COMPUTER DESIGN SYSTEMS. THEORY AND PRACTICE

Vol. 7, No. 2, 2025

Oleh Kernytskyy[1], Andriy Kernytskyy[2]

[1] Department of Automated Control Systems, Lviv Polytechnic National University, 12, S. Bandery str., Lviv, Ukraine, E-mail: oleh.b.kernytskyy@lpnu.ua, ORCID 0009-0007-5318-6506

[2] Computer Aided Design Department, Lviv Polytechnic National University, 12, S. Bandery str., Lviv, Ukraine,E- mail: andriy.b.kernytskyy@lpnu.ua, ORCID 0000-0001-8188-559X

A PETRI NET-BASED APPROACH TO EXTRACTING AND VALIDATING REQUIREMENTS FOR MODELLING AND MODERNIZING ARCHAIC IT SYSTEMS

Received:August 12, 2025 / Revised: August 25, 2025 / Accepted: September 15, 2025

https://doi.org/10.23939/cds2025.02.132

Abstract. Archaic IT systems are critical to the functioning of many organizations but present significant challenges for modernization due to their inherent complexity, outdated technologies, and lack of adequate documentation. Extracting and understanding the workflows and embedded requirements within such systems is essential for adapting them to contemporary infrastructure while maintaining regulatory compliance and operational integrity. This article explores the use of Petri Nets as a formal and graphical modelling tool to address these challenges. Petri Nets offer a powerful framework for representing system workflows, capturing concurrency, synchronization, and decision-making processes. The methodology begins with a structured approach to requirements retrieval from legacy systems through reverse engineering, stakeholder engagement, and data analysis. Using the credit report creation process as a case study, we illustrate how Petri Nets can effectively model intricate workflows, including data preprocessing, authorization checks, backend service integration, and fallback mechanisms.

The results demonstrate their potential for formal analysis, communication between technical and non-technical stakeholders, and incremental modernization of legacy systems. This article highlights how Petri Nets empower organizations to reconstruct undocumented workflows with precision, enabling sustainable modernization while preserving essential functionalities. By providing a detailed methodology and analysis of Petri Nets, this study offers a structured framework for tackling legacy system challenges and advancing innovative modernization strategies.

Keywords: Petri Nets, legacy IT systems, workflow modelling, requirements extraction, formal validation, system modernization, fallback mechanisms, concurrency analysis, dynamic simulation.

Introduction

Legacy IT systems remain critical for many organizations, supporting essential business processes, governmental workflows, and regulatory compliance. Despite their importance, these systems are often outdated, undocumented, and rigid, making them difficult to adapt to modern demands for scalability, interoperability, and regulatory compliance [1, 2]. Modernization of these systems requires transitioning them to contemporary platforms while preserving critical functionalities but extracting and validating requirements from undocumented workflows poses significant challenges. These requirements often include workflows, compliance mechanisms, and fallback processes that are deeply embedded within outdated software and must adhere to complex regulations, such as OFAC and MLA compliance [3, 4].

To address these challenges, this study explores a Petri Net-based approach for systematically reconstructing, validating, and formalizing workflows in legacy IT systems. Petri Nets, particularly Coloured Petri Nets (CPNs), provide a structured and scalable framework to model distributed, concurrent, and

asynchronous processes. They have been employed in academic research to abstract execution traces, convert informal descriptions into formal models, and detect missing or inconsistent behaviours. By bridging gaps caused by incomplete documentation, Petri Nets offer a reliable foundation for understanding and validating system requirements.

This research contributes by presenting a methodology that leverages Petri Nets to accurately represent workflows and dependencies, enabling organizations to formalize requirements and ensure correctness. A case study on credit report creation workflows demonstrates how the approach captures and models complex processes essential to modernizing legacy systems. Unlike existing methods like process mining or manual analysis, the proposed approach reconstructs workflows comprehensively, mitigates risks, and prepares systems for future scalability and adaptability.

Objectives and Problems of Research

The primary objective of this research is to propose and validate a Petri Net-based approach for extracting and validating requirements from legacy IT systems. This methodology aims to address challenges such as undocumented workflows, tightly coupled dependencies, and outdated structures in legacy systems. Specifically, the study seeks to develop a systematic framework for identifying and capturing workflows and operational rules from undocumented systems, represent these workflows using Petri Nets for formal clarity, and demonstrate the methodology's practical application through a case study on credit report generation workflows. Furthermore, the study highlights the benefits of Petri Nets over other techniques, emphasizing their strengths in formal analysis, concurrency handling, and iterative flexibility.

Legacy IT systems face numerous challenges during modernization [5–11]. Many lack documentation, making it difficult to understand workflows and dependencies, while tightly coupled structures and outdated programming languages complicate updates and replication. Workflows are often deeply embedded in multiple code layers, requiring significant reverse engineering efforts to extract and validate them. Additionally, legacy systems frequently encode critical compliance mechanisms that must be preserved during modernization to avoid regulatory risks. Existing methods like manual analysis or process mining struggle to handle undocumented processes, concurrency, fallback paths, and alternate workflows. The absence of formal frameworks for validation further increases the risk of incomplete or inefficient models.

This research addresses these issues by leveraging Petri Nets' formalism and rigor to represent, analyze, and validate requirements. Petri Nets provide clear workflow representation, analytical tools to ensure accuracy and reliability, and visual clarity that fosters collaboration between stakeholders. By addressing key challenges, the proposed methodology aims to offer organizations a robust framework to modernize legacy systems while minimizing risks and preserving essential functionalities.

