# Petri Net Tools: a Comparative Analysis

Abhilash and Rajendra Prasad Mahapatra

December 9, 2024

# Petri Net Tools: A Comparative Analysis

**Abhilash**[1*], **Rajendra Prasad Mahapatra**[2]

*[1*,2]Department of Computer Science and Engineering, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Delhi-NCR Campus, Modinagar, Ghaziabad, Uttar Pradesh 201204, India,*

*Tel.: +91- 79068 49434*

**Corresponding author:** Abhilash
mail id: abhilashsharma@gmail.com

***Abstract*—** This survey paper explores Petri Net tools relevant to concurrent, non-deterministic, stochastic, and parallel systems. It begins with a brief introduction to Petri Nets, discussing their formalism, properties, and variants. The paper then examines several Petri Net tools, including the Oris Tool, Tapaal Tool, and Color Petri Net (CPN) Tool, detailing their modeling capabilities and presenting the results obtained from the problems addressed.

***Keywords*—*Petri Net; Repairman; Petri Net Tools :Oris, Tapaal, Color Petri Net(CPN)***

## I. Introduction

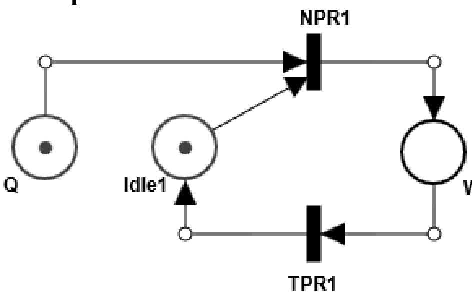
In 1962, Carl Adam Petri presented Petri Nets as part of his PhD thesis, introducing a mathematical model that can be visually represented to illustrate the dynamic behavior of systems. This graphical representation facilitates the understanding of parallel execution within a system, which is a key achievement of Petri Nets. This tool allows for system performance monitoring, enabling users to utilize it as a graphical editor, code generator, and model simulation. Petri Nets find applications across various engineering domains, including office automation, networking protocols, performance assessment, defense systems, telecommunications, e-commerce, railway networks, programming languages, flexible manufacturing, hardware architectures, research operations, and the Internet.

## II. Petri Nets

A Petri Net is characterized as a directed graph in which places are depicted as circles, representing various states within the system. Transitions are illustrated as rectangular boxes that become activated under specific conditions, facilitating the movement of tokens to subsequent states. The arcs in the graph indicate the flow of tokens, with their weights determining the number of tokens transferred to the next state; however, unit tokens are not explicitly shown on the arcs and are instead represented as directed connections. Arcs can connect places to transitions or transitions to places. The configuration of tokens within the places of a Petri Net model is denoted by 'M', indicating the current number of tokens. Throughout the execution of a Petri Net model, both the quantity and distribution of tokens may vary. A Petri Net can be defined as: N= ( $p, t, i, o, M_0$ ), where

1. $p = p_1, p_2, \ldots, p_m$ , where $p$ is a place's finite set ;
2. $t = t_1, t_2, \ldots, t_m$, where $t$ is a transition's finite set, such that $p \cup t \neq \emptyset$, and $p \cap t \neq \emptyset$;
3. $I : p \times t \rightarrow N$, where $i$ is an input function that shows directed arcs from places to transitions, where N is a set of non-negative integers;
4. $o: t \times p \rightarrow N$, where $o$ is an output function that shows directed arcs from transitions to places;
5. $M_0 : p \rightarrow N$ , where $M_0$ is defined as initial marking

**Example 1:**



### A. Transition Firing

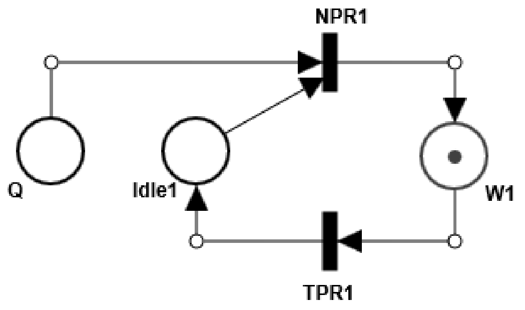
A Petri Net is executed by the event causing the firing of a transition, which occurs when the number of tokens in the various places of a Petri Net model changes due to the occurrence of an event.

**Enabling Rule**: A transition can only be enabled when each input place has a minimum number of tokens equal to the weight of the arc.

**Firing Rule**: Only the enabled transition can be fired which means it follows the enabling rule. From input places, several tokens are moved to the transition and lastly to the output place equal to the weight of the arc.

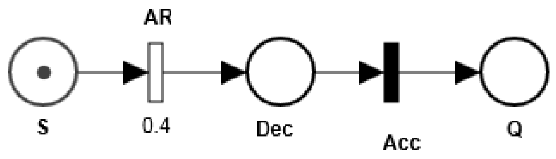
After the firing of the transition, each place always contains non-negative tokens. A source transition is defined where no input place is there, which is enabled and the sink transition is the one without the output place, which implies that no tokens can be generated after the firing of the transition.

