An Inclusive Study of Petri Nets

Jasmine McDuffie, Jules Chenou
Norfolk State University College of Science, Engineering and Technology

July 2025

Abstract

The main purpose of this report is to provide information on Petri nets, including their design, semantics, properties, and uses. The problem addressed is how to model and analyze systems that may be susceptible to unknown issues. By visualizing these issues, we can improve or modify the system. Petri nets are highly powerful tools that can solve several problems, including issues with workflow management, deadlock, data transmission in communication systems, manufacturing and production, software systems, and more. These things are widely used in different work fields such as engineering, project management, and computer science. It is important to solve these problems for maintaining system reliability, and optimal performance. Exploring all the uses and techniques of a Petri net, beginning with the basics and advancing to the more complex ideas, demonstrates just how effective they are. The results confirm that Petri nets are important for identifying systems that need improvement within their performance and components across multiple fields.

Keywords: Petri nets, semantics, workflow management, deadlock, data transmission, communication systems, manufacturing, production, reliability, performance.

I. Introduction

In engineering, computer science, and other fields, the challenge of modeling and analyzing systems and processes that have significant issues running due to unknown factors is prevalent. Petri nets are a graphical and mathematical modeling language used to describe and analyze a variety of different systems and processes. They enable the discovery of these possible issues such as deadlock, overflow, resource conflict, and more. Computer scientists and engineers get the most use out of Petri nets, as they are used extensively to detect errors in systems and optimize production flow, however, many other disciplines find them useful too. They provide a graphical model and a mathematical substructure, allowing you to examine the behavior of a system visually and represent it. They are especially useful for modeling systems where the main states and the processing are separate.

Petri nets were originally introduced by Carl Adam Petri in 1962 in his PhD thesis, “Communication with Automata”. They started simply modeling concurrent processes in systems, but have evolved to do so much more. They can manage workflow, communication protocols, manufacturing systems, and more. When examining recent advancements and case studies, you will find that Petri nets were used to solve some of the adversities that were faced. For example, Petri nets have recently been used to model and optimize healthcare workflows such as treatment pathways. This example not only shows that Petri nets are causing advancements but also proves that they have uses across many disciplines.

All the intricacies of Petri nets and their many applications will be covered along with details of their layout, semantics, and techniques used to analyze them. These topics will be illustrated through case studies and other examples that show how Petri nets are used to model and analyze real-world systems and give solutions to common challenges. Readers will gain a complete understanding of Petri nets and their importance throughout all fields. Section II will encompass a literature review, discussing some work that has been done with Petri nets in fields outside of computer science and engineering and defining Petri nets. Section III details the operational semantics of a Petri net, telling the reader how it behaves. The multiple properties of a Petri net are presented in Section IV. Analysis methods are in Section V. Different applications can be found in Section VI and methodology is in Section VII. Section VIII gives the reader methods to create Petri nets. Section X introduces my project which is an overall accumulation of the information in this paper. Section XI explains my motivation and the purpose behind my project. Section XII details what chatbots are to further explain the background of my project. Section XIII gives an overview of the components used in my project. Section XIV goes further into the component,

explaining them in more detail. Section XV included some testing I conducted while working on my project and the results of the testing.

II. Literature Review

Related Work

Petri nets can be applied to almost any area or system that can be described graphically. They have been heavily used in workflow management to optimize business processes, in the manufacturing field, and in project management. In Van der Aalst's technical report, "The Application of Petri Nets to Workflow Management", he details how Petri nets can be used to detect and repair workflow conflicts such as deadlock. He and others realized that as things become more modernized and technological, there is a need for systems that do much more than older ones did. This work prepared the way for more things that aid in modern business process management systems that use Petri nets and Petri net-based models to guarantee they are error-free.

According to the report "Applying Petri Nets on Project Management" written by Chang-Pin Lin and Hung-Lin Dai, they had a problem with a project management method called the Critical Path method which "considers the logical dependencies between related activities that may have to share common resources". They stated that as the number of resources they introduce increases, the graphs drawn using the Critical Path method become more difficult to draw. To solve this issue, they introduced Petri nets which better model complicated projects and resource sharing.