Review of Modern Information Sources on the Subject of the Paper

The modernization of legacy IT systems has been a significant area of research and development in both academic and industrial domains due to the critical reliance on such systems for business operations, government services, and regulatory compliance. Legacy systems, commonly characterized by their outdated architectures and lack of sufficient documentation, often serve as bottlenecks in organizations' efforts to adapt to evolving technological and business needs. The field has explored a variety of approaches for requirements extraction, workflow modelling, and system validation to address these challenges. This review highlights the key contributions from modern information sources relevant to the themes of this article: legacy system modernization, requirements extraction, and the use of Petri Nets as a modelling and validation framework.

a. Requirements Extraction

Requirements extraction from undocumented legacy systems is a core challenge. Modern research discusses techniques like stakeholder interviews, log analysis, and manual audits to extract workflows and business rules [12]. However, these methods are limited in handling complex interactions, hidden logic, and dependencies within legacy systems. To address validation challenges:

- Mathematical frameworks have been applied to validate extracted workflows, particularly for ensuring correctness and consistency [13]. Despite their effectiveness, their adoption remains limited due to

a lack of integration with practical modelling tools.

- Iterative prototyping has been used to validate workflows extracted from legacy systems by refining processes based on early user feedback [14]. This method struggles to handle concurrency and synchronization issues in workflows.

The inability of current techniques to comprehensively address validation, particularly in multi-threaded, resource-constrained, or fallback-driven workflows, highlights the need for a formalized, analytical approach like Petri Nets.

b. Petri Nets for Workflow Modelling and Validation

Petri Nets have been extensively studied and applied in domains requiring formal representation, simulation, and validation of workflows, particularly in complex systems. Introduced by Carl Adam Petri in the 1960s, Petri Nets combine graphical representation with mathematical rigor, providing a dual role in modelling and ensuring correctness. They have been widely utilized in various fields:

- Petri Nets are used in software engineering to model and analyse concurrent and distributed systems, ensuring properties like liveness, safety, and reachability [15].
- Petri Nets are used in Business Process Management to analyse workflows for bottlenecks and evaluate process conformance [16].
- Petri Nets facilitate interaction modelling between software and physical systems, particularly in control systems and IoT [17–19].
- Petri Nets are used in Healthcare Systems to optimize patient workflows and healthcare resource allocation [20, 21].

Central to Petri Nets' success is their ability to represent complex dependencies, concurrent operations, and fallback mechanisms that are crucial in legacy system workflows. Properties like reachability ensure that all necessary states and actions are possible, liveness guarantees no parts of the system stagnate, and boundedness prevents unexpected overflows or resource contention [22]. These properties are essential to ensure the correctness and scalability of workflows during modernization.

Several tools, such as CPN Tools, WoPeD, and PIPE2, have been developed to support Petri Net modelling and validation. These tools not only allow the graphical design of workflows but also enable formal analysis of their properties, making them particularly valuable for legacy system modernization. Despite their strengths, the use of Petri Nets is underexplored in the context of extracting and validating workflows specific to legacy IT systems, making this article a meaningful contribution to the field.

c. Comparison with Other Methods

Petri nets differ from other modelling and extraction methods primarily in their formal semantics and support for concurrency. Unlike UML or BPMN diagrams (which are informal and often require translation for analysis), Petri nets have precise mathematical definitions: places, transitions, and tokens all have unambiguous behaviour. This formality allows automated model checking and performance analysis that UML / BPMN lack by default. Similarly, traditional finite-state machines or state charts typically model only sequential flows, whereas Petri nets natively express parallel execution and synchronization. In fact, many process mining approaches explicitly use Petri nets as the discovery language: algorithms map event logs to Petri nets (e. g. the α-algorithm, ILP, Heuristic miner) and evaluate the resulting net against quality metrics. In comparison, purely data-driven or NLP-based extraction tools may surface requirement candidates, but they do not inherently capture control flow or support the same level of formal verification as Petri nets. Thus, while UML/BPMN or other semi-formal methods are widely used for modelling, Petri nets stand out for their analysability and ability to explicitly represent concurrency and resource constraints – a crucial advantage when transforming and modernizing complex legacy behaviours.

d. Gap in Current Research

While researchers have proposed various methods for extracting and validating workflows in modern systems, their application to legacy IT systems remains limited due to:

- The lack of formal validation techniques that ensure workflows are reconstructed accurately.
- The difficulty of handling concurrent operations and fallback mechanisms, common in legacy systems.

- The absence of simulation and optimization tools to validate workflows dynamically before modernizing the system.

Petri Nets provide a promising solution to address these gaps by offering a mathematical framework for rigorous modelling and validation. This study builds on the established foundations of Petri Nets and integrates them into the context of requirement extraction and validation for legacy IT systems, providing a holistic and robust methodology for solving longstanding challenges in system modernization.

Problem Statement

Legacy IT systems are foundational to the operations of many organizations, providing critical business functionalities and maintaining compliance with regulatory frameworks. However, these systems are often inherently complex, outdated, and poorly documented, making modernization both necessary and challenging. The modernization process requires careful extraction and validation of the workflows and requirements embedded within these systems, but several issues consistently hinder progress.

One of the most significant problems in modernizing legacy systems is the lack of comprehensive documentation. Many of these systems were developed decades ago, and essential business logic, workflows, and dependencies are often buried within obsolete codebases, inaccessible data structures, or are only known anecdotally by long-serving employees. This absence of clear documentation leads to inefficiencies in the requirements extraction process and creates a significant risk of omitting critical functionalities during modernization.