**Example 2**: Transition Firing

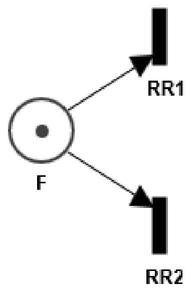


*B. Modeling Power*

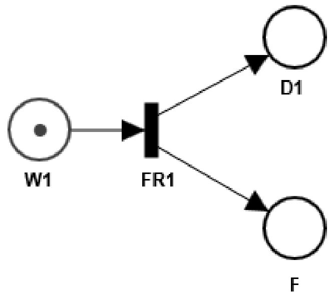
1. **Sequential Execution** - In a sequential execution, transition *Acc* can only be fired after transition *AR*, which shows the precedence sequencing.



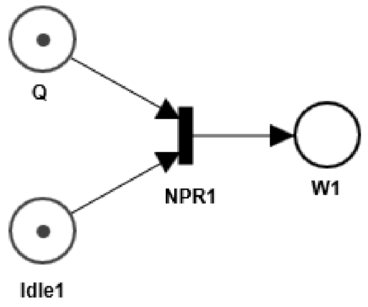
2. **Conflict** - In this execution, transitions *RR1* and *RR2* are enabled, but the processing of one's request leads to the denial of the other transition. By assigning the probabilities to the transitions in the conflict state, conflict can be resolved.



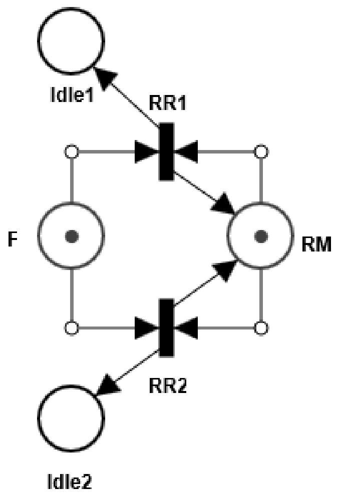
3. **Concurrency** – In this execution, tokens can be distributed to one of the output places *D1 or F* which are in concurrency, after the firing of transition *FR1*.



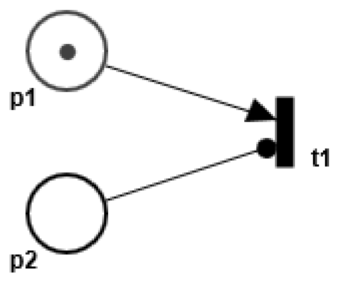
4. **Synchronization** – In this execution, the firing of transition *NPR1* depends on the places *Q* and *Idle1* which leads to the synchronized behavior of a system.



5. **Mutually Exclusive –** Whenever two concurrent processes are running, where a single resource has to be shared, then the processes are in a mutually exclusive stage, i.e. a resource can be shared one by one by both the processes. As shown in the figure, *RM* is the resource that needs to be shared mutually by the transitions *RR1* and *RR2*.



6. **Priorities –** By introducing the inhibitor arc, priorities can be obtained, which is graphically represented by an arc with a black dot connected to it. A transition *t1* can only fire when place *p1* has a minimum number of tokens equal to the weight of the arc and no token in place *p2* where the inhibitor arc is connected, which makes the change in enabling rule to the firing of transition.



## C. *Petri Net Properties*

2 types of properties are:

1. Behavioral Property – This property depends on the initial state or marking of a Petri Net.

2. Structural Property – This property depends on the net structure or topology of a Petri Net.

Behavioural Properties:

1. Reachability – It is defined in terms of places whether, in a Petri Net model, a place can be reached from one to another, and can perform certain behavioral properties accordingly as required by the system. It is important to identify the sequence of the transition's firing which can turn a marking from $M_0$ to $M_i$, where $M_i$ can be represented as a particular state to be defined as the destination after a transition is fired. If so, such a marking would be termed as reachable with a marking $M_1$, if firing of enabled transition results from $M_0$.

2. Safeness – It is referred to be safe if the boundedness is 1, i.e. if the number of tokens in place is less than or equal to j( a non–negative integer) which is defined as j-bounded. If any marking M is reachable from the initial marking $M_0$, a Petri Net is said to be j-bounded if each place is j-bounded (safe).

3. Liveness – A system is said to be live if, for any marking *M* that is reachable, a transition in the model can be fired by some firing sequences.

## D. *Variants*

1. **Time Petri Nets (TPN)** – TPN is defined as the time allocated with the transition (transition is always fired in real-time) i.e. when a transition is fired, a token in one place has to move to another place, but the token remains in transition during the firing sequence execution.
   For constructive solutions, state space and linear equations should be finite for bounded and unbounded PN.
   For firing a transition, a time delay is associated with it to hold the transition for a time delay (Delays are non-deterministic chosen) before the firing is executed, as in every case, transition firing immediately couldn't be possible. So, a time delay is referred to as $t_{min}$ and $t_{max}$, where $t_{min}$ is the time interval which is the minimum waiting time and $t_{max}$ is the maximum waiting time allowed for the firing of a transition. A hierarchical and object-oriented net was introduced at the time of the TPN study, known as the General Hierarchical Enhanced Net System (GHENeSys).

