ISSN: 2831-6789

Research Article

Journal of Robotics and Automation Research

Formalisms for Enterprise Application Integration (EAI): A Survey of Methodologies

Sivasankara Rao Kotha[1*], Gopal T V[2]

[1]*Research Scholar, Department of Computer Science and Engineering, Anna University, College of Engineering Guindy Campus, Chennai, India*

[2]*Professor, Department of Computer Science and Engineering, Anna University, College of Engineering Guindy Campus., Chennai, India*

***Corresponding Author**
Sivasankara Rao Kotha, Research Scholar, Department of Computer Science and Engineering, Anna University, College of Engineering Guindy Campus, Chennai, India

Submitted: 01 Feb 2023; **Accepted**: 06 Mar 2023: **Published**: 15 Mar 2023

Citation: Kotha, S. S. R., Gopal T. V. (2023). Formalisms for Enterprise Application Integration (EAI): A Survey of Methodologies. *J Robot Auto Res, 4*(1), 337-342.

Abstract

Over the years, the number of applications supporting enterprise business processes has increased. The challenge of integrating diverse systems is one of the many reasons why many organizations fail to achieve greater automation. To overcome this obstacle, they are turning to Enterprise Application Integration (EAI). Enterprise Application Integration is a process that enables the integration of different applications. This allows the users to easily modify the functionality, share the information among the various applications and reuse the methods. The paper presents a formal method that includes the various levels of EAI. It highlights the various formal methods that can be used to achieve EAIs seamless interoperation. It also supports the concurrent and dynamic system. This paper also proposes a new architecture for EAI that will help them achieve their goals. There are many formal methods for programming languages in software engineering, but most of them are not adequate for the development of complex systems. The author proposes a new methodology based on Petri net which is a graphical representation of semantics.

Keywords: Enterprise Application Integration (EAI), Formal Methods, Grammar rules, Interoperability, Multi-Agent System (MAS), Petri net, Unified Modeling Language (UML), Verification and Validation, Z Specification Language

Introduction

Application integration is not new in enterprises, but the methods for conducting EAI are presently being understood. The enterprises have many applications that have been developed using heterogeneous environments and platforms. Enterprise Application Integration (EAI) is a methodology to integrate the different applications at the enterprise level. The core functionality of EAI is the ability to create meaningful messages and the ability to guarantee the delivery of these messages to both source and target applications [1].

A point-to-point method was used to exchange the data between the different applications [2]. Later middle ware technologies were used for the same thing to integrate the different applications. Identify some of the most successful technologies in the middleware market and show the impact of their creation on the industry [3]. There are two main models of the middleware technologies in EAI: Hub-Spoke and Message Bus. Hub and Spoke methods are like client and server communication. Hub is acted as a server and spokes are the clients like different applications. In the Message Bus model, the applications utilize the bus model rather than the centralized server model of the hub and spoke model. In addition to the different Models and Architectures, they are some other characteristics of EAI. Some applications are loosely coupled and some are tightly coupled. One more characteristic is based on whether the applications are Synchronous and Asynchronous. Based on these distinguishing characteristics the integration process will perform and exchange the messages and data between applications.

They are some levels of integration in which the integration process will run [4].

- Data Level Integration
- Application Level Integration
- Method Level Integration
- User Interface Level Integration
- Process Level Integration
- Architecture Level Integration

Each level having own methodologies and techniques. Formal proofs and formally verifiable integration processes in the EAI have been a challenge. In the present era, almost all organizations are dependent on new growing technologies and for that, they have been developing many applications. Some applications are developed and designed within few months and some of a few weeks also. These applications are dynamic in nature, distributed in and out of enterprises, and are developed in different infrastructures

and platforms.During the 21st century, the business environment in every organization is tremendously dependent on new technologies and having a great impact on these industries, and it will erase the boundaries of every organization functional-wise, exchanging the data and process between them. The new functionality and business environment approach will appear in the coming years.

The industrial wise integration methodologies are clearly explained in the and summarizes like need to developing new frameworks and methodologies to enable enterprises to integrate their existing applications into new technologies like the Cloud Computing and Internet of Things (IoT) is an area of concern [5]. In reported a large set of patterns that could be used to develop integration solutions, depending on the most adequate type of solution for a particular integration problem [6]. As we discussed above, applications are dynamic in nature and loosely coupled also. Sharing the data and process environment between the loosely coupled applications is very important and carefully integrated. They are many technologies and methods for every level of integrations. Web Services, Service Oriented Architecture (SOA), Enterprise Service Bus (ESB), Application Programming Interface (API), and Extraction Transformation Loading (ETL) so on, are the different methods and technologies that were made to integrate the different applications in levels of Enterprise Application Integration [1, 4].