Definition 1:

A Petri net is a bipartite graph that consists of two main types of nodes, places, and transitions. Places are represented by circles and transitions are represented by rectangles or bars. Places and transitions are connected by directed edges called arcs. Arcs can only exist from place to transition or transition to place, never from place to place or transition to transition. Considerably the most important part of a Petri net is the tokens. Tokens are usually small black dots that reside in places and depict the resources that are being moved around in the Petri net. They represent the dynamic state of the Petri net. The distribution of tokens across places is called a marking.

Definition 2:

A Petri net is formally defined as a 4-tuple N = (P, T, I, O) where

P={p1, p2, p3…..pm} represents a finite set of places,

T= {t1, t2, t3….tn} represents a finite set of transitions,

I = Input places, (P x T) $\rightarrow N_0+$

O = Output places, (P x T) $\rightarrow N_0+$

Petri nets can also be represented using the set of arcs(F) and a weight function(W) in the Petri net definition instead of using input and output functions and adding in M_0 which depicts the initial marking.

F is a set of directed arcs that show flow relation, $F \subseteq (P \times T) \cup (T \times P)$

W is a weight function that assigns a positive integer weight to each arc W: F $\rightarrow N+$ M_0= {P, N} is the initial marking indicating the initial distribution of tokens.

III. Operational Semantics

The overall behavior of a Petri net is controlled by the movement of tokens through the transitions and places. Transitions represent events that can change the state of the system by consuming tokens from input places and producing them at output places.

Markings and Execution

The marking of a Petri net specifies how many tokens are in each place at any time as the Petri net is firing and changes as the Petri net fires. The firing of transitions is the execution of a Petri net. A transition is fired if each input place contains at least as many tokens as the weight of the arc from place to transition. Formally, t is enabled in marking M if for all places p:

$M(p) \geq I(p, t)$

Where I(p, t) is the weight of the arc from place p to transition t.

When an enabled transition t fires, it consumes tokens from its input places and produces tokens in its output places based on the weight of the corresponding arc. There are many ways for marking to be represented. For example, you may denote the initial marking as M_0 or just M. The next markings would be M_1 and M'. When you fire a transition, you get a new marking M'(p) = M(p) - I(t, p) + O(t, p) for any p in P. [8]

IV. Properties of a Petri Net

Reachability: The reachability of a Petri net is a basic idea and allows for a better understanding of the capabilities of the graph. It determines whether a particular marking can be reached from the initial marking according to the rules of the net. After the firing of an enabled transition, the token distribution changes which in turn changes the marking. If there exists a firing sequence that allows the Petri net to fully run from the initial marking to a marking M_n, marking $M_n$is said to be reachable from marking M_0. In real-world terms, it can show you whether all of your tasks will be able to be completed.

The set of all possible markings that are reachable from $m_0$is called the Reachability Set, denoted by the symbol $R(m_0)$.
We write $M_0[t>M_1$to depict reachability. It can also be depicted this way $M_0 \rightarrow M_1 \rightarrow M_2 \ldots$. M (the transition that allows one marking to reach the next must be listed on top of the arrow), or this way $\sigma = M_0 t_1 M_1 t_2 M_2 \ldots t_n M_n$

Boundedness: Boundedness is a key property in the analysis of Petri nets, as it indicates whether there is a limit or restriction to the number of tokens that can appear in each place in the Petri net. This property allows you to ensure that if a system has a maximum capacity, you do not exceed it which helps prevent overflow and resource exhaustion.

A Petri net is said to be *k*-bounded or simply bounded if a positive integer, *k* is the number that you don't want to exceed, exists, and is not exceeded by the number of tokens in any place *p*. If $k = 1$, the Petri net is called safe, meaning no place can have more than one token at any time.

Liveness: Liveness is another key property in the analysis of Petri nets. A Petri net (N, M_0) is live if, for every transition, there is a sequence that allows that transition to eventually fire. It ensures that the system can indefinitely operate without getting stuck in a deadlock state no matter what firing sequence takes place. Since full liveness can be unrealistic for most systems, there are different levels of liveness.