2. **Stochastic Petri Nets (SPN)** – The extended version of the simple PN is SPN. A random transition firing delays, and the firing of a transition is atomic, i.e. the movement of tokens from one place to another is in one go. In the Performance Evaluation, SPN is defined, which can be:

   **a.** One, the actual behavior of the system is monitored under certain situations, whereas the model needs more requirements than the obvious when the system isn't accessible then a prototype of a system can be obtained, in any form – digital or physical.
   **b.** Another one is, introduced in the planning phase of system modeling, which has the simulation models and detailed model. In the case of the simulation model, the model is generated by computer software, and in a detailed model, the model is in mathematical form.

3. **High-Level Petri Nets (HLPNs)** – HLPNs are used for the simulation of complex structures. Another form of HLPN is the Colored Petri Nets (CPNs), as the name suggests the important feature of CPNs is that the colors are involved with the data types to identify the tokens attached with the places, and the expressions to the arcs, so that each token is associated with its color.
   CPN can be created in a hierarchical manner where a net can be associated with the other net by using the toolset for larger models.

   **Color Petri Nets (CPNs):** CPNs are used in models where parallel systems, intelligence, and simultaneity are important. It is a graphical tool for the construction, accuracy, and authorization of systems. A combination of Petri Net and high-level PN forms the CPN, where PN helps in representing the model graphically for parallelism, simultaneity, and intelligence of a system, with high-level PN it helps in model creation, and alteration, with data types such as UNIT, REAL, STRING, etc. Here the model is created the same as PN by simply drag and drop process but with the data types assigned to it with the arcs, expressions are allocated to identify the transitions associated with the arc. Multiple transitions can be activated at the same time, transitions activated will be highlighted on their own, but transition is only activated when the color assigned to the arcs is in sync with the tokens assigned to the places. The important aspect of this tool is that the results are shown on the panel itself by simply clicking on the simulator button from the toolset defined as a play button, CPNs can be created in a hierarchical form which connects the net to the other nets in exact position to have a clearer view of the model defined.

Properties –

1. **Reachability:** This can be defined as the movement from initial marking to a particular state.
2. **Deadlock:** If a system doesn't have the binding elements, then a system can be in a deadlock condition.
3. **Livelock:** For better performance of a system, it should be livelock-free, which implies that if a system enters into the process and can't reach a different state.

### E. *Problem Taken:*

In this paper, the study of three Petri Net tools: Oris Tool, Tapaal Tool, and Color Petri Net (CPN) Tool; are made based on the model described below:

**Model:** A repairable multi-processor system is in use, where finite jobs are arriving at a defined rate into the system which is then stored in a queue. The queue allocates the jobs to the processor available.
When the processor is assigned the job then it may result in successful execution, which adds to the throughput, or may result in failure which is repaired by the repairman followed by the queue maintained on the basis of First-Come-First-Serve (FCFS) scheduling. After repair, the processor can again be allocated with the jobs to perform the task.

**Tools:**

1. **ORIS Tool** - Oris is a well-defined tool where by simply clicking on the options in the panel, a model can be evaluated, and verified graphically to understand the mechanics in real-time. *Places* are defined as circles, *transitions* as rectangular boxes, and *arcs* by the arrow. An *inhibitor arc* is also used for priority which is defined as the arrow ending with a small circle.
In this tool, transitions can be defined in different forms:

a. Simple Transition (t), transition fires on enabling condition
b. Time Transition (n), EFT & LFT can be denoted as [1, 2].
c. Stochastic Immediate Transition (IMM), where the default weight is 1.
d. Stochastic Uniform Transition (UNI), where a simple condition of uniform probability is defined between EFT and LFT.
e. Stochastic Deterministic Transition (1), where value & weight is 1.
f. Stochastic Exponential Transition (λ), where transition can be defined at a particular rate.
g. Stochastic Expolinomyal Transition (expo), where an expression to the transition can be defined.

To run the tool, the *Show Token Game View* button is defined in the toolbar with the symbol 'T' and to check the log, graphs, and details of the system, click on the *Show Engine view* button.
Analysis of the system can be made with the following, by clicking on the play button in the panel:

i. Regenerative transient analysis
ii. Regenerative steady-state analysis
iii. Forward transient analysis
iv. Transient analysis of GSPN
v. Steady state analysis of GSPN
vi. Enabling restriction transient analysis
vii. Non – Non-Deterministic Analysis
viii. Transient simulation

Let's look at Figure -1 and Figure -2 below to understand the working of Oris Tool on a model described earlier.

Figure–1 depicts the working, where ten incoming jobs can arrive at an 'AR' rate, which is being accepted by 'Q', and allocation of jobs to the processors is made on a random basis, if the job is executed then it will count the job in 'TP', and if the processor fails then it will move to repairman's queue 'F' which is based on FCFS scheduling. After repair, the processor will be re-aligned with the job to their initial position.