Additionally, legacy systems frequently exhibit tightly coupled dependencies that are not easily visible at first glance. The workflows within such systems are complex, involving hidden interdependencies, fallback mechanisms, and concurrent operations that vary widely across implementations. Identifying and properly modelling these workflows is vital to ensure that the modernized system replicates or improves upon existing functionality without introducing errors or inconsistencies.

Furthermore, validation of extracted requirements poses a unique challenge. While reverse engineering and process mining techniques can recover workflows or partially reconstruct undocumented processes, there is often no formal mechanism to validate the correctness, completeness, and scalability of these reconstructed workflows. Without such validation, organizations face the risk of implementing incomplete or incorrect workflows during modernization, potentially leading to operational disruptions, regulatory non-compliance, or data integrity issues.

Modern techniques like process mining and prototyping offer partial solutions but do not adequately address these challenges. For instance, process mining relies on complete and reliable log data, which may not always exist in legacy systems. Prototyping, while useful for iterative development, lacks the ability to formally model system behaviours like concurrency, synchronization, and fallback mechanisms. These gaps necessitate a structured and analytical approach to extracting and validating requirements from legacy IT systems.

The problem is further complicated by the dynamic nature of workflows in legacy systems, where activities such as data preprocessing, authorization checks, and fallback processing are critical to the overall functionality. However, the logic for these processes often spans multiple system modules with little transparency, adding to the difficulty of correctly capturing and modelling them.

To address these challenges, there is a need for a formal framework that enables the structured extraction, modelling, and validation of workflows and requirements in legacy systems. Such a framework should:

1. Provide a clear representation of workflows, capturing conditions, transitions, dependencies, and concurrency.

2. Enable formal validation techniques to ensure that reconstructed workflows are complete, correct, and optimized.

3. Offer the ability to analyse fallback mechanisms and failure scenarios, ensuring that system integrity is maintained, even under exceptional conditions.

This research identifies Petri Nets as a powerful tool for tackling these issues. Petri Nets provide a mathematical yet visually intuitive framework for modelling workflows, capturing complex dependencies, and validating workflows through properties such as reachability, liveness, and boundedness. By leveraging Petri Nets, it becomes possible to formally extract requirements, verify their accuracy, and identify potential inefficiencies or areas for optimization. These capabilities make Petri Nets an ideal solution for navigating the complexities of legacy system modernization while minimizing risk and ensuring business continuity.

Main Material Presentation

The modernization of legacy IT systems requires a structured methodology to extract, model, and validate workflows often buried in undocumented systems. This section outlines the proposed Petri Net-based approach, addressing challenges such as hidden interdependencies, lack of documentation, and complex dependencies. Using the credit report creation workflow as an example, Petri Nets are shown to be highly suitable for reconstructing and analysing system operations, leveraging their ability to handle concurrency, synchronization, and distributed processes effectively.

a. Methodology for Extracting, Modelling, and Validating Requirements

The first step involves extracting requirements from legacy systems that often lack documentation by retrieving, categorizing, and structuring workflows hidden in operational processes, code, logs, and stakeholder knowledge. Techniques include reverse engineering, which analyses system logs, audit trails, source code, and interfaces to trace workflows and dependencies, particularly for tightly coupled components like data validation and authorization in credit report workflows. Stakeholder engagement gathers insights from users and domain experts on undocumented rules, compliance needs (e. g., OFAC and MLA), and fallback mechanisms for missing data. Process mining uses logs and audit trails to reconstruct data flows, revealing transitions and dependencies critical for modernization efforts.

This step provides the raw requirements that will be formally represented using the Petri Net framework.

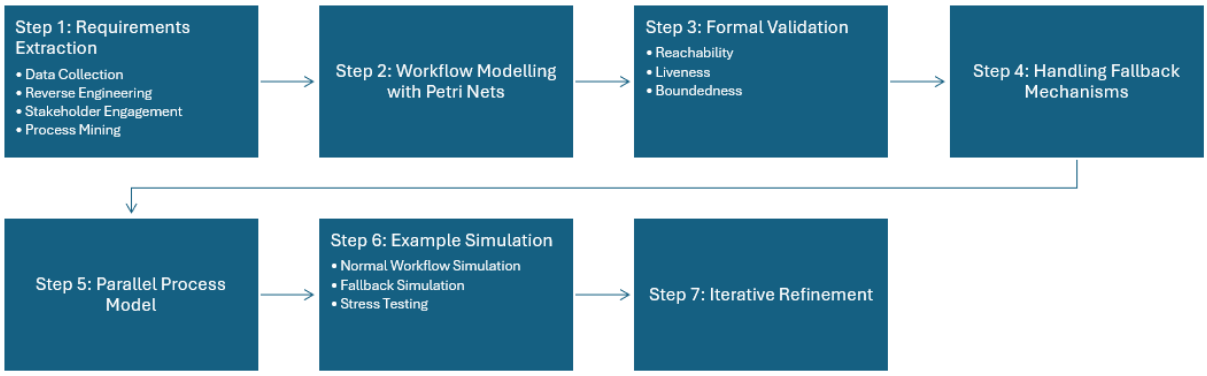

Fig. 1. Integrated Steps for Extracting and Validating Workflows

Once requirements are extracted, workflows are modelled as a Place / Transition (P/T) Petri Net to represent states, transitions, and dependencies mathematically and visually. In the credit report creation workflow, places represent conditions like “Request Received”, transitions capture actions such as “Validate Request”, tokens simulate system states moving through workflows, and arcs define dependencies. The resulting Petri Net provides a foundation for validating workflows through reachability, ensuring critical states like “Final Report Generated” are accessible; liveness, confirming no part of the workflow stalls; and boundedness, preventing token overflow in places like preprocessing or authorization. Tools such as CPN Tools or PIPE2 evaluate these properties to ensure workflow correctness and reliability.