A transition *t* in a Petri net (N, M_0) is said to be dead (L0-live) if *it can* never be fired in any firing sequence.

L1-live(potentially fireable) if *t* can be fired at least one time.

L2-live if, given any positive integer *k*, *t* can be fired at least *k* times in some firing sequence. L3-live if *t* appears infinitely, often in some firing sequence in M_0.

L4-live or live if t is L1-live for every marking M.

Finally, a Petri net is said to be Lk-live if every transition in the net is Lk-live, k=0, 1, 2, 3, 4.

Dead-lock Freeness: A deadlock in a Petri net occurs when the system reaches a marking from which no transitions are enabled, making it stuck and causing no further actions to take place. For it to be deadlock-free, it must be able to reach all markings from the initial marking for at least one transition that is fired. To check for deadlock freeness, you can construct a reachability graph or examine the Petri net's structure.

Reversibility: A Petri net is reversible if any markings can return to the initial marking M_0. $\forall M \in R(m_0)$

V. Analysis Methods of Petri Nets

Coverability/Reachability graphs

The coverability of a Petri net helps to determine whether a certain marking can be reached from the initial marking. From the initial marking of a net, we can reach as many new markings as the number of enabled transitions, and from every new marking, we can reach more markings. According to Todao Murata's "Petri Nets: Properties, Analysis, and Applications", there are steps to construct a coverability tree. They include the following. Start with the initial marking M_0 as the root of the tree and name it "new". Expand upon the tree by firing transitions making sure to keep up with the new and old paths you get when firing by labeling them. If no transition is enabled name it "dead-end". If a marking is created that has more tokens in some places than seen previously in any of the markings, replace the extra tokens with ω and merge nodes that have this to prevent infinite growth.

VI. Applications

There are a plethora of things Petri nets can be applied to. The following sections detail a few of these.

Workflow management systems: Workflow management entails the controlling, monitoring, optimizing, and supporting of business processes to ensure efficiency, effectiveness, and adaptability. The main goal is to make sure the right things are executed by the right person at the right time. Before recently, no tool supported

workflow management causing struggles with modeling the complexity and concurrency that the processes had. Although it is possible to do workflow without a management system, they make it much easier. Petri nets have improved workflow management by providing a tool that can easily represent and analyze these processes. They support all the things needed to model a workflow process, as they can capture the dynamic behavior and dependencies of a task within the process. This allows for the detection and resolution of many issues that may be present such as deadlock. They also allow businesses to experiment with the flow of things to ensure they are placed in the preferred order that will allow for the completion of all necessary tasks. Overall, using Petri nets in workflow management leads to more efficient and scalable workflows that improve productivity.

Communication protocols: A communication protocol is a set of rules that allow systems to consistently receive and send messages. An example is hypertext transfer protocol (HTTP), which is the foundation of the World Wide Web and is used to load web pages using hypertext links. Another commonly known example is ethernet which is a common protocol used for communication between wired local area networks and devices. Petri nets allow protocols to be mapped out and show all the possible states and transitions which can help to identify issues such as deadlock where communications stop completely and livelock where processes cycle without progressing. They also can verify that data reaches its destination and prevent data corruption.

Manufacturing Systems: Manufacturing systems are networks or collections of processes and equipment that produce goods and services. Their flows must be carefully and precisely coordinated to make sure they are efficient and productive. Petri nets can model each step in a production line, along with how they might interact with each other, and different things that will happen based on the flow of the resources. They can resolve issues that could potentially slow down the system including resource conflicts where multiple processes might be trying to use the same resource.

VII. Methodology

The methodology I am following is structured to gain a deep understanding of a Petri net to gain the ability to model, validate, analyze, and implement the learned ideas into complex systems. Learning the necessary background information that has been detailed in the paragraphs before this one is crucial to understanding how a Petri net works. After that, we can begin modeling Petri nets that represent systems.

Modeling

When modeling a system, you first want to identify the elements of the system to relate them to which piece of the Petri net they will represent. Most likely, places will represent the conditions and transitions will represent events that change the conditions. Tokens will represent the state of the conditions, defining whether they are active or not. Depending on your system, you may need to assign weights to your arcs to indicate if multiple resources are consumed or produced during their transitions. Doing a consistency check is important to ensure the model has correct and meaningful connections and make sure it meets all the requirements of the system.

