Grenze International Journal of Engineering and Technology, January Issue

Systematic Approaches to Petri Net-based PLC Code Generation for Industrial Automation

V. B. Kumbhar[1] and M. S. Chavan[2]
[1]PhD Scholar, Shivaji University Kolhapur, Maharashtra, India
Email: vinodkumbhar2012@gmail.com
[2]Professor, Department of Electronics and Telecommunication Engineering, KITCOE Kolhapur, Maharashtra, India
Email: @hostname2.org

***Abstract*— This review examines the significant advancements in translating Petri nets into Programmable Logic Controller (PLC) code for industrial automation applications. Petri nets provide a powerful formal method for modeling complex concurrent systems, while PLCs remain the cornerstone of industrial control. This article analyzes the challenges and solutions in bridging these domains, with particular focus on recent methodologies including direct conversion approaches, signal interpreted Petri nets, and state diagram intermediaries. We examine multi-vendor support capabilities, verification techniques, and safety integration features across different implementation approaches. The review highlights the substantial benefits of automated conversion, including reduced development time, lower error rates, and improved system reliability, supported by case studies across manufacturing, process control, and material handling domains. Finally, we identify research gaps and future directions including hybrid system modeling, distributed control architectures, and integration with Industry 4.0 technologies, providing a comprehensive roadmap for researchers and practitioners in industrial automation.**

***Index Terms*— Petri nets, Programmable Logic Controllers, Industrial automation, Formal verification, Model-driven engineering, Safety-critical systems.**

I. INTRODUCTION

The industrial automation landscape has undergone significant transformation in recent decades, driven by increasing demands for efficiency, flexibility, and reliability in manufacturing and process control systems. At the heart of this evolution is the ongoing challenge of translating theoretical models of system behavior into practical, executable control programs for industrial hardware.

Petri nets, first introduced by Carl Adam Petri in 1962 [1], have emerged as a powerful mathematical and graphical modeling tool for representing complex systems with concurrent, asynchronous, and distributed behaviors. Their formal foundation provides rigorous verification capabilities that ensure system correctness before implementation. However, a significant gap exists between these theoretical models and their practical implementation in industrial controllers.

This review article examines the state of the art in bridging this gap through Petri net to PLC conversion systems. We analyze various methodologies, tools, and approaches that have been developed to translate formal Petri net models into executable PLC code.

Grenze ID: 01.GIJET.12.1.343

The review encompasses both academic research and practical applications, providing a comprehensive understanding of the current landscape and identifying promising directions for future development.
The integration of formal methods like Petri nets into industrial automation practices offers substantial benefits, including improved system reliability, reduced development time, and enhanced maintenance capabilities. By systematically analyzing the conversion approaches, this review aims to provide valuable insights for researchers and practitioners seeking to leverage these advantages in industrial settings.

II. FOUNDATIONS OF PETRI NETS AND PLCS

A. Petri Net Theory and Applications

Petri nets represent a powerful mathematical and graphical formalism for modeling concurrent, asynchronous, and distributed systems. The fundamental structure of a Petri net consists of places, transitions, arcs, and tokens, which together create a dynamic model of system behavior.
A Petri net is formally defined as a 5-tuple $PN = (P, T, F, W, M_0)$ where:
P is a finite set of places, representing conditions or states
T is a finite set of transitions, representing events or actions
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs connecting places and transitions
$W: F \rightarrow \{1, 2, 3, ...\}$ is a weight function assigning weights to arcs
$M_0: P \rightarrow \{0, 1, 2, 3, ...\}$ is the initial marking defining the initial state
This foundational structure has been extended into various specialized forms to address different modeling needs:
Elementary Nets: The simplest form, allowing only one token per place. While limited in expressiveness, they efficiently model basic control logic with minimal computational overhead.
Timed Petri Nets: Incorporate timing constraints for performance evaluation and real-time systems modeling [3]. By associating time delays with transitions or places, these models enable quantitative analysis of system performance metrics like throughput and cycle times.
Colored Petri Nets: Introduced by Jensen [2], these nets incorporate data types (colors) to tokens, enabling complex data manipulations while reducing model size. This extension permits handling of structured information flow and implementing data-dependent control logic.
Hierarchical Petri Nets: Allow abstraction and refinement through subnet structures [4], enabling management of complexity in large-scale systems through encapsulation of subsystems as single elements at higher levels.
Signal Interpreted Petri Nets (SIPNs): Enhance traditional Petri nets with input/output signals to facilitate direct interface with PLC hardware, creating a more natural bridge between the theoretical model and practical implementation.
In industrial contexts, Petri nets have found applications across numerous domains:

