Computing Science | Review Article

Smart selection from petri net modeling tools for fast developing a manufacturing system

Yi-Nan Lin¹, Cheng-Ying Yang², Gwo-Jen Chiou³, Sheng-Kuan Wang⁴, Victor R.L. Shen⁵*, Yu-Ying Wang⁶, Hai Hoang Bui⁷ and Jianzhi Wang⁸

Abstract: Petri net (PN) is a well-known graphical and mathematical model that can be used to describe a system quickly and dynamically. Over the last decade, it has provided many applications for academia research and industrial practice, especially, for the system verification and performance evaluation. Compared with other system models, the PN model is easy to implement. Therefore, it can be recommended as an effective way for people to study the model checking and validation. For the purposes of reducing time span to get familiar with PN tools, it is necessary to have an informative guide, providing general information about a PN software tool for system developers. This paper aims to review HiPS, PIPE, and WoPeD tools which are simple, lightweight, and free of charge. Our proposed approach also compares their strengths and limitations with each other to help the system developers determine a suitable PN software tool and shorten the time to markets for their products. Thus, it is truly beneficial to fast develop a manufacturing system.

Subjects: Automation Control; Petri Net Modeling and Analysis; Concurrent Computing; Systems Engineering; Software Engineering & Systems Development; CAD CAE CAM - Computing & Information Technology; Industrial Engineering & Manufacturing; Manufacturing Engineering; Production Engineering; Systems & Control Engineering;

Keywords: System Development; System Modeling; Petri Net Software Tool; System Verification; Performance Evaluation; Simulation

About the Author

Yi-Nan Lin received his B.S. degree in Electrical Engineering from National Taiwan Institute of Technology, Taipei City, in 1989; M.S. degree in Computer Science & Engineering from Yuan Ze University, Tao-Yuan City, Taiwan, in 2000; and Ph.D. degree in Electrical Engineering from Chang Gung University, Tao-Yuan City, Taiwan, in 2009. Since 1990, he has joined the Department of Electrical Engineering at Ming Chi University of Technology (MCUT), Taishan District, New Taipei City, Taiwan. He is now serving as Professor at the Department of Electronic Engineering in MCUT. His current research interests include Petri net theory and applications, systems engineering, error-control coding, digital transmission systems, and embedded system design.

Public Interest Statement

Petri nets are a graphical and mathematical model widely applied in the field of systems engineering. It is a promising tool for describing information systems that are characteristic of concurrency, parallelism, distribution, nondeterminism, probability, time, fuzziness, and so on. Due to a graphical model, Petri nets can be used to act as a visualization aid similar to flowcharts. However, the flowcharts do not have all above-mentioned characteristics. In this article, the manufacturing processes are modelled by using Petri net tools for the purpose of optimizing and reducing manufacturing costs. Meanwhile, three Petri net simulation software tools, namely, HiPS, PIPE, and WoPeD, are open sources, which are free of charge and easy to use. Thus, it is the main reason for modeling and analyzing the smart manufacturing systems by using Petri nets from the view-points of practical applications.
1. Introduction

A skillful system developer always knows that the most important quality is to functionally keep correct and to save some resources. Thus, the workflow processes can run smoothly. It is a big risk for manufacturing companies to just put a software/hardware system into practice without any verification procedure (Kalistratov, 2019). The most effective solution to this issue is model simulation. Model simulation allows us to obtain safer systems and perform program testing and to minimize the cost and extra time span (Consuelo, 2020). Therefore, it can help us protect companies’ resources and business. Additionally, developing correct systems is also a vital task as it helps us enhance the performance and eliminate the bugs inside, which can be performed well through model simulation (Bakri et al., 2021). Obviously, the simulation tools have made their great contributions to software engineering or systems engineering.

Currently, some simulation tools use Petri net (PN) theory as a basis for the discrete event systems. PN is a graphical and mathematical model, which offers several characteristics of a system, including concurrent, asynchronous, distributed, parallel, nondeterministic, stochastic features, and so on (Perez-Palacin et al., 2019). It can be considered as a great modeling technique, which was developed by Dr. Carl Adam Petri in 1962 (Brauer & Reisig, 2009). Since then, a variety of systems have been developed based on PN theory (Souravlas and Roumeliotis, 2015; Chakraborty, 2019). It allows the system developers to simulate their systems for analysis and verification. Thus, they can deal with the behavioral and structural properties of the PN model. Moreover, Petri nets have been extended to have colored, timed, fuzzy, and hierarchical features (Hou et al., 2014; Lv et al., 2013). This leads to the establishment and development of various PN simulators to serve as the uprising demands for the integrated functions. However, the downside of this trend is that PN tool users might find it hard to select a suitable one.