Fallback mechanisms, essential for workflow continuity, are modelled to handle incomplete data by triggering alternate workflows, such as external utility data retrieval or generating a “No-Hit Report” when no data is available. Concurrent processes, like authorization checks for product types, OFAC data, and MLA data, are run in parallel, ensuring flexible and efficient resource use without delays or conflicts. Simulations

validate the system's functionality under various scenarios: normal workflow simulation validates token progression and state transitions, fallback simulation ensures alternate paths produce correct results, and stress testing evaluates scalability under high workloads. Metrics such as throughput, bottlenecks, and resource usage guide the iterative refinement of the model to address inefficiencies and optimize performance.

Petri Nets not only provide a structured framework for reconstructing undocumented workflows but also serve as a qualitative communication tool for stakeholders, abstracting technical complexities into accessible models. By leveraging Petri Nets' ability to handle concurrency, modularity, and formal analysis, organizations can reconstruct workflows, validate requirements pre-implementation, minimize modernization risks, and optimize legacy system functionality while ensuring compliance with evolving business needs.

b. Use case: The Process of Credit Report Creation for American Citizens Requested by Governmental or Commercial Entities

The implementation of credit report creation in legacy systems is shaped by governmental requirements, which vary due to regional regulations, data privacy laws, and system-specific features. Legacy systems often lack documentation, making retrieval, mapping, and analysis of workflows essential. The workflow in this study was created using the "Methodology for Extracting, Modelling, and Validating Requirements", which combines reverse engineering, stakeholder engagement, process mining, formal workflow modelling, and mathematical validation. This approach ensures accurate representation, validation, and handling of concurrency, fallback mechanisms, and compliance requirements. Credit report creation involves stages such as data collection, validation, authorization, processing, and report generation, demonstrating the methodology's ability to reconstruct workflows effectively.

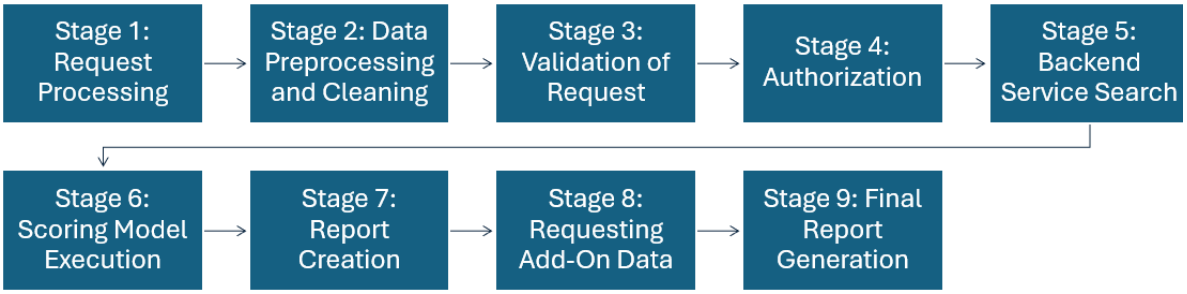

Fig. 2. Workflow Stages in Credit Report Creation

Stage 1 Request Processing: the process begins with the receipt of a request, specifying the type, format, product, PII (e.g., name, addresses, SSNs), and add-ons such as OFAC and MLA compliance or scoring models (e.g., FICO). These parameters define the scope, depth, and content of the report while ensuring the appropriate compliance and scoring frameworks are applied.

Stage 2 Data Preprocessing and Cleaning: data preprocessing ensures accuracy by cleaning user data, addressing inconsistencies or missing fields, and normalizing formats. Address standardization aligns user address information with postal or geographic standards for consistency.

Stage 3 Validation of Request: the system validates the request by checking the completeness of required fields, including product type, PII, and report parameters, to ensure that the system can fulfil the request properly.

Stage 4 Authorization: this stage includes multiple layers of authorization checks for compliance with product type access, OFAC regulations, MLA lending protections, and permissions for scoring models. Each layer ensures the requester has the necessary permissions to access specific components of the system.

Stage 5 Backend Service Search: after validation and authorization, the system queries backend services to locate user files. Possible outcomes include files being retrieved, unavailable, or blocked due to legal restrictions, triggering fallback measures if needed.

Stage 6 Scoring Model Execution: when user files are successfully retrieved, the credit data is processed through a scoring model, calculating creditworthiness based on the specified scoring parameters.

Stage 7 Report Creation: in this stage, the system generates the credit report, containing personal details, credit history, compliance verification (e. g., OFAC and MLA), and credit scores. This report is formatted according to the initial request.

Fallback mechanisms handle missing or incomplete data by sending requests to external sources (e. g., utility bills) to fill data gaps. If no data is retrieved, the system generates a "No Hit Report" to provide a response indicating insufficient information.

Stage 8 Requesting Add-On Data: the system may initiate additional workflows to retrieve flagged compliance-related data such as OFAC and MLA information, ensuring the completeness and accuracy of the report under regulatory requirements.

Stage 9 Final Report Generation: regardless of the fallback outcomes or data availability, the system completes the process by generating the final report, ensuring that every request receives a formal response.