- Manufacturing systems for modeling flexible manufacturing cells, production lines, and resource allocation [4]
- Process control for designing and verifying control logic in continuous and batch processes [5]
- Robotics and automation for sequence control and coordination
- Business process modeling for workflow management

The formal properties of Petri nets, including reachability, boundedness, liveness, and deadlock-freedom, provide critical insights for system verification before implementation, making them particularly valuable for safety-critical applications.

B. PLC Architecture and Programming

Programmable Logic Controllers serve as the backbone of industrial automation, providing robust and reliable control capabilities in harsh industrial environments. Modern PLCs typically comprise:
Central Processing Unit (CPU): Executes the control program and manages system operations through scan cycle processing
Input/Output (I/O) System: Interfaces with field devices including sensors and actuators
Memory System: Stores the control program and data, including program memory and data memory
Communication Interfaces: Enable networking and integration with other systems
Power Supply: Provides electrical power to PLC components
The operation of a PLC follows a cyclical pattern:

- Input scan - reading the state of all input devices
- Program execution - processing the control logic

- Output scan - updating the state of output devices
- Housekeeping - performing diagnostic checks and communication tasks

The IEC 61131-3 standard defines five programming languages for PLCs [12]:

Ladder Diagram (LD): A graphical language based on relay ladder logic diagrams, remaining the most widely used due to its intuitive visualization of control logic and accessibility to maintenance personnel with electrical backgrounds.

Function Block Diagram (FBD): A graphical language representing signal flow through blocks, well-suited for continuous process control applications and complex algorithms.

Structured Text (ST): A high-level text-based language similar to Pascal or C, providing powerful capabilities for complex mathematical operations, iterative loops, and data structure manipulation.

Instruction List (IL): A low-level assembler-like language with accumulator-based operations, offering efficient execution for simple boolean logic.

Sequential Function Chart (SFC): A graphical language based on Grafcet for sequential control operations, excelling in batch processing and procedural control applications.

Modern PLCs incorporate advanced features including integrated motion control, sophisticated process control functions, complex data handling capabilities, safety functions, web server capabilities, and increasingly, object-oriented programming paradigms.

The evolution of PLCs continues with Industry 4.0 initiatives, incorporating features such as OPC UA integration for standardized communication, edge computing capabilities for local data processing, enhanced cybersecurity features, digital twin integration, and AI/machine learning capabilities.

III. Petri Net to PLC Conversion Methodologies

A. Direct Translation Approaches

Direct translation methodologies focus on converting Petri net elements directly into PLC constructs without intermediate representations. These approaches typically map places to boolean variables and transitions to logical conditions in the target PLC programming language.

Several significant direct translation methodologies have emerged over time:

Token Passing Ladder Logic (TPLL): Developed by Uzam et al. [6], this approach works with Colored and Timed Petri Nets, converting them to Ladder Logic while preserving their colored and timed properties. It's particularly valuable for complex industrial applications requiring different token types and temporal constraints.

Self-holding Circuit Method: Introduced by Chirn & McFarlane (2000), this methodology specifically addresses the "avalanche effect" - where one change triggers a cascade of other changes. By using modified mapping rungs, it provides more stable and predictable control system behavior.

Hierarchical Petri Net Translation: Developed by Burns & Bidanda (1994), this approach automatically generates ladder logic from hierarchical structures, preserving organizational relationships that make resulting code easier to maintain and understand.

Table 1 provides a comparative analysis of key direct translation methodologies found in the literature.