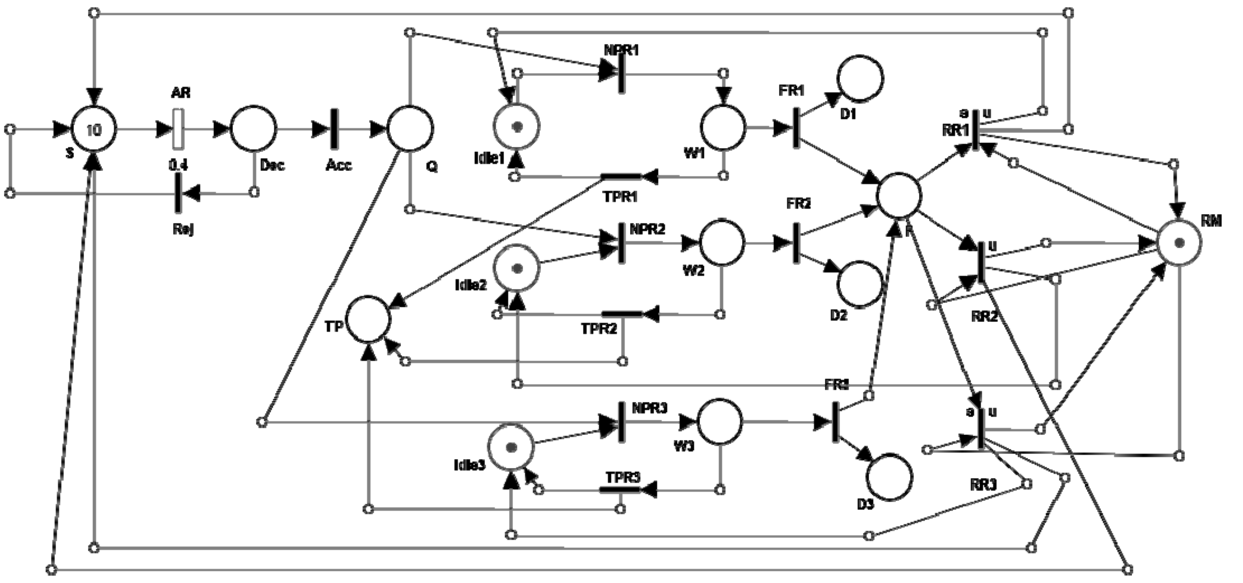
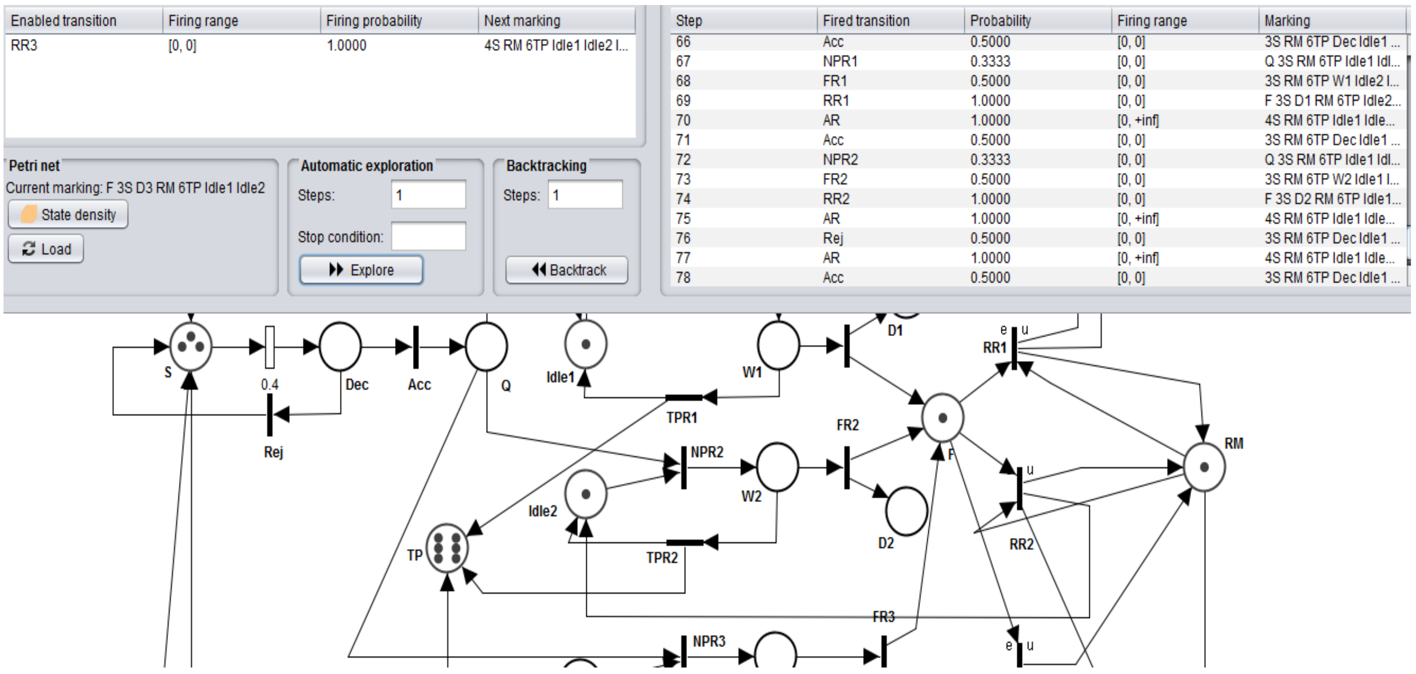