Whatever technologies are made and methods are developed, the verification and validation of each approach are necessary. Whether the applications are integrated correctly or not, whether the integrated applications are work properly and share their data or not. So formal methods are taken care of the above problem. There is a need for use of formal methods in software engineering process. They are some commandments for applying the formal methods [7]. It is a mathematical or logic based technique to systematically develop, describe, and verify a software system. By formal methods, we can verify and validate the applications before the implementation stage of the software process. They are some myths to use formal methods in software engineering, even it can help to reduce lead times and lower development costs [8, 9].

Because of distributed and dynamic nature of the application, we need the same characteristic of formal method and verifiable formal technology. Petri net is the one, which is distributed in nature, perfectly verifiable to the loosely coupled applications while integrating at an enterprise level. The details of Petri net and how it will be useful for EAI has explained in the next session. Our proposed approach is to develop a Petri net model for different objects and processes of each application while integration occurs. There are many levels of integration in which we have to develop a prototyping model to verify each level of EAI. The layering approach is very useful and helpful to understanding our proposed concept. It is showed us how to build a network. Already we have mentioned some layering models of different levels of integration. The multilevel approach has been good in software systems. Based on the above theory, we have approached different types of formalisms for Enterprise Application Integration (EAI). Several models were proposed in the literature, such as UML, Z Language, Formal Grammar, Multi-Agent System, and Petri Net. As it will be shown in the next section Petri net model has several advantages compared to other models.

The rest of the paper is organized as follows: In section 2 focuses on the di_erent formal approached views to Enterprise Application Integration (EAI). Conclusion and Future Work has given in section 3 and section 4, followed by references.

Different Formal Views

Unified Modeling Language (UML)

UML (Unified Modeling Language), with more notations embodied, is suggested as a general and standard notation for the analysis, design, and development of object-oriented software systems [10]. The semantics of UML is intended for the latter field [11]. It is a very popular formal modeling language (somewhat semi-formal). Most of the time it demands Object Orientation. As a layering approach to a software process, the object-oriented concept is very understandable and useful to create an application. We argue that concepts should be correctly represented, at appropriate levels and that clear semantic links between them should be provided for useful integration. Then the resulting in a more powerful, useful, and flexible system from all points of view.

The modeling power of UML is very high and can be demonstrated by applying it to some systems. The new approaches to address the analysis and design of application systems must be a study and obtain interesting properties of the systems. It is necessary the application of formal methods to the enterprise level to integrate different applications. Each step in the process of software is also very important. Deployment and integration after this very difficult. Maintenance of software is very easy but the maintenance of EAI is very difficult. The execution time of the software maintenance process may get increased due to the integration of different types of applications, thus, increasing the cost and decreasing the performance of the process [12]. In mentioned the standard definition of maintenance like “the totality of activities required to provide cost-effective support to a software system. Activities are performed during the pre-delivery stage as well as the post-delivery stage” [13]. UML diagrams are powerful tools for system design but they are unable to address nonfunctional parameters. This means UML diagrams cannot be used for performance evaluation. By using UML, we can describe the user requirements, static and dynamic properties, and behavior of a system in a convenient way. Easy to transform the UML to the source code of the program. It is difficult to analyze the UML model since it is an informal language [14].

Z Specification Language

As application integration wise, fault tolerance is the desirable feature for every integration tool. So that, EAI solutions can keep running despite the occurrence of failure. Errors monitoring is the main activity in fault tolerance since it enables the detection of errors. Rule-based language is one of the solutions to provide an error detection mechanism to detect the errors in the system based

on the monitoring system [15]. Z is one of the _nest rule-based and formal specification languages, in that some few tools are supported to monitor and detect semantic errors. That is the reason we take the option to choose the Z formal specification language in our proposed EAI system. Z is a formal specification language based on set theory and first-order logic [16, 17]. It is a formal notation for specifying the functionalities of sequential systems, it does not support the concurrent and distributed system. It includes a set of entities, called schema which are representing of an abstract class of system and its operations. Every entity and its related operations expressed through a rich set of mathematical notations.

