpmn choreographies with multiple instances, Distributed Ledger Technologies: Research and Practice (2023).

[11] F. Corradini, A. Marcelletti, A. Morichetta, A. Polini, B. Re, F. Tiezzi, A flexible approach to multi-party business process execution on blockchain, Future Generation Computer Systems 147 (2023) 219–234.

[12] T. Eichinger, M. Ebermann, Can we effectively use smart contracts to stipulate time constraints?, in: 2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), IEEE, 2022, pp. 11–18.

[13] S.-W. Chae, J.-I. Kim, Y. Park, Practical time-release blockchain, Electronics 9 (2020). URL: https://www.mdpi.com/2079-9292/9/4/672. doi:10.3390/electronics9040672.

[14] L. Argento, F. Buccafurri, A. Furfaro, S. Graziano, A. Guzzo, G. Lax, F. Pasqua, D. Saccà, Id-service: A blockchain-based platform to support digital-identity-aware service accountability, Applied Sciences 11 (2021). URL: https://www.mdpi.com/2076-3417/11/1/165. doi:10.3390/app11010165.

[15] M. Alfarisy, Suhardi, W. Muhamad, Development of a service-oriented platform and blockchain as a service to secure recording of measured fishing data based on quota in support of the blue economy, in: 2024 International Conference on ICT for Smart Society (ICISS), 2024, pp. 1–6. doi:10.1109/ICISS62896.2024.10751451.

[16] M. Pirani, A. Cucchiarelli, T. Naeem, L. Spalazzi, A blockchain-driven cyber-systemic approach to hybrid reality, Systems 13 (2025). URL: https://www.mdpi.com/2079-8954/13/4/294. doi:10.3390/systems13040294.

[17] X. Shen, W. Li, H. Xu, X. Wang, Z. Wang, A reuse-oriented visual smart contract code generator

for efficient development of complex multi-party interaction scenarios, Applied Sciences 13 (2023). URL: https://www.mdpi.com/2076-3417/13/14/8094. doi:10.3390/app13148094.

[18] J. Feist, G. Grieco, A. Groce, Slither: A static analysis framework for smart contracts, in: 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), 2019, pp. 8–15. doi:10.1109/WETSEB.2019.00008.

[19] A. Bigiotti, L. Mostarda, A. Navarra, A. Pinna, R. Tonelli, M. Vaccargiu, Interoperability between evm-based blockchains, in: L. Barolli (Ed.), Advanced Information Networking and Applications, Springer Nature Switzerland, Cham, 2024, pp. 98–109.

[20] M. Westerkamp, J. Eberhardt, zkrelay: Facilitating sidechains using zksnark-based chain-relays, in: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroSPW), 2020, pp. 378–386. doi:10.1109/EuroSPW51379.2020.00058.

[21] R. Zarick, B. Pellegrino, C. Banister, Layerzero: Trustless omnichain interoperability protocol, 2021. URL: https://arxiv.org/abs/2110.13871. arXiv:2110.13871.

[22] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, D. Song, zkbridge: Trustless cross-chain bridges made practical, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 3003–3017. URL: https://doi.org/10.1145/3548606.3560652. doi:10.1145/3548606.3560652.