Figure - 1



| Enabled transition | Firing range | Firing probability | Next marking |
|---|---|---|---|
| RR3 | [0, 0] | 1.0000 | 4S RM 6TP Idle1 Idle2 I... |

**Petri net**
Current marking: F 3S D3 RM 6TP Idle1 Idle2

**Automatic exploration**
Steps: 1
Stop condition:

**Backtracking**
Steps: 1

| Step | Fired transition | Probability | Firing range | Marking |
|---|---|---|---|---|
| 66 | Acc | 0.5000 | [0, 0] | 3S RM 6TP Dec Idle1 ... |
| 67 | NPR1 | 0.3333 | [0, 0] | Q 3S RM 6TP Idle1 Idl... |
| 68 | FR1 | 0.5000 | [0, 0] | 3S RM 6TP W1 Idle2 I... |
| 69 | RR1 | 1.0000 | [0, 0] | F 3S D1 RM 6TP Idle2... |
| 70 | AR | 1.0000 | [0, +inf] | 4S RM 6TP Idle1 Idle... |
| 71 | Acc | 0.5000 | [0, 0] | 3S RM 6TP Dec Idle1 ... |
| 72 | NPR2 | 0.3333 | [0, 0] | Q 3S RM 6TP Idle1 Idl... |
| 73 | FR2 | 0.5000 | [0, 0] | 3S RM 6TP W2 Idle1 I... |
| 74 | RR2 | 1.0000 | [0, 0] | F 3S D2 RM 6TP Idle1... |
| 75 | AR | 1.0000 | [0, +inf] | 4S RM 6TP Idle1 Idle... |
| 76 | Rej | 0.5000 | [0, 0] | 3S RM 6TP Dec Idle1 ... |
| 77 | AR | 1.0000 | [0, +inf] | 4S RM 6TP Idle1 Idle... |
| 78 | Acc | 0.5000 | [0, 0] | 3S RM 6TP Dec Idle1 ... |

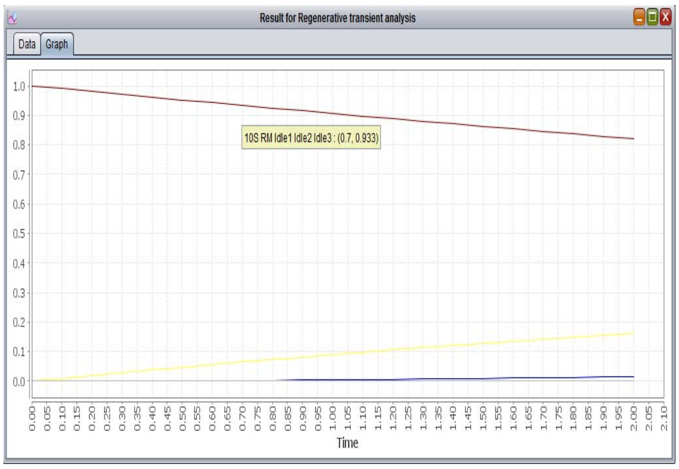
Figure – 2

In Figure –2, the execution of the model is shown, where incoming jobs at 'S' are three and at 'TP' six which means out of ten jobs six are executed and one job is in processor 3 which failed and is in repairman's queue, to get the processor repaired by following FCFS scheduling.

**Abbreviations:**

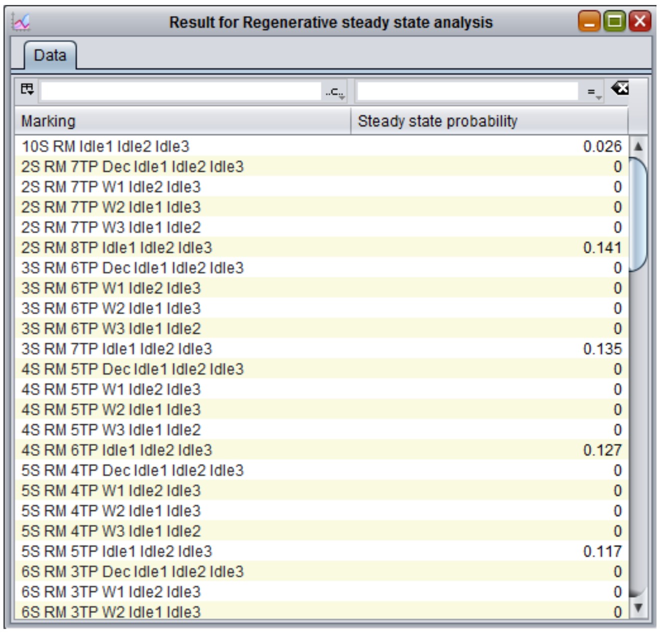
S – Source
AR – Arrival Rate
Dec - Decision
Acc – Accept
Rej – Reject
Q – Queue
Idle1 – Idle State 1
Idle2 – Idle State 2
Idle3– Idle State 3
W1- Working State 1
W2 – Working State 2
W3 – Working State 3
NPR1 – Normal Processing Rate 1
NPR2 – Normal Processing Rate 2
NPR3 – Normal Processing Rate 3
TPR1 – Throughput Rate 1
TPR2 – Throughput Rate 2
TPR3 – Throughput Rate 3
FR1 - Failure Rate 1
FR2 - Failure Rate 2
FR3 - Failure Rate 3
D1 – Dead State 1
D2 – Dead State 2
D3 – Dead State 3
F – Repairman's Queue
RR1 – Repair Rate 1
RR2 – Repair Rate 2
RR3 – Repair Rate 3
RM - Repairman

**Output:** Regenerative Transient Analysis



This output shows the Regenerative Transient Analysis of Figure – 2, where the Y-axis shows the 'Time' and the X-axis shows the Probability where the discretization value is 0.1.