Despite the outlined steps, legacy systems often exhibit inconsistencies due to historical design decisions and outdated practices. For instance, some systems use variable-length SSNs instead of fixed formats, hard-code authorization rules, lack fallback mechanisms, or rely on proprietary scoring models instead of standardized ones like FICO. These discrepancies emphasize the need for careful retrieval, standardization, and modernization of legacy workflows to ensure compliance and operational reliability.

c. Petri Net model for credit report request processing example

The Petri Net model presented in this section is the final result of applying the "Methodology for Extracting, Modelling, and Validating Requirements" to the process of credit report creation for American citizens, as discussed in the previous subsection. This model was constructed by systematically analysing the legacy system, combining reverse engineering, stakeholder input, and process mining techniques to extract workflows and requirements. Based on the extracted requirements, the workflows were first organized into structured stages, such as data preprocessing, authorization checks, backend service interactions, and fallback mechanisms. Each identified step was then formalized into the Petri Net framework, capturing the system's states, transitions, and dependencies.

This Petri Net model serves as a faithful representation of the reconstructed credit report processing workflow, demonstrating how tokens (representing requests) flow through various states and transitions. It visually and mathematically captures the system's behaviour, allowing it to be analysed for critical properties such as reachability, liveness, and boundedness. Below, the places and transitions in the Petri Net are described in detail, followed by an explanation of how the model operates to fulfil the credit report request.

Fig. 3 shows the Petri Net model of the workflow. The legends of places and transitions in this model are as follows:

P0: Request Received – represents the initial point when the system receives the request to create a credit report, holds personal data required to process the request (e. g., name, SSN, address).

P1: Data Pre-processed – represents the state where data cleaning and normalization are complete.

P2: Address Standardized – indicates that user addresses have been standardized within the system.

P3: Request Validated – represents the validation step where the system checks for adequate data completeness.

P4: Request Contains Type of Product – Represents the type of product specified in the request.

P5: Service Returns User File – Denotes successful retrieval of a user file from the backend service.

P6: File Blocked – Handles the case where the user file is found but blocked by external agencies.

P7: No User File Found – Represents failure of the backend service to locate a user file.

P8: Credit File Scored – Represents the point where the credit file is processed in the scoring model pipeline.

P9: Utility Bills has not been Found – Represents failure of the retrieval of utility bill data from external systems

P10: Utility Bills Found – Represents the retrieval of utility bill data from external systems.

P11: Utility Bills Scored – Indicates the utility bill data sent to the scoring process pipeline.

P12: No Hit Report Created – Represents the creation of a report when no user file or utility bill data is retrieved from any source.

P13: Addon – OFAC Data – Indicates whether OFAC data is required (depends on the product type).

P14: Authorized for OFAC Data – Represents authorization to handle OFAC addon data.

P15: Additional OFAC Data Requested – Denotes that OFAC data is explicitly requested when other attempts fail.

P16: Addon – MLA Data – Indicates whether MLA data is required (depends on the product type).

P17: Authorized for MLA Data – Indicates authorization to process MLA addon data.

P18: Additional MLA Data Requested – Denotes that MLA data is explicitly requested when other attempts fail.

P19: Additional Data Received – Denotes that additional data is received.

P20: Request Contains Type of Report – Holds the request's specified type.

P21: Request Contains Report Format – Ensures the report format is defined.

P22: Authorized for Scoring Model – Represents scoring model authorization for the system.

P23: Authorized for Product Type – Indicates that the requester is authorized for the specified product type.

P24: Final Report Generated – Indicates the generation of the report regardless of data availability.

T0: Start Data Preprocessing – Cleans and normalizes PII and other input data.

T1: Standardize Address – Implements address formatting and cleaning.

T2: Validate Required Data – Checks if all necessary fields for creating the report are populated.

T3: Send Data to Service – Represents sending the request to the backend service for user data.

T4: Send Credit File to Scoring Model – Sends retrieved credit file data to the scoring model pipeline.

T5: No Hit Report Creation – Represents creating a "no hit" report when no valid data is found.

T6: Send Request to External System – Sends a request to external services like utility bill databases.

T7: Process Utility Bills –Handles external data processing and scoring based on utility bill data.

T8: Check OFAC Authorization – Verifies authorization for handling OFAC addon data.

T9: Check MLA Authorization – Verifies authorization for processing MLA addon data.

T10: Request Additional OFAC and MLA Data – Makes explicit requests for these addons when primary sources fail.

T11: Generate Final Report – Represents the final step where the system generates the report, regardless of data availability.

The workflow begins in P0 (Request Received) when the system receives a new credit report request. The request is first pre-processed via T0 (Start Data Preprocessing), moving tokens to P1 (Data Pre-processed). The address standardization process is triggered via T1 (Standardize Address), progressing the workflow to P2 (Address Standardized). Subsequently, T2 (Validate Required Data) checks whether the request contains adequate and valid data to proceed, transitioning tokens to P3 (Request Validated).

If the request specifies the type of product, T3 (Send Data to Service) advances tokens to P4 (Request Contains Type of Product), ensuring the workflow proceeds to the backend service operation. Authorization checks for specific addons (e. g., OFAC, MLA) or product types are validated through transitions T8 (Check OFAC Authorization) and T9 (Check MLA Authorization), advancing tokens through places P13 (Addon – OFAC Data), P14 (Authorized for OFAC Data), P16 (Addon – MLA Data), and P17 (Authorized for MLA Data) accordingly.