To solve the above-stated problem, this paper aims to survey three PN software tools in common use, namely, HiPS, PIPE, and WoPeD, which are graphical PN simulators designed to be friendly and easy to use (Pla, et al., Jun. 2014). Furthermore, they are all distributed online without any charge. These benefits make the trio become recommendable to any researcher or system developer intending to use these simulators. However, each one has its own advantages and disadvantages, as well as different capabilities of the specific use case.

Although there have already been some works comparing different PN tools, for example, the works by references (Osman & Harrison, 2019; Thong & Ameedeen, 2014), none of them discussed the differences among HiPS, PIPE, and WoPeD. Our main goal is to clarify their unknown properties, and to help users get a clear idea about their strengths. For these reasons, we not only compare three tools based on their specialties and features available, but also conduct practical experiments to present their behavioral properties of one PN model. One chosen example is 1-wire search algorithm (Maxim Integrated Inc, 2018), which has been applied in a wide range of products manufactured by Maxim Integrated Corporation, so that the system developer can fast determine the PN tool of interest.

The remainder of this paper is organized as follows: Section 2 presents the definitions of important terms, the PN tools, and related work. Our proposed approach is illustrated in Section 3. Section 4 provides the main results with five comparison aspects. Finally, the remarkable conclusion is given in Section 5.

2. Literature review

This Section first presents the definitions of PN theory, including its basic properties and analysis methods. Then three PN tools regarding HiPS, PIPE, and WoPeD are briefly presented.
2.1. Petri net theory
A basic PN is a bipartite graph with two elements, namely, place and transition that are connected by a directional arc (Shen et al., 2018).

Definition 1: A Petri net is a 5-tuple, \( PN = (P, T, F, W, M_0) \), where
\[\begin{align*}
(1) & \; P = \{p_1, p_2, \ldots, p_n\} \text{ is a finite set of places;} \\
(2) & \; T = \{t_1, t_2, \ldots, t_m\} \text{ is a finite set of transitions;} \\
(3) & \; F \subseteq (P \times T) \cup (T \times P) \text{ is a set of arcs (i.e. flow relation);} \\
(4) & \; W: F \to \{1, 2, 3, \ldots\} \text{ is a weight function;} \\
(5) & \; M_0 = \{M(p_1), M(p_2), \ldots, M(p_n)\}: P \to \{1, 2, 3, \ldots\} \text{ is an initial marking.} \\
(6) & \; P \nsubseteq T = \emptyset \text{ and } P \nsubseteq T = \emptyset.
\end{align*}\]

A Petri net structure \( PN = (P, T, F, W) \) without any specific initial marking is denoted by \( PN \). A Petri net with the given initial marking is denoted by \( (PN, M_0) \). A Petri net is ordinary if all arc weights are 1s. A state machine is a Petri net in which every transition has exactly one incoming arc and one outgoing arc. A transition \( t \) is said to be enabled if each input place \( p \) of \( t \) is marked with at least \( w(p,t) \) tokens, where \( w(p,t) \) denotes the weight of an arc from \( p \) to \( t \). An enabled transition may or may not fire, depending on whether the event occurs or not.

Definition 2: Behavioral properties of \( PN \) are referred to as initial marking-dependent properties, including the following properties (Shen et al., 2018):
\[\begin{align*}
(1) & \; \text{Reachability: } M \in R(M_0), M = A^T x + M_0, \text{ where } A \text{ denotes an incidence matrix; } M_0 \text{ denotes initial marking; } M \text{ denotes final marking; } x \text{ denotes a transition firing vector.} \\
(2) & \; \text{Boundedness: } M(p) < \infty, \forall p \in P, \forall M \in R(M_0) \text{ (or safeness: 1-boundedness).} \\
(3) & \; \text{Liveness: It has different levels of liveness, namely, } L_0, L_1, L_2, L_3, L_4. \\
\text{L}_0\text{-live (i.e. dead): if } t \text{ can never be fired in any firing sequence } \sigma. \\
\text{L}_1\text{-live (i.e. potentially fireable): if } t \text{ can be fired at least once in some transition firing sequence } \sigma. \\
\text{L}_2\text{-live: if, given any positive integer } k, t \text{ can be fired at least } k \text{ times in some transition firing sequence } \sigma. \\
\text{L}_3\text{-live: if } t \text{ can be fired infinitely in some transition firing sequence } \sigma. \\
\text{L}_4\text{-live: if } t \text{ is } L_1\text{-live for every marking } M \text{ in } R(M_0). \\
(4) & \; \text{Reversibility: } \forall M \in R(M_0), M_0 \in R(M). \\
(5) & \; \text{Coverability: } \forall M_1 \in R(M_0), M_1(p) \geq M(p), \forall p \in P. \\
(6) & \; \text{Persistence: A PN is said to be persistent if, for any two enabled transitions, the firing of one transition disables the other.}
\end{align*}\]
(7) **Synchronic Distance**: It means a metric closely related to a degree of mutual dependence between two events, i.e. synchronic distance between two transitions \( t_j \) and \( t_k \), denoted as \( d_{jk} = \max(\sigma(t_j) - \sigma(t_k)) \). 