Z specification language offers a rich type of definition facility and it supports formal reasoning of the system. However, it does not support concurrent and distributed systems and does not have explicit operational semantics [18]. It has been used as a specification language to formally describe and analyze the requirements and the design architectures of a wide range of hardware and software system. Despite some advantages, specification languages having some weaknesses also. They are like limited scope of properties, limited tool support, isolation, poor guidance, and cost, and so on. By Z specification languages we can specify the state space and sets of operations very clearly but we cannot express the combination of the operations. In enterprises, we also need to be able to talk about the behavioral aspects of the system.

It is a very successful formal specification language. Z specification language represents the different objects, events, and processes, the relation between them to integrate the applications. We found that it is suitable mostly for specific domains not all. It is good enough about some verticals, nuclear systems, and some specific domains. Z specifications do not have explicit operational semantics and it does not support an effective definition of distributed and concurrent systems. EAI need not be tied up to any domain.

Formal Grammar

The rule-based concept is also considered in this scenario. The rules are written by some grammars and automatically which is decided the choice of the particular level of integration. The formal way of the rule-based concept is the approach in which anyone can write their own rules which will suit their problem. Commonly Context-Free Grammar (CFG) and Context-Sensitive Grammar (CSG) are the grammars that will use for writing the rules. These grammars are run mainly based on the context. Complete ordering of events is very difficult to match the hierarchy or levels. This is the reason we went to process view of a model. Finite State Machine (Finite State Automaton) is a notable formalism in the automata theory to represent all the states and the transitions between its states [19]. This formal method generally less powerful in complex and concurrent systems.

Pi-Calculus is a process algebra and mathematical formalism for describing and analyzing properties of concurrent computation and the process interaction by sending communication links to each other. EAI happens ad hoc in nature [20, 21]. There is no firm plan of applications emerging. So the emerging scenario is difficult in Pi- Calculus. If you can make a flexible process that is a very high level. The optimizing process is what is necessary for the approach. The optimizing process definition in EAI is not fitting well in Pi-Calculus. Where there is the strong process, where there is the strong adherence. It is capable of a process definition. But we cannot say it is optimized. It is good in process definition and protocol design. It is good in the level crossed model also. But, EAI is not dependent on a single level of integration, needed a higher level of maturity for a strong process definition. It is not good for ad hoc.

Multi-Agent System (MAS)

Generally, the Multi Agents approach is designed for open systems. Autonomy, Heterogeneity, and Dynamics are the main characteristics of open systems. Agents system is characterized by modularity, abstraction, dynamism, and interoperability. This is the main reason that the agents approach was considered for application integration in the dynamic and open system environment [22]. The agent is a system that is situated in some environments, and it acts autonomously to satisfy the design objectives. Autonomous and Environment are the two important parameters in agent technology. The agents are autonomous that they can act according to their will. They understand what to do based on the environment. The agent's action will influence the environment. In most cases, the agent's actions only have partial control over its environment. These actions which are taken based on the environment are not influenced by a human, or any other agents [23]. Agents are used agent communication languages such as KQML and FIPAs ACL to exchange messages [24, 25]. The main aim of agent communication languages is to provide precise semantics and syntax for interaction between agents.

It is a highly reputed approach at an enterprise level. Each agent has to connect with the environment to do the communication between agents. It is environment-based communication. Most complexity in this approach is agent interaction with the environment. The environment or ecosystem is the best applicable for information retrieval, search engines because of the cyclic approach. There are many approaches to integrate the applications by the multi-agent scenario such that, use each agent as a wrapper of applications, construct a multi-agent architecture in which each agent is interacting with other agents and provide an integration solution, and consider each agent as an intelligent manager of an open environment [26, 27]. EAI having many applications which are developed in a different environment. The complexity of the agent depends on the number of agent interactions with the environment. So, it is difficult to maintain and communicate between many environments at a time in a multi-agent system.