Table I. Comparison of Direct Translation Methodologies

Methodology	Authors	Year	Source Language	Target Language	Key Features	Limitations
Token Passing Ladder Logic (TPLL)	Uzam et al.	1996	Colored/Timed Petri Nets	Ladder Logic	Direct conversion including colored and timed nets	Limited verification capabilities
Self-holding Circuit Method	Chirn & McFarlane	2000	Standard Petri Nets	Ladder Logic	Modified mapping rungs, addresses avalanche effect	Complexity increases with model size
Hierarchical Petri Net Translation	Burns & Bidanda	1994	Hierarchical Petri Nets	Ladder Logic	Automatic generation for hierarchical structures	Limited support for advanced properties
Rule-based System	Jafari & Boucher	1994	Standard Petri Nets	Ladder Logic	High-level system model translation	Lacks formal verification
PLC-based Implementation	Fernández Adiego et al.	2014	Petri Nets	Structured Text	Model checking integration	Implementation overhead

The implementation of direct translation approaches typically involves three key steps:

- Place-to-Variable Mapping: Each Petri net place is represented by a boolean variable in the PLC program, with TRUE/FALSE values indicating token presence or absence.
- Transition-to-Logic Mapping: Transition firing conditions are translated into logical expressions that check input place states and execute appropriate actions when conditions are met.

- Token Game Implementation: The token movement rules are implemented as control logic that handles the flow of tokens through the system according to the Petri net semantics.

While direct translation approaches offer simplicity and efficiency, they may lack the sophistication required for complex industrial applications, particularly regarding verification and safety considerations.

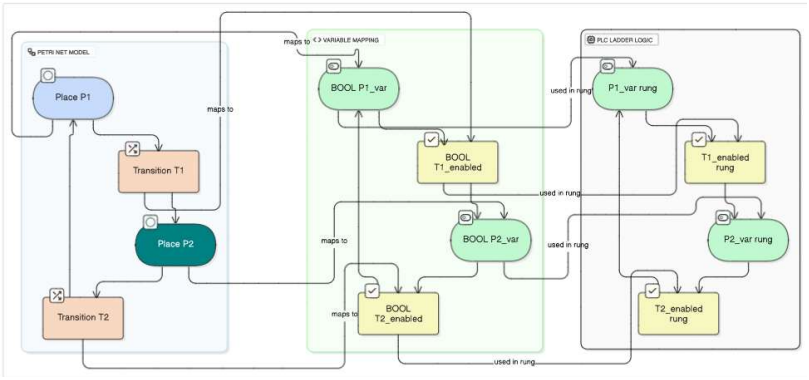

Figure 1: Direct translation process from Petri net elements to Ladder Logic constructs, showing the mapping of places to variables and transitions to logical conditions.

B. Intermediate Model-based Approaches

Intermediate model approaches use transitional representations to facilitate more sophisticated translations and enable verification capabilities. This methodology adds a layer between the Petri net model and the final PLC code to address semantic gaps and enhance verification possibilities.

Key intermediate model-based translation methods include:

Automata-based Intermediate Models: Developed by Fernández Adiego et al. [8], this approach uses synchronized automata as an intermediate representation. It supports multiple verification tools including nuXmv, UPPAAL, and BIP, with advanced reduction techniques such as Cone of Influence (COI), rule-based methods, and variable abstraction to manage complexity.

CAEX/MathML: Estévez & Marcos [9] created an approach utilizing XML-based metamodels as intermediate representations, offering excellent interoperability through standardization. Verification relies on Schematron rules with domain-specific patterns tailored for industrial applications.

Signal Interpreted Petri Nets: This approach enhances traditional Petri nets with input/output signals, creating a more natural bridge between the theoretical model and PLC implementation. SIPNs maintain the formal verification properties of Petri nets while directly representing the interface to the physical system.

Table 2 provides a comparative analysis of prominent intermediate model-based approaches for Petri net to PLC conversion.

TABLE II. COMPARISON OF INTERMEDIATE MODEL-BASED TRANSLATION METHODS

Method	Reference	Intermediate Model	Verification Support	Reduction Techniques	Target Languages	Execution Efficiency
Automata-based IM	Fernández Adiego et al., 2014	Synchronized Automata	nuXmv, UPPAAL, BIP	COI, rule-based, variable abstraction	ST, LD	Medium
CAEX/MathML	Estévez & Marcos, 2012	XML-based metamodel	Schematron rules	Domain-specific patterns	Multiple	Low
PetriDotNet	Baresi et al., 2002	Graph transformation	CTL properties	Graph-based reduction	LD, ST	Medium
Binary Petri Net	Cabral et al., 2015	State observer PN	LTL verification	Minimized state space	LD	High
Net Condition/Event Systems	Vyatkin et al., 2009	NCES	CTL, reachability analysis	Structural reduction	IEC 61499	Medium-High

The advantages of intermediate model approaches include:

- Enhanced verification capabilities through specialized formal verification tools
- Better abstraction of platform-specific details
- Improved handling of complex control patterns
- Support for safety property verification

However, these approaches may introduce additional complexity and potential overhead in the translation process. The effectiveness of an intermediate model depends significantly on how well it bridges the semantic gap between Petri nets and PLC execution models while maintaining the essential properties of the original model.