**Definition 3**: Structural properties of PN are referred to as initial marking-independent properties, including the following properties (Shen et al., 2018):

1. **Structural Liveness**: A PN is said to be structurally live if there exists a live initial marking in PN.

2. **Controllability**: If a Petri net PN with \( m \) places is completely controllable, then \( \text{Rank } A = m \).

3. **Structural Boundedness**: A PN is structurally bounded if and only if there exists an \( m \)-vector \( y \) of positive integers such that \( Ay \leq 0 \), where \( y \) denotes a place vector.

4. **Conservativeness**: A PN is (partially) conservative if and only if there exists an \( m \)-vector \( y \) of positive (nonnegative) integers such that \( Ay = 0, y \neq 0 \).

5. **Repetitiveness**: A PN is (partially) repetitive if and only if there exists an \( n \)-vector \( x \) of positive (nonnegative) integers such that \( A^T x = 0, x \neq 0 \).

6. **Consistency**: A PN is (partially) consistent if and only if there exists an \( n \)-vector \( x \) of positive (nonnegative) integers such that \( A^T x = 0, x \neq 0 \).

**Definition 4**: The analysis methods of Petri nets are classified into the following two groups:

1. **Coverability Tree**: Given a Petri net \( (PN, M_0) \), from the initial marking \( M_0 \), we can obtain as many “new” markings as the number of enabled transitions. From each new marking, we can again reach more markings. This process results in a tree-presentation of the markings.

2. **Incidence Matrix and State Equation**:

   **Incidence Matrix**: For a PN with \( n \) transitions and \( m \) places, the incidence matrix \( A = [a_{ij}] \) is an \( n \times m \) matrix of integers and its typically entry is given by \( a_{ij} = a_{ij}^+ - a_{ij}^- \) where \( a_{ij}^+ = w(i,j) \) is the weight of the arc from transition \( i \) to its output place \( j \) and where \( a_{ij}^- = w(i,j) \) is the weight of the arc from its input place \( i \) to transition \( j \).

   **State Equation**: A marking \( M_k \) can be written as an \( m \times 1 \) column vector, \( M_k = M_{k-1} + A^T u_k \), where \( k = 1, 2, \ldots \) and \( u_k \) is an \( n \times 1 \) column vector of \( k \)th transition firing.

   Suppose that a destination marking is reachable from \( M_0 \) through a transition firing sequence \( \{u_1, u_2, \ldots, u_d\} \). Writing the state equation for \( i = 1, 2, \ldots, d \) and summing them up, we obtain \( M_d = M_0 + A^T \sum_{k=1}^{d} u_k \) which can be rewritten as \( A^T x = \Delta M \) where \( \Delta M = M_d - M_0 \), and \( x = \sum_{k=1}^{d} u_k \), called the firing count vector.

   **\( T \)-invariant** (Transition invariant): It is an integer solution \( x \) of the transposed homogeneous equation \( \Delta M = 0 \) is \( A^T x = 0 \).

   **\( S \)-invariant** (Place invariant): It is an integer solution \( y \) of the homogeneous equation \( \Delta M = 0 \) is \( Ay = 0 \).
2.2. HiPS, PIPE, and WoPeD

Hierarchical Petri net Simulator (HiPS) is a Petri net tool implemented in C# and C++ on the .NET framework. The tool has been designed to advance operational ability and to improve the execution speed by applying multi-threading into the system. HiPS has an intuitive Graphical User Interface (GUI) which supports hierarchical models and can be used to analyze the behavioral and structural properties of a Petri net by generating the state spaces through multi-threading (Harie et al., 2017). The work by Kavi et al. introduced C2 Petri, which employs multi-threading to create Petri nets simulations (Zhang, 2016). However, there remain limited capabilities for now.

Platform Independent Petri net Editor (PIPE) is an open source and platform-independent software system for modelling and analyzing Petri nets. This tool is implemented in Java to achieve the independence from different platforms. The designers also created a good-looking and handy GUI that allows users to handle Petri nets with the XML/PNML interchange format. Ultimately, PIPE offers a full suite of analysis modules which allows users to check the behavioral properties, to perform data statistics, and to conduct PN comparison and classification (Bonet et al., 2017).

Workflow Petri net Designer (WoPeD) is also a Java-based software tool like PIPE for PN modeling and analysis. In the WoPeD editor, users can create, modify, and export PNs which are like a workflow; and create hierarchical sub-processes to deal with huge system models. Moreover, it also allows users to modify the model’s resources with parameters like objects, roles, and groups on a graphical editor. Finally, WoPeD supports the conversion function between two common models (Freytag & Sänger, 2014).