Petri Net

Petri Nets is one of the formal specification and graphical oriented modelling languages for the design, specification, and verification of distributed systems [28]. By using Petri net, we can analyze the dynamic properties and structure of systems through strict

mathematics analysis and visualized computer simulation as well as model distributed and parallel processes [29]. In find that Petri nets are an attractive alternative of above mentioned models due to their extensive capability to perform analytics and simulation [30]. In represents the simulation of integration solution using conceptual models and concluded like the scope of EAIs investigation is still vast, it is hoped that in the future, new tools and methods will be developed to support this area of study [31]. Unified Modeling Language (UML) is also a powerful modeling language having many notations and design diagrams. Petri net has a graphical representation and well-defined semantics, which allow compact, manageable representation, and more powerful analysis than the UML. Some of the other features of UML and PNs are mentioned here [14],

- Petri Nets possess formal strictness Than the UML.
- Petri Nets model is suitable for simulation while the UML model can be implemented easily.
- Petri Nets can analyse systems strictly whereas UML can describe systems effectively.
- Petri nets can be divided based on modeling power, mechanisms for data abstraction, and refinement [29]. They are different types of Petri nets such as,
- High-Level Petri Net [32]
- Coloured Petri Net [33-35]
- Relation Transition Net [36]
- Algebraic Petri Nets [37]
- Timed Petri Nets [38]
- Stochastic Petri Nets [39, 40]

The Petri net definition and syntax being to be changed based on the requirement and type of nets which we could use. The formal definition of Petri net is,

A Petri net having 5-tuple,

PN = (P, T, F, W, M0)

P = {p1, p2, .., p_m} is a finite set of places denoted by circles

T = {t_1, t_2, ..., t_n} is a finite set of transitions denoted by rectangles

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs denoted by lines

W : f → (1, 2, 3, ...) is a weight function,

M0 : P → (0, 1, 2, 3, ...) is the initial marking like tokens denoted by bolded dots.

$P \cup T \neq \emptyset$ and $P \cap T = \emptyset$

The dynamic behavior of the system can change depends on the changes of the places (P) and transitions (T) in the Petri net.

In this field, some basic questions will arise,

- Can we reach one particular state from another?
- Will a storage place overflow?
- Will the system die in a particular state?

Stepwise elaboration of above questions on Petri net:

Step 1: Design the model of the system based on the requirement

Design the net model by using Places, Transitions, and Arcs which acts as a communicator between places and transitions.

Step 2: Analysis of the properties

Reachability: It is a fundamental basis for studying the dynamic properties of any system. It works based on the firing of tokens as an enabled transition, it will change the total net according to the transition rule.

Boundedness: A Petri net is said to be simply bounded if the number of tokens has fired from a place which is not exceeding the finite number. If it k-bounded, means it does not exceed the k token values.

Liveness: The liveness property is like a deadlock property of the system.

Besides, Petri nets provide various analysis techniques such as,

- Reachability Tree,
- Incidence Matrix,
- Invariant Analysis Method.

Through these analysis techniques, the properties of the Petri nets models such as Reachability, Liveness, and Boundedness can be examined. By using Petri nets we can analyze the structure and dynamic properties of systems through strict mathematics analysis and visualized computer simulation as well as model distributed and parallel processes. As we said earlier Enterprise Application Integration (EAI) integrates methods, objects, and tools for the classification, coordination, connection of applications within organizations. The main goal of EAI is to integrate a business processing of applications of different generations and architecture. These applications consistently change through upgrades or adding of new applications with modified technologies and other influences. One of the prerequisites for reaching this goal is the documentation of business processes of the individual applications and their interfaces should be unified. Table 1 represents the comparisons of our approached formal views with different characteristics of EAI. UML having more complex because of having more diagrams, level of the hierarchy is more and most important is In-formal characteristic, even though having more reliability, and effective communication property with heavy industrial usage.

The body of schema in Z specification language may refer to items that are not declared directly in the schema. Generally, Z specifications look clumsy when it is used to specify large systems. It only specifies the functionalities of the system, not to communicate and handling the entire system. Having more complexity, difficulty to maintenance, somewhat in-formal so on are some limitations to the approach of this specification language.

Table 1: Comparison of Different Formal Views.