The backend service operation begins with T3 (Send Data to Service), sending the user's request to the backend service. Based on the service results:

- If the user file is successfully retrieved, tokens move to P5 (Service Returns User File).
- If the file is blocked, tokens progress to P6 (File Blocked) via T9.
- If no file is found, T10 (No User File Found) transitions tokens to P7 (No User File Found).

Once a valid user file is retrieved, T4 (Send Credit File to Scoring Model) triggers the credit scoring process, transitioning tokens to P8 (Credit File Scored). The scoring model ultimately results in the generation of the final report through T11 (Generate Final Report), moving tokens to P24 (Final Report Generated).

If missing data (user file or utility bills) is identified, further actions are taken:

- T6 (Send Request to External System) transitions tokens to P10 (Utility Bills Found), retrieving utility bills.
- If utility bill retrieval fails, T7 (Process Utility Bills) transitions tokens to P9 (Utility Bills has not been Found).
- Additional attempts to acquire required data (e. g., OFAC or MLA data) begin with T10 (Request Additional OFAC and MLA Data), transitioning tokens to P18 (Additional MLA Data Requested) and P15 (Additional OFAC Data Requested).

Scoring for utility bills proceeds with T7 (Process Utility Bills), ultimately placing tokens in P11 (Utility Bills Scored). Combined with other incoming data (e. g., retrieved user files, utility bill scores), tokens are sent to P19 (Additional Data Received) via T12.

When no valid data is retrieved from any source, the system generates a "No Hit Report" through T5 (No Hit Report Creation), advancing tokens to P12 (No Hit Report Created). Regardless of the data availability, T11 (Generate Final Report) ensures that a final report is produced, finalizing the workflow at P24 (Final Report Generated).

This workflow model ensures the generation of a report for all valid requests, with detailed data processing, validation, scoring pipelines, and conditional handling of missing or blocked data.

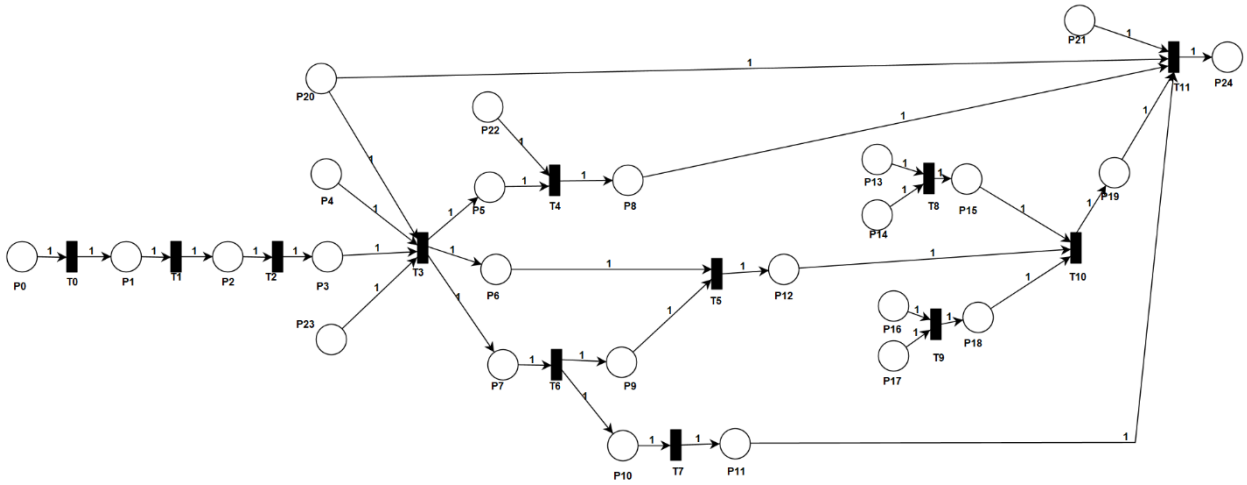

Fig. 3. Petri net model of the credit report creation in Fig. 2

Results and Discussion

The Petri Net-based approach effectively demonstrated its ability to model, validate, and simulate workflows extracted from legacy IT systems, addressing challenges such as incomplete documentation, concurrency, fallback mechanisms, and scalability constraints. It provided a structured representation of critical states and transitions, capturing milestones like "Request Received" and "Final Report Generated", alongside processes such as validation and authorization. Formal validation ensured correctness and reliability, with reachability confirming all required states were accessible, liveness preventing deadlocks, and boundedness controlling token flow to avoid bottlenecks. Dynamic simulations validated the workflow under normal operations, fallback scenarios, and high-volume stress tests, highlighting scalability and smooth resource utilization without performance degradation.

The approach proved adaptable to evolving requirements, enabling extensions for credit risk analysis or new compliance mechanisms while preserving its formal structure and validation properties. It was particularly effective at reconstructing undocumented workflows using techniques like reverse engineering, process mining, and stakeholder insights, ensuring robustness during modernization. Unlike manual reconstruction or incomplete process mining, Petri Nets provided precision by handling all potential states and transitions with rigor.

Despite its strengths, the methodology's success relied on the completeness of requirements extraction, and constructing comprehensive Petri Nets required substantial expertise and time. Computational limits were also noted in real-time simulations of highly concurrent workflows. However, its ability to optimize performance, validate failure scenarios, and ensure compliance makes the approach invaluable for organizations modernizing legacy systems. This structured method minimizes risks, ensures operational continuity, and prepares systems for scalability and adaptability. Future research could explore advanced variants like Timed or Stochastic Petri Nets for enhanced applicability in performance-constrained scenarios.