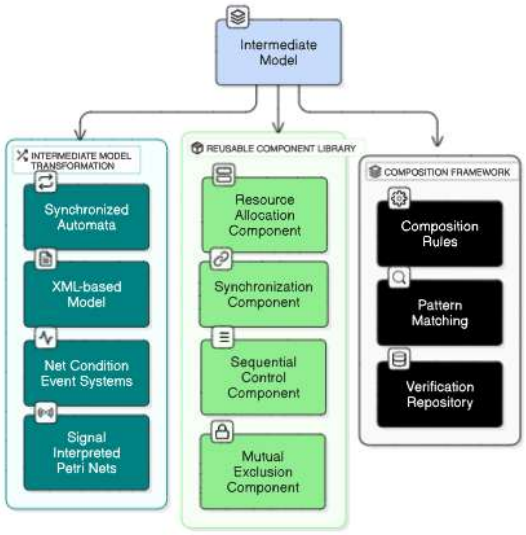

Figure 2: Architecture of an intermediate model-based translation system, showing the flow from Petri net model through intermediate representation to final PLC code with verification feedback loops.

C. Component-Based Approaches

Component-based methodologies focus on modular translation and reusability, leveraging pre-verified components to build complex systems. This approach aligns with modern software engineering practices and addresses scalability challenges in industrial applications.

Key aspects of component-based approaches include:

Petri Net Component Libraries: These specialized libraries contain commonly used control patterns encapsulated as parameterizable Petri net modules with proven properties and optimized PLC implementations. Examples include resource allocation mechanisms, mutual exclusion protocols, sequencing patterns, and synchronization mechanisms. The components are typically accompanied by formal documentation of their behavioral characteristics and verified properties.

Composition Rules: These formal rules govern how component instances can be interconnected while preserving critical system properties like safety, liveness, and boundedness. They define precise semantics for different connection types including synchronization through shared transitions, communication through place fusion, and hierarchical nesting relationships.

Table 3 compares different component-based approaches for Petri net to PLC conversion, highlighting their characteristics, benefits, and limitations.

TABLE III: COMPARISON OF COMPONENT-BASED APPROACHES

Approach	Authors	Year	Component Type	Composition Method	Verification Approach	Target Platform	Reusability Level
Module-based Translation	Vyatkin & Hanisch	2001	Net Condition/Event Systems	Signal-based composition	Modular verification	IEC 61499	High
Formal Composition	Hagge & Wagner	2008	Petri net modules	Interface composition	Component-based verification	Siemens	Medium-High
Design Patterns	Peng & Zhou	2004	Petri net patterns	Pattern instantiation	Pattern-level verification	Multiple	High
Reusable Components	Thramboulidis	2005	Function blocks	Port-based composition	Interface contract verification	IEC 61131-3	High
Modular Synthesis	Leitner & Mahnke	2010	Parameterized nets	Hierarchical composition	Compositional model checking	OPC UA enabled	Medium

Vyatkin and Hanisch [13] pioneered the component-based approach with their module-based translation framework, later extended by Hagge and Wagner with formal composition semantics. These contributions established the foundation for component-based development of industrial automation systems using formal models, addressing scalability challenges through separate verification of individual components before integration.

The advantages of component-based approaches include:

- Improved reusability of verified components across projects
- Enhanced scalability for large, complex systems
- Reduced verification effort through composition of pre-verified components
- Better maintainability through standardized interfaces and documentation

While component-based approaches offer significant benefits for complex systems, they require careful attention to interface specifications and composition rules to ensure that component properties are preserved when integrated into larger systems.

D. Model-Driven Engineering Approaches

Model-driven engineering (MDE) approaches leverage metamodels and transformation rules to generate PLC code from Petri net models. These approaches focus on raising the abstraction level from code to models, improving development efficiency and quality.