Different Models/ Properties	UML	Z Specification Language	Formal Grammar/ Automata	Multi-Agent System	Petri Net
Reliability	More	More	More	Less	More
Dynamicity	More	Less	Less	More	More
Distribution	Possible	Possible	Less	Possible	High Possible
Complexity	High	High	More	High	Less
Maintainability	Difficult	Difficult	Not an Easy	Difficult	Easy
Concurrency	possible	Not supported	Less	More	More
Hierarchy	possible	Possible	Possible	Possible	Less
Process Collaboration	Possible	Possible	Possible	Possible	Possible
Strong Formalism	In-formal	Not fully	Possible	Less	High Formalism
Communication	Effective	Effective	Less	Effective	More Effective
Industrial Practice	More	Less	Less	Less	More
Tool Support	Possible	Limited	Limited	Possible	Possible
Reusability	Possible	Possible	Possible	Less Possible	Possible

The automata and grammar theory also having less dynamicity and somewhat informal, limited tool support.

Same as the above models, the multi-agent system also had some limitations to continue as an approached model of EAI. Process collaboration of the entire hierarchy system is somewhat difficult, its maintenance and complexity are high, not much tool support, less formalism, and difficult to communicate different environments to different agent applications in large systems.

By the all models, finally we have approached Petri Net is a most sophisticated and useful formal model for EAI. Petri net has a graphical representation and well-defined semantics, which allow compact and manageable representation and more powerful analysis. The tool set has been developed to automate Petri net analysis, which examines behavioral properties of Petri net such as deadlocks, conflicts, blocking, and performance parameters ranging from throughput rate, utilization to expected buffer size so on. Moreover, concurrency can be modeled, allowing action to take place simultaneously, and it allows interactive simulation also. So, user can easily identify throughout the model to locate a bottleneck and to troubleshoot the problem.

The author has observed that it has some primary advantages of Petri Net:

• The graphical model uses very few but powerful primitives making it easy to understand.

• Models can be represented as tuples, which the computer can interpret and analyze.

It can unambiguously describe a system, showing explicitly both states and actions, whereas other formal methods focus on either states or actions but not both. This allows users to change between the two perspectives as desired.

Conclusion

In this paper, we presented an introduction of Enterprise Application Integration (EAI) and Techniques, methodologies, and some integration levels of EAI. This paper is a review of some widely used formal methods for EAI. A quick summary of formal methods like Unified Modeling Language (UML), Z specification Language, Context-Free Grammar (CFG), Pi- Calculus, Multi- Agent System (MAS), and implementation for EAI accordingly has been explained. A summarized comparison report of selected formal method views has been given. Finally, we have chosen a Petri net model for our problem at hand. A brief explanation of the Petri net model and how it will use for our problem has been given. In future work, the authors implement a Petri net model for analyzing the properties like Reachability, Boundedness, and Liveness using techniques like Reachability Tree, Incidence Matrix, Invariant Analysis for the Enterprise Application Integration.

References

1. Linthicum, D. S. (2000). Enterprise application integration. Addison-Wesley Professional.
2. Brown, W. J., Ruh, W. A., & Maginnis, F. X. (2002). Enterprise application integration: a Wiley tech brief. John Wiley & Sons.
3. Emmerich, W., Aoyama, M., & Sventek, J. (2008). The impact of research on the development of middleware technology. ACM Transactions on Software Engineering and Methodology (TOSEM), 17(4), 1-48.
4. Kotha, S., & Gopal, T. V. (2021). Formal methods for enterprise application integration. Complex Control System, 3(1), 9-24.
5. Gorkhali, A., & Xu, L. D. (2016). Enterprise application integration in industrial integration: a literature review. Journal of Industrial Integration and Management, 1(04), 1650014.
6. Hohpe, G., & Woolf, B. (2004). Enterprise integration pat-