Analysis

After your model is complete, you can apply multiple analysis techniques that will show you the properties and behavior of the system. You can do a reachability analysis which determines all the states the system can reach from the initial state by constructing a reachability graph. To confirm the system will not enter a deadlock state, you should perform a liveness analysis by analyzing the reachability graph and checking for any markings where no transition is enabled.
This will tell you if, from any reachable marking, it is possible for any transition to fire eventually, allowing the system to run indefinitely if need be. You can test the system with the levels of liveness in mind and decide whether it has weak liveness or strong liveness. Making sure the number of tokens in each place does not exceed a specified limit with a boundedness analysis will help keep your system running smoothly and ensure the system doesn't require unbounded resources. You can do this also by checking the reachability graph and determining the maximum number of tokens across all reachable markings. If the number of tokens in p is less than or equal to k, then the Petri net is k-bounded.

Implementation

When you have completed all of your analyses, you can implement the insights gained from the model Petri net to build or optimize the real system. Then you want to test the system to see if it behaves how the model predicted it would.

Reporting

During your process, it is important to document all the steps you took when constructing your model and applying it to the real system in case something goes wrong and to make it easier to communicate your work and findings to others. You will want to have detailed markings from the Petri net model including all of the elements, the formation, and the initial marking. It is vital to document the results you get from your analysis techniques, as these tell you the most about your system. If you apply your findings to a real system, you may also want to document how you built or optimized your system based on your

model.

Case studies and Examples

Studying existing Petri nets and case studies that use Petri nets is important because it allows for a deeper understanding of them, as it shows their applications and basic structure. If you need an example that details how complex concepts can be represented accurately, analyzing a Petri net that has already been built and applied is a good idea. Case studies that use Petri nets emphasize how Petri nets can fix adversities encountered in various fields. They give you real-world examples of how Petri nets are applied.

VIII. Simulating and Coding Petri Nets

Many applications have been created over the years to aid in the simulation of Petri nets. Choosing the best one for you depends on what your needs may be. If you want to simply see a Petri net, you can use an online simulator such as Petri Net Simulator, developed by Igor Kim. It is very simple and easy to use. It also has a tutorial on the intro page if you need help. Though this is a good simulator for creating simple Petri nets, most Petri nets are more complex, therefore you may want to use one of the following simulators. Though there are lots of different tools, these, in my opinion, are the best ones to use.

PIPE(Platform Independent Petri Net Editor)

PIPE is a well-known, Java-based tool for constructing simple and complex Petri nets created in 2003 by postgraduate students in the Department of Computing at Imperial College London. It can be downloaded on any operating system that can also run Java. It supports various types of Petri nets including timed and stochastic making it versatile. It is user-friendly as it has drag-and-drop features and everything is easily accessible from the toolbar. It can perform detailed simulations and keep track of them with the animation history. The newest version, PIPE5, does not have all of the features of the older versions yet such as the ability to create reachability graphs and more analysis features. They do, however, give you the option to create and add your own modules. Despite this, there are still many useful features. For example, it has a feature that adds weights to the Petri net, which is surprisingly not a feature many of the simulators have.

CPN TOOLS/IDE

CPN Tools, now CPN IDE, originally developed by the CPN Group at Aarhus University from 2000 to 2010 is used mostly for the creation of colored Petri nets but can also create many other kinds. It can only run on Windows currently and requires Java. It also

includes drag-and-drop features and many simulation controls. Creating Petri nets using CPN IDE is extremely easy since you can form full nets all within the simulation box instead of having to use a toolbar. It can also create reachability and coverability graphs. There is a feature that allows you to examine the amount of resources you would use based on your model which is perfect for systems that have limited resources. There are also several other useful features.

SNAKES(Net Algebra Kit for Editors and Simulators)

Snakes, mainly created by Franck Pommereau, is a Python library that can be downloaded in your choice of IDE which provides all the necessary code to create and execute different kinds of Petri nets. It's a great application for researchers since it allows you to create prototypes of new ideas. It allows you to build basic, timed, and colored Petri nets. An important feature it has is the extensibility of the plug-in feature which allows users to change and extend the features to work with different kinds of Petri nets. One that is very useful is Graph Viz, which draws the Petri net that you code.