Conclusions

Legacy IT systems pose significant challenges during modernization due to their complexity, outdated technologies, and lack of documentation. These systems often encode essential workflows and compliance mechanisms critical for meeting regulatory and business requirements. Extracting, understanding, and modelling these embedded requirements is crucial for ensuring consistency, operational continuity, and progress toward modern infrastructure.

This article presented a detailed approach to these challenges, focusing on Petri Nets as a robust tool for modelling and analysing workflows extracted from legacy systems. The methodology addressed issues such as incomplete data, hidden dependencies, and unclear processes, combining techniques like reverse engineering, stakeholder engagement, and validation to reconstruct workflows systematically. Petri Nets provide a structured framework that represents concurrency, synchronization, and dependencies, enabling precise modelling of conditions (places), transitions, and arcs in legacy processes. Their versatility was demonstrated through applications to real-world scenarios, specifically the credit report creation workflow for American citizens.

The credit report case study illustrated how Petri Nets capture complex processes, including user data preprocessing, authorization checks, backend service integration, and fallback mechanisms. By defining states like "Request Received" and transitions like "Validate Required Data", Petri Nets effectively modeled workflows while accounting for concurrency and dependencies. Detailed analysis of properties such as reachability, liveness, and boundedness confirmed the reliability and robustness of the constructed model.

This article also emphasized discrepancies in legacy system implementations due to design choices, system-specific constraints, or custom-built solutions, which pose barriers to standardization. Petri Nets address these irregularities by reconstructing and optimizing workflows without compromising process integrity. Beyond modelling, Petri Nets enable formal validation to ensure the accuracy and completeness of workflows, simulation to test and refine processes, and incremental modernization to update subsystems methodically while preserving overall functionality. Furthermore, their graphical representation facilitates alignment between technical and non-technical stakeholders.

In conclusion, Petri Nets not only streamline the reconstruction of undocumented workflows but also empower organizations to modernize legacy systems with precision and scalability. The case study validated their utility in addressing challenges related to undocumented processes, regulatory compliance, and operational constraints. By providing a structured methodology and rigorous analysis, this approach equips organizations to modernize legacy systems confidently while ensuring adaptability for future needs.

References

[1] Hogan, G., Shalkauskaite, P., Zhu, M., Derwin, M., Yilmaz, M., McCarren, A., & Clarke, P. (2024). Investigating Systems Modernisation: Approaches, Challenges and Risks. *Systems, Software and Services Process Improvement*, 147–162. https://doi.org/10.1007/978-3-031-71139-8_10

[2] Strobl, S., Bernhart, M., & Grechenig, T. (2020). Towards a Topology for Legacy System Migration. *ICSEW'20: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 586–594. https://doi.org/10.1145/3387940.3391476

[3] Akhtyamov, R., Vingerhoeds, R., & Golkar, A. (2018). Measures and Approach for Modernisation of Existing Systems. *2018 IEEE International Systems Engineering Symposium (ISSE)*, 1–8. https://doi.org/10.1109/SysEng.2018.8544427

[4] Leemans, M., van der Aalst, W. M. P., van den Brand, M. G. J., Schiffelers, R. R. H., & Lensink, L. (2018). Software Process Analysis Methodology: A Methodology Based on Lessons Learned in Embracing Legacy Software. *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 665–674. https://doi.org/10.1109/ICSME.2018.00076

[5] Strobl, S., Zoffi, C., Haselmann, C., Bernhart, M., & Grechenig, T. (2020). Automated Code Transformations: Dealing with the Aftermath. *2020 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, London, ON, Canada.

[6] Sahlabadi, M., Muniyandi, R. C., Shukur, Z., Qamar, F., & Kazmi, S. H. A. (2022). Process Mining Discovery Techniques for Software Architecture Lightweight Evaluation Framework. *Computers, Materials and Continua, 74*(3), 5777–5797. https://doi.org/10.32604/cmc.2023.032504

[7] Pika, A., Ouyang, C., & Hofstede, A. H. M. (2022). Configurable Batch-Processing Discovery from Event Logs. *ACM Transactions on Management Information Systems (TMIS), 13*(3), 1–25. https://doi.org/10.1145/3490394

[8] Castillo, R. P., Weber, B., & Piattini, M. (2011). Process Mining Through Dynamic Analysis for Modernising Legacy Systems. *IET Software, 5*(3), 304–319.

[9] Kernytskyy, O., Kernytskyy, A., Melnyk, M., Łukaszewicz, A., Pytel, K., & Banas, M. (2024). Applying Software Development Black-Box, Grey-Box and White-Box Reverse Engineering Frameworks to the Mechanical Industry. *CAD in Machinery Design: Implementation and Educational Issues*, 233–240.