terns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional.
7. Bowen, J. P., & Hinchey, M. G. (2005). Ten commandments ten years on: An assessment of formal methods usage. SEEFM, 18-19.
8. Hall, A. (1990). Seven myths of formal methods. IEEE software, 7(5), 11-19.
9. Bowen, J. P., & Hinchey, M. G. (1995). Seven more myths of formal methods. IEEE software, 12(4), 34-41.
10. Fowler, M. (2004). UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional.
11. Garrido, J. L., & Gea, M. (2002). A coloured petri net formalisation for a UML-based notation applied to cooperative system modelling. In Interactive Systems: Design, Specification, and Verification: 9th International Workshop, DSV-IS 2002 Rostock, Germany, June 12–14, 2002 Revised Papers 9 (pp. 16-28). Springer Berlin Heidelberg.
12. Nabeel, M., Anwar, Z., & Ahsan, A. (2018). Global Journal of Computer Sciences: Theory and Research. Global Journal of Computer Sciences: Theory and Research, 8(1), 01-13.
13. Software Engineering Standards Committee et al. Ieee standard for software maintenance. IEEE Std, 1998.
14. Hongmei, G., Biqing, H., & Shouju, R. (2000). A UML and Petri Nets Integrated Modeling Method for Business Processes in Virtual Enterprises. In American Association for Artificial Intelligence.
15. Klein, M. J., Sawicki, S., Roos-Frantz, F., & Frantz, R. Z. (2014, April). On the Formalisation of an Application Integration Language Using Z Notation. In ICEIS (1) (pp. 314-319).
16. Ter Bekke, Johan H., and J. H. Ter Bekke. Semantic data modeling. Hemel Hempstead: Prentice Hall, 1992.
17. Diller, A. (1994). Z: An introduction to formal methods. John Wiley & Sons, Inc..
18. He, X. (2001). PZ nets—a formal method integrating Petri nets with Z. Information and Software Technology, 43(1), 1-18.
19. Sakarovitch, J. (2009). Elements of automata theory. Cambridge University Press.
20. Parrow, J. (2001). An Introducion to the Pi-calculus. Handbook of Process Algebra, Bergstra and Ponse and Smolka.
21. Padua, D. (Ed.). (2011). Encyclopedia of parallel computing. Springer Science & Business Media.
22. Yoo, M. (2007). Enterprise Application Integration from the Point of View of Agent Paradigm. In Enterprise Architecture and Integration: Methods, Implementation and Technologies (pp. 225-238). IGI Global.
23. Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. The knowledge engineering review, 10(2), 115-152.
24. Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994, November). KQML as an agent communication language. In Proceedings of the third international conference on Information and knowledge management (pp. 456-463).
25. FIPA, A. (2004). Message structure specification. foundation for intelligent physical agents, 2000. URL: http://www. fipa. org/specs/fipa00061, Abruf am, 09-29.
26. Wang, G., Zheng, J., Wu, H., & Tang, Y. (2009, August). Research of the Enterprise Application Integration Platform Based on Multi-agent. In 2009 Fifth International Joint Conference on INC, IMS and IDC (pp. 329-331). IEEE.
27. Benmerzoug, D. (2013). Agent approach in support of enterprise application integration. arXiv preprint arXiv:1311.6149.
28. Carl, A. (1962). Petri. kommunikation mit automaten. PhD, University of Bonn, West Germany.
29. Murata, T. (1989). Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4), 541-580.
30. Kim, R., Gangolly, J., & Elsas, P. (2017). A framework for analytics and simulation of accounting information systems: A Petri net modeling primer. International Journal of Accounting Information Systems, 27, 30-54.
31. Belusso, C. L., Sawicki, S., Roos-Frantz, F., & Frantz, R. Z. (2016). A study of Petri Nets, Markov chains and queueing theory as mathematical modelling languages aiming at the simulation of enterprise application integration solutions: a first step. Procedia Computer Science, 100, 229-236.
32. He, X., & Murata, T. (2005). High-level Petri nets—extensions, analysis, and applications. In The Electrical Engineering Handbook (pp. 459-475). Academic Press.
33. Jensen, K. (1987). Coloured petri nets. In Petri nets: central models and their properties (pp. 248-299). Springer, Berlin, Heidelberg.
34. Jensen, K. (1996). Coloured Petri nets: basic concepts, analysis methods and practical use (Vol. 1). Springer Science & Business Media.
35. Wil Van Der Aalst, M. P., & Stahl, C. (2011). Modeling business processes: a petri net-oriented approach. MIT press.
36. Reisig, W. (1985). Petri nets with individual tokens. Theoretical Computer Science, 41, 185-213.
37. Reisig, W. (1991). Petri nets and algebraic specifications. High-level Petri Nets: Theory and Application, 137-170.
38. WANG, J., & JIANG, S. (2012). and Flexible Systems. Petri Nets in Flexible and Agile Automation, 310, 207.
39. Haas, P. J. (2006). Stochastic petri nets: Modelling, stability, simulation. Springer Science & Business Media.
40. Marsan, M. A., Balbo, G., Conte, G., Donatelli, S., & Franceschinis, G. (1998). Modelling with generalized stochastic Petri nets. ACM SIGMETRICS performance evaluation review, 26(2), 2.