2.3. Related work

Z. Li, M. Zhou, and N. Wu proposed a deadlock-free scheduling algorithm for the automated manufacturing systems, combining deadlock control strategies with hybrid heuristic search (Ekmecki et al., 2016; Li et al., 2019). The deadlock control strategy prevents the siphon from being empty by installing a central place and associated arcs to each of the minimal siphons. The hybrid heuristic search is based on the state space exploration and therefore it encounters the problem of state space explosion. To reduce the state space explosion, the search space is limited in the local search window. They developed a deadlock-free heuristic search algorithm by embedding the deadlock avoidance strategy in the search processes.

G. Mejia, J. P. Caballero-Villalobos, and C. Montaya proposed a concept of siphon based on PN theory to avoid deadlocks from the flexible manufacturing system (FMS) (Dwyer et al., 2020). They proposed a controllable siphon-based algorithm and established a deadlock prevention strategy.

3. Proposed approach

In this Section, our proposed approach aims to compare HiPS, PIPE, and WoPeD in detail. Three tools are discussed in several aspects required for performance evaluation, and we can make a smart selection from the three PN tools.

3.1. PN software installation

First, we are required to activate the system programs. All of them are freely distributed on sourceforge.net, i.e. a web-based service that provides software system developers with a centralized online location to control and manage free and opensource software projects. The tested versions are HiPS1.1.12, PIPE4.3.0 and WoPeD3.2.0. For HiPS and WoPeD, it is necessary to run the setup files to install the software systems on the machine. However, PIPE is an instant program as we only need to run the.bat launching file to open the editor. Then, the general information of three tools from their self-introduction utilities regarding the development characteristics, compatible environment, and the types of Petri net supported, is all included.
3.2. Feature comparison
It is important that a software interface and its simulation results are simple and easy to understand. From the interfaces of PN tools, their integrated features are placed for comparison, divided into two categories, namely, model editing with simulation features and model analysis features.

For the editing with simulation features, Table 1 describes the basic tools required for PN simulators and the one which is supported by the software systems among HiPS, PIPE, and WoPeD. The features which are available to users in the tested versions are all marked as “X”, while a blank box denotes the feature unsupported. In total, nearly 20 features are used to verify their presence in the software, as a demonstration of the quality of the user interface. They are the file operators, workspace editors, net drawing tools, and simulation options.

Tables 2 and 3 are constructed for the analysis modules available, as they are essential for model checking and verification. As we aim to compare the performance of the simulators, the analysis tools investigated are only related to the behavioral and structural properties, together with two analysis methods. The detailed description of these properties and methods has been presented in Section 2.1. The last row in Table 2 indicates whether the tools possess enhanced analysis features or not, like net classification. For the basic feature comparison, we also use X marks to denote the modules covered by the observed version of each software tool.

3.3. Performance comparison by case study
To give the readers a visualized experience in recommending PN tools, one common algorithm has been employed for constructing PN models, which are further used for performance comparison. Formally and easily conducting experiments is favorable to the demonstration for PN tool users. Hence, we promote this algorithm to result in ordinary state machines after transformation. The advantages of an ordinary state machine are that they are easy to manage and the total number of tokens remains unchanged in the entire processes.

The 1-wire search algorithm (Maxim Integrated Inc, 2018) was defined and provided by Maxim Integrated corporation for application in their 1-wire device family, i.e., DS2483, DS18B20, and DS1482. 1-wire is a device communication protocol designed by Dallas Semiconductor Corporation, defined as multiple devices which can share the same bus on one line for transmitting and receiving signals. Thus, each device on the bus must be assigned a unique address. Thus, each Maxim’s 1-wire device has a 64-bit unique registration number in read-only-memory (ROM) which is used to navigate them by a 1-wire master in a 1-wire network. If the number of the device’s ROMs is unknown, then they can be found by using the 1-wire search algorithm.