**Output:** Regenerative Steady State Analysis

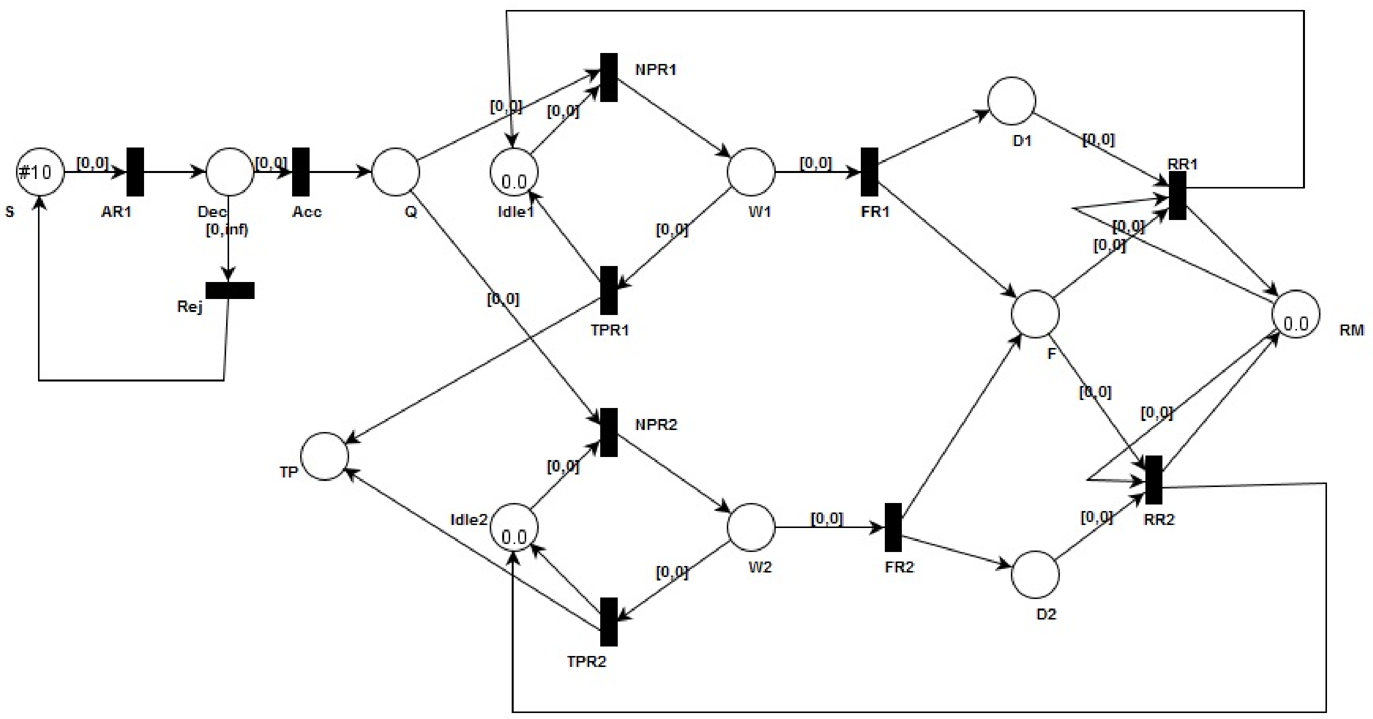

Result for Regenerative steady state analysis
Data

| Marking | Steady state probability |
|---|---|
| 10S RM Idle1 Idle2 Idle3 | 0.026 |
| 2S RM 7TP Dec Idle1 Idle2 Idle3 | 0 |
| 2S RM 7TP W1 Idle2 Idle3 | 0 |
| 2S RM 7TP W2 Idle1 Idle3 | 0 |
| 2S RM 7TP W3 Idle1 Idle2 | 0 |
| 2S RM 8TP Idle1 Idle2 Idle3 | 0.141 |
| 3S RM 6TP Dec Idle1 Idle2 Idle3 | 0 |
| 3S RM 6TP W1 Idle2 Idle3 | 0 |
| 3S RM 6TP W2 Idle1 Idle3 | 0 |
| 3S RM 6TP W3 Idle1 Idle2 | 0 |
| 3S RM 7TP Idle1 Idle2 Idle3 | 0.135 |
| 4S RM 5TP Dec Idle1 Idle2 Idle3 | 0 |
| 4S RM 5TP W1 Idle2 Idle3 | 0 |
| 4S RM 5TP W2 Idle1 Idle3 | 0 |
| 4S RM 5TP W3 Idle1 Idle2 | 0 |
| 4S RM 6TP Idle1 Idle2 Idle3 | 0.127 |
| 5S RM 4TP Dec Idle1 Idle2 Idle3 | 0 |
| 5S RM 4TP W1 Idle2 Idle3 | 0 |
| 5S RM 4TP W2 Idle1 Idle3 | 0 |
| 5S RM 4TP W3 Idle1 Idle2 | 0 |
| 5S RM 5TP Idle1 Idle2 Idle3 | 0.117 |
| 6S RM 3TP Dec Idle1 Idle2 Idle3 | 0 |
| 6S RM 3TP W1 Idle2 Idle3 | 0 |
| 6S RM 3TP W2 Idle1 Idle3 | 0 |