[10] Moraga, M., & Zhao, Y. (2018). Reverse Engineering a Legacy Software in a Complex System: A Systems Engineering Approach. *INCOSE International Symposium, 28*(1), 1250–1264. https://doi.org/10.1002/j.2334-5837.2018.00546.x

[11] Ponnusamy, S., & Eswararaj, D. (2023). Navigating the Modernization of Legacy Applications and Data: Effective Strategies and Best Practices. *Asian Journal of Research in Computer Science, 16*(4), 239–256. https://doi.org/10.9734/ajrcos/2023/v16i4386

[12] Tsai, J. C. A., Jiang, J. J., Klein, G., & Hung, S. Y. (2022). Legacy Information System Replacement: Pursuing Quality Design of Operational Information Systems. *Information & Management, 59*(1), 103592. https://doi.org/10.1016/j.im.2022.103592

[13] Miskell, C., Diaz, R., Ganeriwala, P., Slhoub, K., & Nembhard, F. (2023). Automated Framework to Extract Software Requirements from Source Code. *ACM NLPIR, South Korea*. https://doi.org/10.1145/3639233.3639242

[14] Southwick, D., Resch, G., & Ratto, M. (2021). Iterative Prototyping and Co-Design. *Knowledge, Innovation, and Impact*, 231–238. https://doi.org/10.1007/978-3-030-34390-3_30

[15] He, X. (2013). A Comprehensive Survey of Petri Net Modeling in Software Engineering. *International Journal of Software Engineering and Knowledge Engineering, 23*(5). https://doi.org/10.1142/S021819401340010X

[16] Han, D., Wang, C., Bian, G., Shao, B., & Shi, T. (2023). A Novel Process Recommendation Method That Integrates Disjoint Paths and Sequential Patterns. *MDPI, Applied Sciences, 13*(6), 3894. https://doi.org/10.3390/app13063894

[17] Zhu, H., Chen, J., Cai, X., Ma, Z., Jin, R., & Yang, L. (2019). A Security Control Model Based on Petri Net for Industrial IoT. *2019 IEEE International Conference on Industrial Internet (ICII)*. https://doi.org/10.1109/ICII.2019.00040

[18] Yang, C. H., Lin, Y. N., Shen, V. R. L., Shen, F. H. C., & Jheng, W. S. (2023). A Novel IoT-Enabled System for Real-Time Monitoring Home Appliances Using Petri Nets. https://doi.org/10.36227/techrxiv.22362472.v1

[19] Bouyakoub, S., & Belkhir, A. (2022). Things-Net: A Hierarchical Petri Net Model for Internet of Things Systems. *International Journal of Software Innovation, 10*(1), 1–27. https://doi.org/10.4018/IJSI.297981

[20] Wang, J. (2022). Healthcare Patient Flow Modeling and Analysis with Timed Petri Nets. *Advances in Computing, Informatics, Networking and Cybersecurity*, 181–204. https://doi.org/10.1007/978-3-030-87049-2_6

[21] Jabbar, M. A., & Hussain, M. (2022). Formal Modeling and Analysis of Integrated Healthcare System Using Colored Petri Nets. *VFAST Transactions on Software Engineering, 10*(2), 211. http://vfast.org/journals/index.php/VTSE

[22] Sobrinho, A., Almeida, I., Chaves e Silva, L. D. D., Araújo, A., Costa, T. F. F., & Perkusich, A. (2022). Coloured Petri Nets for Abstract Test Generation in Software Engineering. *Wiley, Software Testing, Verification & Reliability, 1*(6), 1. https://doi.org/10.1002/stvr.1837

Олег Керницький[1], Андрій Керницький[2]

[1] Кафедра автоматизованих систем управління, Національний університет "Львівська політехніка", вул. С. Бандери, 12, Львів, Україна, oleh.b.kernytskyy@lpnu.ua, ORCID 0009-0007-5318-6506
[2] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери, 12, Львів, Україна, andriy.b.kernytskyy@lpnu.ua, ORCID 0000-0001-8188-559X

ПІДХІД НА ОСНОВІ МЕРЕЖ ПЕТРІ ДО ВИДОБУВАННЯ ТА ПЕРЕВІРКИ ВИМОГ ДЛЯ МОДЕЛЮВАННЯ ТА МОДЕРНІЗАЦІЇ АРХАЇЧНИХ ІТ-СИСТЕМ

Отримано: Серпень 12, 2025 / Переглянуто: Серпень 25, 2025 / Прийнято: Вересень 15, 2025

Анотація. Застарілі ІТ-системи мають вирішальне значення для функціонування багатьох організацій, але створюють значні труднощі для модернізації через їхню складність, застарілі технології та відсутність належної документації. Вилучення та розуміння робочих процесів і вбудованих вимог у таких системах важливі для їх адаптації до сучасної інфраструктури, зберігаючи відповідність нормативним вимогам та операційну цілісність. У цій статті досліджено використання мереж Петрі як формального та графічного інструменту моделювання для вирішення цих проблем. Мережі Петрі пропонують потужну основу для відображення системних робочих процесів, фіксації паралельності, синхронізації та процесів прийняття рішень. Методологія починається зі структурованого підходу до визначення вимог із застарілих систем за допомогою зворотного проєктування, залучення зацікавлених сторін та аналізу даних. Із використанням процесу створення кредитного звіту як тематичного дослідження проілюстровано, як ефективно мережі Петрі можуть моделювати складні робочі процеси, зокрема попереднє опрацювання даних, перевірку авторизації, інтеграцію серверних служб та резервні механізми.

Результати демонструють їхній потенціал для формального аналізу, комунікації між технічними та нетехнічними зацікавленими сторонами та поступової модернізації застарілих систем. Висвітлено, як мережі Петрі дають організаціям змогу точно реконструювати недокументовані робочі процеси, забезпечуючи сталу модернізацію і зберігаючи основні функціональні можливості. Надаючи детальну методологію та аналіз мереж Петрі, це дослідження пропонує структуровану основу для вирішення проблем застарілих систем та просування інноваційних стратегій модернізації.

Ключові слова: мережі Петрі, застарілі ІТ-системи, моделювання робочих процесів, видобування вимог, формальна валідація, модернізація системи, резервні механізми, аналіз паралельності, динамічне моделювання.

[*] Corresponding author