| Table 1. General information about HiPS, PIPE, and WoPeD |
|---------------------------------|-------------|-------------|-------------|
| Information                                    | HiPS        | PIPE        | WoPeD        |
| Licensed years                                 | 2012–2017   | 2003–2011   | 2003–2014    |
| Developing language                           | C# and C++  | Java        | Java         |
| Operating system                              | Windows 10/8.1/8/7/ Vista(x64) with.NET Framework 4 | Windows, Linux, Mac | Windows, Linux, Mac |
| Types of Petri Net supported                   | Basic, timed | Basic, colored, timed, stochastic | Basic, colored, timed, stochastic |
Table 2. Comparison of editing and simulating tools among HiPS, PIPE, and WoPeD

| Features                                | HiPS  | PIPE  | WoPeD |
|-----------------------------------------|-------|-------|-------|
| File features (new, open, save, save as, close) | x     | x     | x     |
| Editing (undo, redo, copy, paste)       | x     | x     | x     |
| Snapshot                                | x     |       |       |
| Print                                   |       | x     | x     |
| Export                                  |       |       | x     |
| Zoom                                    | x     | x     | x     |
| Grid regulation                         |       | x     | x     |
| Group and ungroup                       | x     |       | x     |
| Drag model                              | x     | x     | x     |
| Flip model                              | x     | x     | x     |
| Automatical net optimization            |       |       | x     |
| Add colored token                       | x     |       | x     |
| Place capacity restriction               | x     | x     |       |
| Draw enhanced transition                |       | x     | x     |
| Arc weight modification                  | x     |       | x     |
| Add sub page                            | x     |       | x     |
| Manual simulation                       | x     |       | x     |
| Automatical simulation                  | x     |       | x     |
| Enhanced editing tools                  | x     |       | x     |

The 1-wire search algorithm considered a binary tree whose branches are followed until a device's ROM number is found as a leaf. The search procedure continues until all devices participating are discovered (Maxim Integrated Inc, 2018), which is then transformed into a Petri net model serving as a case study. Because this algorithm required multiple procedures in search of multiple loops to exactly find every device on the shared bus, the equivalent Petri net model is complicated, but it remains as a state machine. Hence, we qualify this model as a good candidate to test the exactness of the simulator.

From the above-stated algorithm, we use the flowchart-to-Petri-net transformation scheme (Shen et al., 2018) to obtain Petri nets for simulation, which are known as ordinary state machines. Figures 1, 2, and 3 represent the transformed 1-wire search Petri nets drawn by HiPS, PIPE, and WoPeD, respectively. Regarding the models for 1-wire search algorithm, only HiPS and WoPeD support hierarchical modeling. Therefore, we can break the bulky net into another sub-page. In contrast, the model is displayed totally in just one page by PIPE simulator, which might cause confusion in tracking the Petri net.

With these models, we perform simulations with the analysis module of each PN tool. Hereby, all PN tools must be able to precisely deal with the basic PN models. Every property checking tool for behavioral and structural properties is required to accurately return results, and the modules for analysis methods can exactly provide reports about the Petri net verified. All the tools must also be stable in the whole experiment. These sorts of qualities can be checked and compared from the basic analysis modules activated. As the number of features provided by these tools is too huge and we are unable to cover everything, only the noticeable outcome is given in the last part of the results. The result first provides a test report regarding the achievement from the tool with
supporting most analysis modules, and then make a comparison for the similarities and differences among three simulators. These works can give a clear point of view about their quality and the distinction among them. The resulting comparison might become valuable fundamentals to determine the appropriate software tool for PN tool users according to their desires.

4. Main results

In this Section, the general information about HiPS, PIPE, and WoPeD is first presented in Table 1. The next Sub-sections show the resulting comparison in editing, simulation, and analysis features. Last, from the simulation experiment, some significant statements regarding the software performance on three Petri net models are provided, giving a recommendation on tool selection for users.

4.1. Main findings of the present study

Some general facts regarding the licensed years, developing language, operating system, and types of Petri net supported are shown in Table 1.

Among three PN tools, HiPS has been licensed and developed later. However, compared to other two software tools, PIPE and WoPeD, developed in Java language, it supports fewer operating systems and types of Petri net. For a higher version of HiPS, HiPS 2 does support colored Petri net.

4.2. Comparison with other studies

In this Sub-section, the features offered by the PN tools are compared with one another, as shown in Table 2. As a PN tool recommendation, all the PN tools have graphical editors and interactive simulators which are quite readable and easy to handle.

Meanwhile, among the trio, HiPS seems to support fewer options for users than other two competitors. The biggest inconvenience about HiPS is that it does not support manual simulation for all types of PN. Regarding PIPE and WoPeD, the weakness of PIPE is the lack of hierarchical

| Features Analysis       | HiPS | PIPE | WoPeD |
|-------------------------|------|------|-------|
| Reachability            | x    | x    | x     |
| Boundedness             | x    | x    | x     |
| Liveness                | x    | x    | x     |
| Reversibility           |      | x    |       |
| Coverability            |      |      |       |
| Persistence             |      |      |       |
| Synchronic distance     |      | x    |       |
| Structural liveness     |      |      |       |
| Controllability         |      |      |       |
| Structural boundedness  | x    |      |       |
| Conservativeness        | x    |      |       |
| Repetitiveness          | x    |      |       |
| Consistency             |      |      |       |
| Coverability tree       |      |      | x     |
| Incidence matrix        | x    |      |       |
| T-Invariant             | x    |      |       |
| S-Invariant             |      | x    |       |
| Other modules available | x    |      |       |
modeling as stated before, while the problem for WoPeD is that the arc weight cannot be modified and always remains 1. However, PIPE and WoPeD have their own additional editing tools, like inhibitor arc for PIPE and multiple windows for WoPeD.