This output shows the Regenerative Steady State Analysis of Figure 2, where marking at places is shown with the steady-state probability.
Example – 10S RM Idle1 Idle2 Idle3 means there are 10 tokens at Source, 1 at Repairman, 1 each at Idle1, Idle2, and Idle3; and steady-state probability 0.026 means the possible outcome to the marking defined at a point.

2. **TAPAAL Tool –** TAPAAL was developed at **Aalborg** University in Denmark. TAP stands for **T**imed-**A**rc **P**etri Nets.
   Here, *places* are represented as circles, *transitions* as rectangular boxes, and *arcs* by an arrow, *inhibitor arc* is enabled when no token is present in the input place with the place having a token in it, only then a transition would fire.
   To insert or remove a token in the *places*, separate icons are present in the toolbar and to run a model a simulator button is specified in the form of a 'flag' symbol. In case, where two transitions are enabled at the same time, we can specifically execute a transition by a simple click.

Figure – 3



3. **Color Petri Net (CPN) Tool –** The name itself defines the characteristics of this tool, where *places, transitions, and arcs* can be given different colors to analyze the work. The CPN tool can be downloaded from http://cpntools.org/.

While building a net, the token game can be easily analyzed by just watching the token movement on every transition firing, where places can be given different color sets.

Different tool sets are provided in the toolbar:

a. *Net* - A new net can be created, saved, closed, and open saved nets.

b. *Create* – By simply dragging and dropping an icon from the net, a place, transition, arc, inhibitor arc, clone, delete, or horizontal/vertical guideline can be created.

c. *Style* - Different colors to places can be defined to identify the token game.

d. *View* is a toolset from where a net can be zoomed in/out, a new group can be created, can also toggle the elements to identify whether it belongs to a group or not.

e. *Simulation* is a toolset from which a model can be simulated one by one by pressing the play button or multiple movements at one go by pressing the play button which indicates the number of moves. The backward button is to reach the initial stage.

f. *Standard Declaration* is a toolset where pre-defined declaration sets are available, also we can define our declaration set as per the requirement.

To assign the token to the places, press the Tab key once to define the declaration set and twice to initiate the marking. An expression to the arc is defined by clicking onto the arc once.

Figure – 4 shows the binder available in the Color Petri net tool and Figure – 5 shows the model described earlier in this paper.