Key elements of MDE approaches for Petri net to PLC conversion include:

Metamodel-Based Translation: This approach establishes explicit metamodels defining the abstract syntax and constraints for both Petri net formalisms and PLC programming languages. Transformations operate at the metamodel level, defining formal mappings between corresponding elements in both domains. This separation of transformation logic from implementation provides exceptional flexibility for adapting to different Petri net variants or PLC platforms.

Transformation Rules: These formalized specifications define precise mappings between source and target model elements, capturing both structural correspondences and behavioral semantics in machine-executable form. Rule specifications typically employ specialized transformation languages with precise execution semantics. Complex transformations often use rule chaining techniques with intermediate models to bridge the semantic gap between domains.

Traceability Management: This critical capability establishes and preserves explicit links between elements in the source Petri net model and their corresponding implementations in the generated PLC code. These traceability links support impact analysis for model changes, verification of transformation correctness, code debugging, and round-trip engineering.

Table 4 presents a comparison of model-driven engineering approaches for Petri net to PLC translation found in recent literature.

TABLE IV: COMPARISON OF MODEL-DRIVEN ENGINEERING APPROACHES

Approach	Authors	Year	Metamodel Type	Transformation Technology	Traceability Support	Target Languages	Industrial Validation
PetriCode	Simonsen et al.	2015	CPN Metamodel	ATL Transformations	Explicit mapping	Multiple	Limited
MDE Framework	Estévez & Marcos	2012	UML Profile for Petri nets	QVT Transformations	Comprehensive	IEC 61131-3	Yes
AutomationML	Lüder et al.	2013	XML-based metamodel	XSLT Transformations	Partial	Ladder Logic	Yes
PNML Framework	Hillah et al.	2010	PNML Metamodel	Java Transformations	Limited	Multiple	No
PLCOpen XML	Thramboulidis & Frey	2017	IEC 61131-3 Metamodel	Model-to-text	Comprehensive	All IEC 61131-3	Yes

Estévez and Marcos [9] developed comprehensive MDE frameworks specifically for Petri net to PLC transformation, addressing issues of scalability and maintainability. Their approach incorporated multiple abstraction levels bridging the gap between abstract models and concrete implementations through systematic refinement steps with comprehensive validation at each level.

The advantages of MDE approaches include:

- Separation of concerns between domain knowledge and implementation details
- Systematic transformation process with formal foundations
- Enhanced traceability between models and code

- Support for evolution and maintenance through model-level changes
- Platform independence with target-specific code generation

MDE approaches represent a significant advancement in bridging the gap between formal models and industrial implementation, providing a structured methodology that enhances both development efficiency and system quality while maintaining the rigor of formal methods.

IV. Comparative Performance Analysis

Comparing the performance of different implementation approaches across various PLC platforms provides valuable insights into the efficiency and effectiveness of Petri net to PLC conversion. This analysis considers metrics such as translation time, code size, execution efficiency, and memory usage.

Table 5 presents a comparative analysis of the performance characteristics of different conversion approaches based on empirical measurements.

Table V: Performance Comparison of Different Conversion Approaches

Method	Translation Time	Code Size	Execution Efficiency	Memory Usage	Complexity Handling	Debugging Support
Direct Mapping	<1s	Minimal	High	Low	Limited	Minimal
Automata-based	0.6s	Moderate	Medium	Medium	Good	Moderate
XML-based	15.3s	Large	Low	High	Excellent	Excellent
Binary PN	4.5s	Optimized	High	Low	Good	Limited
Component-based	2.8s	Modular	Medium-High	Medium	Excellent	Good

Direct Mapping approaches generally offer the best performance metrics, with translation times under 1 second, minimal code size, high execution efficiency, and low memory usage. This efficiency makes them well-suited for resource-constrained environments, though they may lack some of the advanced features of other approaches.

Automata-based methods achieve rapid translation at approximately 0.6 seconds with moderate code size. Their execution efficiency is medium, suggesting some runtime overhead from the state-based approach, with corresponding medium memory usage. This represents a balanced trade-off between performance and functionality.

XML-based approaches typically require longer translation times (around 15.3 seconds) and produce larger code sizes. Their execution efficiency tends to be lower due to the overhead of processing XML structures, with higher memory usage limiting their applicability in resource-constrained environments.

Binary Petri Net methods take approximately 4.5 seconds for translation but generate optimized code with high execution efficiency and low memory usage. This suggests effective optimization techniques that balance translation time with runtime performance.