4.3. **Implification and explanation of findings**

The features available to model analysis are considered as a key factor for Petri net tool selection. The modules available for analyzing the behavioral properties, structural properties, and analysis methods are presented in Table 3. Obviously, three PN tools do not support analyzers for coverability, persistence, structural liveness, and controllability.
Figure 2. 1-wire search Petri net model by PIPE.
Figure 3. 1-wire search Petri net model by WoPeD.
In contrast to editor and simulator supported, HiPS provides significant greater quantity of basic analysis tools than PIPE and WoPeD. Two Java-developed software systems only allow users to verify a few essential properties such as reachability, boundedness, liveness, and coverability tree. However, they do support tools for analyzing other noticeable characteristics. WoPeD can do capacity planning and quantitative simulation, as well as edit and visualize process metrics with their relation to the process graph. PIPE, as opensource software, not only provides a suite of analysis modules, but also allows users to install other packages, including system developer-supported and user-defined modules, into the suite. Consequently, the verification potential in PIPE is higher than that in HiPS. Some interesting modules in PIPE are net classification, comparison, minimal siphons and minimal traps, and response time analysis.

Thus, each PN tool has its own applicability in each use case. Yet, HiPS has been proved that it is the best choice of PN simulator regarding sufficient analysis modules, which is simple and adequate.

4.4. Strengths and limitations

With regards to observing the capability of HiPS, PIPE, and WoPeD in practical simulation, the trio conducted one experiment to reveal their effective performance. The results and related comparison are presented hereafter. From the comparative study in Section 4.3, HiPS is the simulator supporting with most features analysis, and its overall results are chosen to display as a table in the corresponding use case. Then, the results from PIPE and WoPeD are compared with HiPS’s outcome to show their similarity and contrast among them.

4.4.1. 1-wire search

The behavioral properties, structural properties, and analysis methods for 1-wire search are summarized in Table 4.

In Table 4, the 1-wire search net satisfies almost all the properties of Petri nets through HiPS’s analysis, as its nature is an ordinary state machine. However, liveness returns a deadlock when the token reaches place P29 after firing transition T38, which is not exact because the token cannot move anymore just due to its staying at the terminal. Deadlock checking is an important task for PN analysis because this has been viewed as a huge problem for many resource allocation systems over the last decade (Barylska et al., 2020; Helouet & Kecir, 2020). Hence, this kind of result needs to be avoided to improve the reliability of the analysis stage. To do that, we can simply add a computation node N40 corresponding to transition T40 between the start and the final places.

| Property/Method         | Results                                                                 |
|-------------------------|-------------------------------------------------------------------------|
| Reachability            | 29 reachable markings                                                   |
| Boundedness             | 1-bounded                                                               |
| Liveness                | 1 Deadlock: T1->T2->T4->T6->T7->T8->T9->T37->T38                         |
| Reversibility           | No result yet (Not functioning)                                         |
| Structural Boundedness  | Structurally bounded                                                    |
| Conservativeness        | Conservative                                                            |
| Repetitiveness          | Not repetitive                                                          |
| Consistency             | Not consistent                                                          |
| T-Invariant             | 10 T-invariants                                                         |
| S-Invariant             | 1 S-invariant                                                           |
The additional node, i.e. transition, represents the restart of a process, assuring that it can run repeatedly without standoff state. Another issue is that the reversibility calculating module takes a huge amount of time, so this feature needs to be improved in the future release of the software tool. The FC/AC Liveness/Safeness Checker module is suffered from a transparent appearance (Davison et al., 2020). Apart from that, the reduction rules are helpful features as they assist users in automatically decreasing the complexity of the PN model.

For PIPE and WoPeD, these simulators also produce the same results as HiPS in reachability, boundedness, $T$-invariant, and $S$-invariant. The identical outcomes among HiPS, PIPE, and WoPeD mutually show the accuracy in their computation. PIPE also experienced one deadlock in the sink place, and this problem can also be solved by an additional transition $T_40$ like HiPS. However, by WoPeD, as this tool is designed for the workflow-like PNs, the deadlock in the sink place is automatically eliminated. Thus, the result from the net by WoPeD is liveness, which is deadlock free. Otherwise, both PIPE and WoPeD provide a user-friendly interface representing coverability tree, together with other enhanced analyzers which enable users to feel more confident in the validity of the PN.