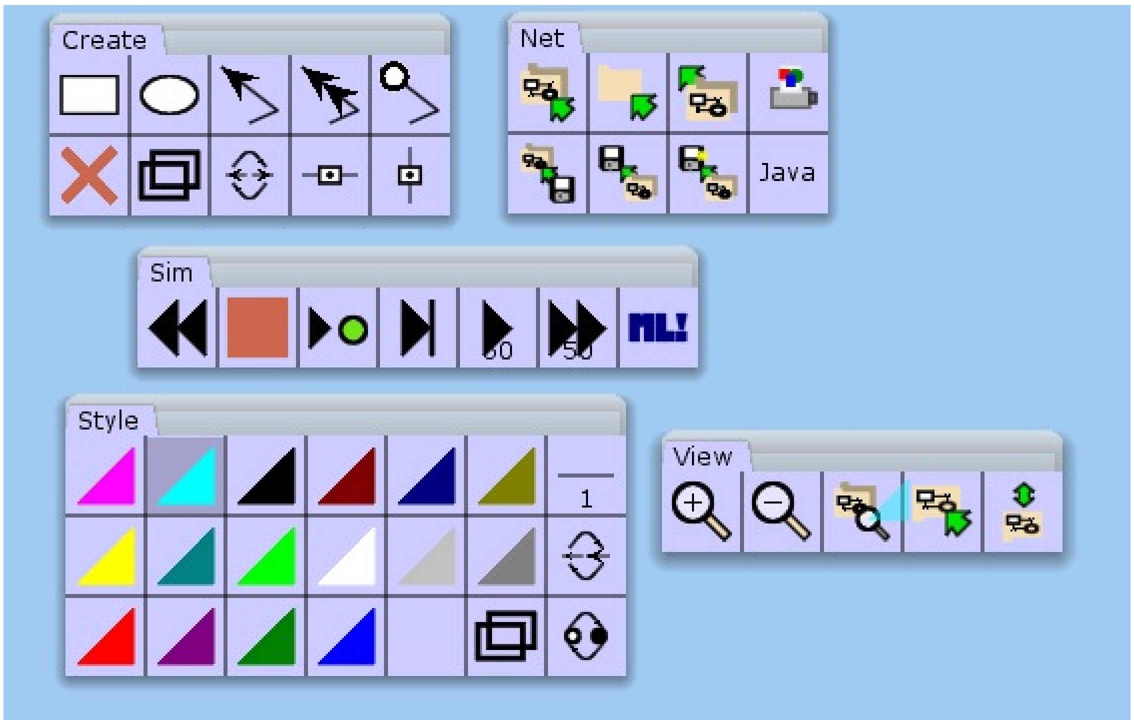


Figure - 4

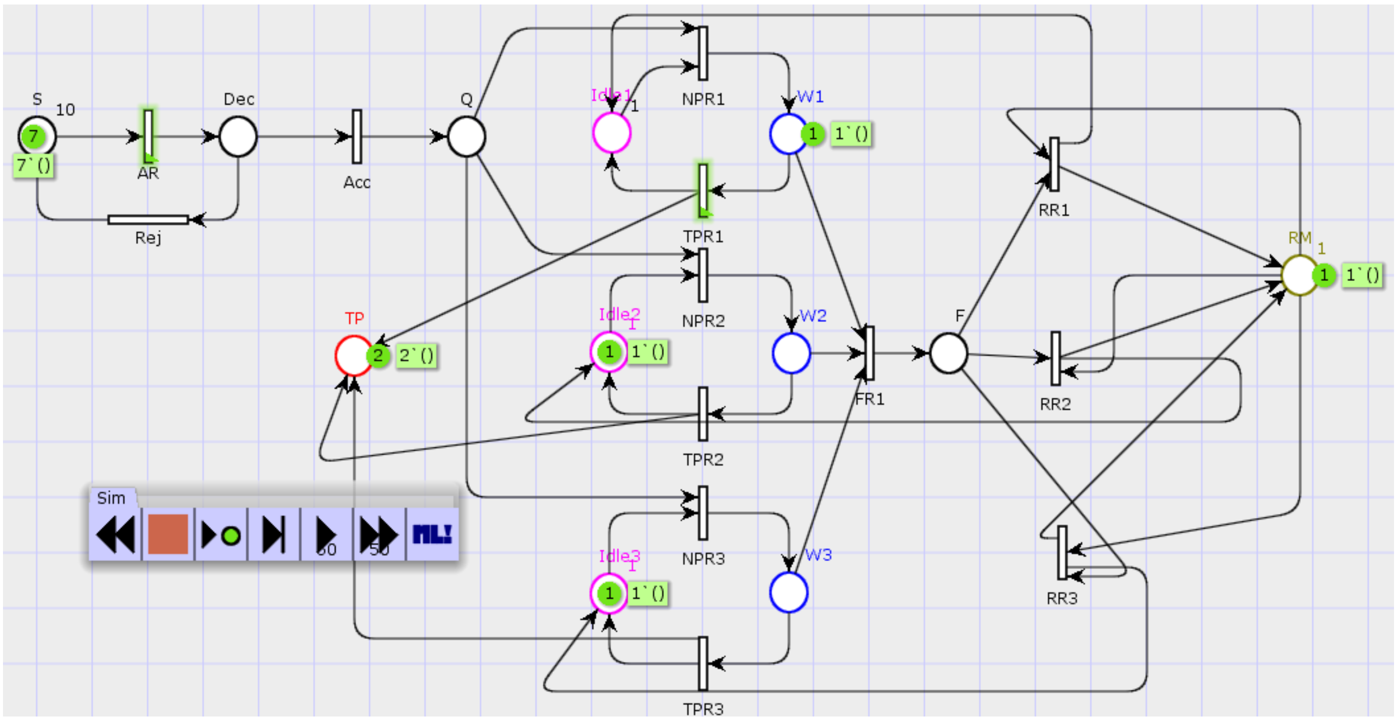


Figure - 5

References

[1] Jiacun Wang, Petri Nets for Dynamic Event-Driven System Modeling, Monmouth University, West Long Branch, NJ 07764.

[2] Van der Aalst, Wil, and Kees van Hee, Workflow Management: Models, Methods, and Systems, Massachusetts: MIT Press.

[3] Ajmone Marsan, M., M. G. Balbo and G. Conte. 1986. Performance Models of Multiprocessor Systems, Massachusetts: The MIT Press.

[4] Desrochers, A., and R. Ai-Jaar. 1995. Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis. IEEE Press.

[5] Murata, T. 1989. Petri nets: properties, analysis and applications. Proceedings of the IEEE 77(4): 541-580.

[6] Peterson, J. L. 1981. Petri Net Theory and the Modeling of Systems. N.J.: Prentice-Hall.

[7] Petri, C.A. 1962.Communication with Automata, Kommunikation mit Automaten. Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York.

[8] Marco Biagi, Laura Carnevali, Enrico Vicario; University of Florence, Italy

[9] Jonas F. Jensen, Thomas Nielsen, Lars K Oestergaard and Jiˇr´ı Srba; AALborg University, Denmark

[10] Lars M. Kristensen, Bergen University College, Norway

[11] Abhilash, Ram Chakka, Rama Krishna Challa, "Numerical Performance Evaluation of Heterogeneous Multi-Server Models With Breakdowns and FCFS, LCFS-PR, LCFS-NPR Repair Strategies", Advance Computing Conference (IACC), Volume: pp. 566-570, 2013, 2013-02-22.