Table 6 provides platform-specific performance comparisons, illustrating how different PLC systems handle Petri net-derived code.

Table VI: Platform-Specific Performance Considerations

PLC Platform	Scan Cycle Impact	Memory Usage	Execution Efficiency	Code Readability	Optimization Potential
Siemens	Low	Moderate	High	Moderate	High
Allen Bradley	Low	Moderate	High	Good	Moderate
Mitsubishi	Low-Moderate	Moderate	Medium-High	Moderate	Moderate
Omron	Low	Moderate	High	Good	Moderate
Generic IEC 61131-3	Variable	Moderate-High	Medium	Good	Limited

Siemens PLCs generally demonstrate low scan cycle impact with moderate memory usage across different conversion methods.

The MC7 byte-code architecture benefits from optimization techniques focused on atomic instruction handling, resulting in compact and efficient code.

V. Conclusion

This review has examined the significant advancements, persistent challenges, and future opportunities in translating Petri nets into PLC code for industrial automation applications. The integration of formal modeling techniques with practical implementation technologies represents a promising approach to enhancing the reliability, safety, and efficiency of industrial control systems.

The diverse methodologies for Petri net to PLC conversion—including direct translation, intermediate model-based approaches, component-based methodologies, and model-driven engineering—offer different trade-offs between simplicity, expressiveness, verification capabilities, and implementation efficiency. Each approach has demonstrated value in specific contexts, contributing to a rich landscape of techniques that address various aspects of the conversion challenge.

In conclusion, Petri net to PLC conversion systems represent a valuable contribution to industrial automation, offering a structured approach to developing reliable and efficient control systems based on formal models. By addressing the identified challenges and pursuing the suggested research directions, these systems can play an increasingly important role in enhancing the quality, safety, and performance of industrial automation systems in diverse applications and industries.

The integration of Petri nets with PLCs through these innovative methods not only simplifies the programming process but also ensures more robust and error-free control, paving the way for more advanced and resilient industrial automation solutions.

REFERENCES

[1] Petri, C. A. (1962). "Kommunikation mit Automaten." Ph.D. dissertation, University of Bonn.

[2] Jensen, K. (1981). "Coloured Petri nets and the invariant-method." Theoretical Computer Science, 14(3), 317-336.

[3] Berthomieu, B., & Diaz, M. (1991). "Modeling and verification of time dependent systems using time Petri nets." IEEE Transactions on Software Engineering, 17(3), 259-273.

[4] Zhou, M. C., & Venkatesh, K. (1999). "Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach." World Scientific.

[5] Frey, G., & Litz, L. (2000). "Formal methods in PLC programming." In IEEE International Conference on Systems, Man and Cybernetics, 2431-2436.

[6] Uzam, M., Jones, A. H., & Ajlouni, N. (1996). "Conversion of Petri net controllers for manufacturing systems into ladder logic diagrams." In Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation, 649-655.

[7] Brusey, J. (2006). "PetriLLD: A tool for modeling and generating PLC code." Journal of Manufacturing Systems, 25(2), 115-127.

[8] Adiego, B. F., Darvas, D., Viñuela, E. B., et al. (2015). "Applying model checking to industrial-sized PLC programs." IEEE Transactions on Industrial Informatics, 11(6), 1400-1410.

[9] Estévez, E., & Marcos, M. (2012). "Model-based validation of industrial control systems." IEEE Transactions on Industrial Informatics, 8(2), 302-310.

[10] Thapa, D., Park, C. M., Park, S. C., & Wang, G.-N. (2009). "Auto-generation of IEC standard PLC code using t-MPSG." International Journal of Control, Automation and Systems, 7(2), 165-174.

[11] Gergely, E. I., Coroiu, L., & Gacsadi, A. (2010). "Design of safe PLC programs by using Petri nets and formal methods." Proceedings of the 14th WSEAS International Conference on Systems, 454-459.

[12] Karl-Heinz, J., & Tiegelkamp, M. (2010). "IEC 61131-3: Programming Industrial Automation Systems." Springer.

[13] Vyatkin, V., & Hanisch, H. M. (2009). "A modeling approach for verification of IEC 61499 function blocks using net condition/event systems." In IEEE International Conference on Emerging Technologies and Factory Automation, 261-270.