The similar reports in the 1-wire search case are also retrieved by PIPE and WoPeD. There is only one noticeable phenomenon related to WoPeD's behavior as we observe from the experiment. The incident is that due to bulky size of the net, when we open a property-editing window for a transition near the net's bottom, it might appear outside the desktop screen. This bug causes the editor blocked from other actions, which is like a functioning stop. One temporarily solution to resolving this problem is resizing the net in the workspace to be smaller so that we can get the editor appearing inside the screen.

4.5. PN tool recommendation

One case study has proved that three tools are able to accurately perform their analysis modules available to simple PNs, despite some minor limitations. Regarding the strengths and weaknesses of each tool, we provide some reasonable clues about suitable options for PN tool users.

HiPS's disadvantages include the lack of supporting operating system, editing tools, and simulation functions in comparison with PIPE and WoPeD. However, it outweighs two Java programmed software systems in the basic analysis aspect. Therefore, we recommend HiPS for people starting to study the properties and analysis methods of PNs, which required an easy-to-use and adequate supporting tool. Nevertheless, the reversibility checking module is not approved due to its poor performance.

On the other hand, PIPE and WoPeD are great but simple software tools for PN designation and testing, thanks to their convenient drawing and simulation components. In addition, users beginning to be familiar with ordinary PNs can also enhance their skills and proficiency with other PN classes, characteristics, and verification methodologies through the extended options supported by these simulators.

Last but not the least, HiPS, PIPE, and WoPeD usage can always be combined to achieve higher efficiency in the study of PN tools. To illustrate an example, we can use PIPE or WoPeD to first simulate the behaviors of a PN model. Thus, three tools might be applied together to analyze every single characteristic of that net for more information and to verify the correctness of the system developed.

5. Conclusion

In this paper, the utilities of HiPS, PIPE and WoPeD have been completely surveyed for PN tool users. As computer simulation can obtain safer systems, perform program testing, and minimize the manufacturing cost and extra time span, it helps us protect companies'
resources and business. Most importantly, fast developing some manufacturing systems is also a vital task because it can enhance the system performance and eliminate the bugs inside, which are performed well through model simulation. Consequently, the simulation tools have made their great contributions in the field of systems engineering.

The contributions of this work include the following items:

(1) It provides users with the general information about three practical simulators based on PN theory.

(2) Some experiments were conducted to examine their strengths and limitations depending on their use case. HiPS has shown its benefits in analyzing PN basic properties. On the other hand, PIPE and WoPedD have presented their favors to PN design and simulation, as well as their potential functions for system verification and performance evaluation.

(3) The suitable simulator for PN tool users can be recommended, giving a clear idea about how the tools should be appropriately applied based on their own advantages. In consequence, these three PN simulation tools have their own potentials for assisting users in quickly managing PN properties for the flexible manufacturing system development.

(4) The well-known 1-wire search algorithm was illustrated to clearly identify the strengths and limitations of the three PN tools.

5.1. Future direction
Our future research on this topic will provide deeper information about the analysis modules for PIPE and WoPedD together with exploiting HiPS 2 usage. In other words, if we can determine their more practical applications, we will gain more experiences in the smart selection from the three PN modeling tools. In addition to these three PN simulation tools, more PN-related ones will be further explored.

Acknowledgements
The authors are grateful to the anonymous reviewers for their constructive comments which have improved the quality of this paper. Also, the great efforts put by Mr. Jianzhi Wang to help us redraw all figures of Petri net model are highly appreciated.

Funding
This work was supported by the Ministry of Science and Technology, Taiwan, ROC, under grants MOST 107-2221-E-845-001-MY3 and MOST 110-2221-E-845-002.

Author details
Yi-Nan Lin1
Cheng-Ying Yang2
Gwo-Jen Chou3
Sheng-Kuan Wang4
Victor R.L. Shen5
E-mail: rshen@mail.nptu.edu.tw
Yu-Ying Wang6
Hai Hoang Bu1
Jianzhi Wang7
Caggiano AlessandraReviewing editor
1 Department of Electronic Engineering, Ming Chi University of Technology, New Taipei City, Taiwan.
2 Department of Computer Science, University of Taipei, Taipei City, Taiwan.
3 Department of Electrical Engineering, National Formosa University, Taiwan Taiwan.
4 Department of Electrical Engineering, Ming Chi University of Technology, New Taipei City, Taiwan.
5 Department of Computer Science and Information Engineering, National Taipei University; and Department of Information Management, Chaoyang University of Technology, Taichung City, Taiwan.
6 Department of Applied Foreign Languages, Jinwen University of Science and Technology, New Taipei City, Taiwan.

Citation information
Cite this article as: The practical Petri net modeling and analysis for smart manufacturing system, Yi-Nan Lin, et al. Cogent Engineering (2020).

Data availability
The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure statement
No potential conflict of interest was reported by the author(s).

Citation information
Cite this article as: Smart selection from petri net modeling tools for fast developing a manufacturing system, Yi-Nan Lin, Cheng-Ying Yang, Gwo-Jen Chou, Sheng-Kuan Wang, Victor R.L. Shen, Yu-Ying Wang, Hai Hoang Bu & Jianzhi Wang, Cogent Engineering (2022), 9: 2020609.
References

Bokri, A. H., Alkibir MF, Awang N, Januzzi F, Ismail MA, Ahmad AN, Zokari IA, et al. (2021). Addressing the issues of maintenance management in SMEs: towards sustainable and lean maintenance approach. Emerging Science Journal, 5(3), 367–379. https://www.researchgate.net/publication/352371874_Addressing_the_Issues_of_Maintenance_Management_in_SMEs_Towards_Sustainable_and_Lean_Maintenance_Approach. https://doi.org/10.28991/esj-2021-01283

Barylska, K., Koutny, M., Mikulski, L., & Piatkowski, M., et al. (2020). Reversible computation vs. reversibility in petri nets. Science of Computer Programming, 151 (1), 48–60. https://scholar.google.com.tw/citations?view_op=view_citation&hl=en&user=ebQsXQIAAAAJ&citation_for_view=ebQsXQIAAAAJ:QIV2ME_5wuYC&for_view=ebQsXQIAAAAJ:QIV2ME_5wuYC&v2.5: http://ceur-ws.org/Vol-1295/352371874_Addressing_the_Issues_of_Maintenance_Management_in_SMEs_Towards_Sustainable_and_Lean_Maintenance_Approach. https://doi.org/10.28991/esj-2021-01283

Hou, C. Q., Li, S. Y., Cai, Y., Wu, H. M., An, A. M., & Wang, Y., et al. (2014, September). A livelock control policy for a flexible manufacturing system modeling with a subclass of generalized petri nets. Cogent Engineering, 1(1), 1–15. https://doi.org/10.1080/23311916.2014.944766

Kalistratov, D. (2019). Wireless video monitoring of the megacities transport infrastructure. Civil Engineering Journal, 5(5), 1033–1040. https://www.civiljournal.org/index.php/cj/article/view/1445. https://doi.org/10.28991/ciej-2019-03091309

Li, Z., Zhou, M., & Wu, N., et al. (2019). A survey and comparison of petri net-based deadlock prevention policies for flexible manufacturing systems. IEEE, Trans. On Systems, Man, and Cybernetics, 38 (2), 173–188. https://www.researchgate.net/publication/220509833_A_Survey_and_Comparison_of_Petri_Net-Based_Deadlock_Prevention_Policies_for_Flexible_Manufacturing_Systems.

Lin, Y.-N., Hsieh, T.-Y., Yang, C.-Y., Shen, V. R. L., Juang, T.-T., & Huang, T.-J., et al. (2020). Review on petri net modeling and analysis of a smartphone manufacturing system. Cogent Engineering, 7(1), 1–25. https://doi.org/10.1080/23311916.2020.1851630

Osman, R., & Harrison, P. G. (2019). Approximate closed fork-join queueing networks using product-form stochastic petri nets. Journal of Systems and Software, 110 (8), 264–278. https://www.sciencedirect.com/science/article/pii/S0164121215001859https://doi.org/10.1016/j.jss.2015.08.036. https://www.sciencedirect.com/science/article/pii/S0164121215001859https://doi.org/10.1016/j.jss.2015.08.036

Pla, A., Gay, P., & Melendez, J., et al. (2014). Petri net-based process monitoring: A workflow management system for process modeling and monitoring. Journal of Intelligent Manufacturing, 25(3), 539–554. https://doi.org/10.1007/s10845-012-0704-z

Shen, V. R. L., Shen, R.-K., Yang, C.-Y., & Chen, Y.-C., et al. (2018). Petri net modeling and analysis of an iPad manufacturing system. Systems Engineering 21, (2), 115–130. https://doi.org/10.1080/10991341.2018.1476650

Souravlos, S. I., & Roumeliotis, M. (2015, June). Petri net modeling and simulation of pipelined redistributions for a deadlock-free system. Cogent Engineering, 2(1), 1–15. http://doi.org/10.1080/23311916.2016.1351670

Souravlos, S. I., & Roumeliotis, M. (2015, June). Petri net modeling and simulation of pipelined redistributions for a deadlock-free system. Cogent Engineering, 2(1), 1–15. http://doi.org/10.1080/23311916.2016.1351670

Hefetz, L., & Kecir, K. (2020). Realizability of schedules by stochastic time petri nets with blocking semantics. Science of Computer Programming, 157 (2), 71–102. https://doi.org/10.1016/j.scico.2017.12.004