X. Applied Extension: Penelope, the Petri Net Chatbot

While the first portion of this report thoroughly explores the structure, properties, and practical application of Petri nets, this next phase applies that foundational knowledge in a modern, interactive form. To expand accessibility and understanding for learners at all levels, I am developing a chatbot named Penelope. Penelope is a Python-based educational tool designed to provide natural-language explanations of Petri net concepts. This section outlines the motivation, some background information, structure, implementation, and evaluation of the chatbot.

XI. Motivation and Purpose

Although Petri nets are powerful modeling tools, their mathematical foundations and visual representations can be difficult for new learners to understand. Textbooks and lectures are not always interactive, and learners may have unanswered questions. Penelope was created to solve that problem by acting as a virtual tutor capable of answering Petri-net related questions in plain, understandable language. The chatbot leverages natural language processing to explain definitions such as places, transitions, tokens, arcs, and marking, as well as concepts like deadlock, reachability, and boundedness.

XII. Chatbots

A chatbot is a computer program designed to simulate conversations with human users. They use natural language processing and machine learning to understand and respond to queries in a conversational manner. There are many different kinds of chatbots, including, rule-based, AI-powered, voice chatbots, keyword recognition-based chatbots, etc.. Some

of the earliest chatbots, created between 1966 and 2000, were essentially interactive FAQ programs, which relied on a limited set of common questions with pre-written answers. These FAQs generally required selection from simple keywords and phrases to make the conversation move forward. Advancements made in the 2020's have improved Natural Language Understanding, which is a key component of AI chatbots that allows them to better interpret human language. They also leverage context-aware algorithms that allows them to not only understand the words but also the context and intent behind the words. This allows for a smoother user-experience because the conversations are more fluid and natural. Chatbots have come an extremely long way, going from simple rule based bots to generative bots that can answer almost any question the user has. They are still being improved on and changing every day.

XIII. System Overview

Penelope consists of three main components:

1. **Knowledge Module**
   A structured dictionary of definitions, created using content from the first part of this report, was embedded directly into the chatbots code. These include formal definitions and simplified explanations tailored to student friendly phrasing.
2. **Language Understanding**
   Penelope operates through an online platform, which uses the OpenAI GPT-3.5 language model to generate more flexible, conversational responses. GPT is prompted with the same knowledge module to ensure the chatbot's answers stay aligned with the definitions used in this report.
3. **User Interface**
   The chatbot interface is built using Streamlit, a python library that allows rapid deployment of web-based tools. The interface is clean, with a chat window and styled responses using institutional colors from Norfolk State University. User input is captured, processed, and responded to in real time.

XIV. Implementation Details

Penelope was written in Python 3.10. The chatbot was modularized for ease of testing and customization. The logic responsible for response generation resides in a file called responses.py. For online use, the model compares user input to stored questions, formats the user input as a prompt, and sends it to the OpenAI API, with my Petri net definitions included to guide the model's output.

The Streamlit interface (streamlitApp.py) displays user and bot messages as chat bubbles, captures input with st.chatInput(), and includes custom HTML, CSS, and JavaScript to control layout. Users can enter questions such as "What is a transition?" or "How does firing work?" and receive accurate, understandable responses.

XV. Testing and Results

Testing

Since Penelope is still in development, formal user testing has not been conducted. However thorough internal testing was performed during development to verify functionality, refine responses, and improve usability. This testing included repeated manual interactions with Penelope, using a variety of inputs to simulate how users will actually engage with her.

Preliminary testing focused on Penelope's ability to:

-Correctly identify the user's intended questions even when phrased differently
-Provide accurate definitions from the knowledge base
-Remain user-friendly and accessible
-Use context and history to answer a users follow-up questions

I tested her by inputting things like direct Petri net questions, casual or vague prompts, follow-up questions and conversational context, spelling errors, and fragmented phrases.

Early Issues

-Responses to vague inputs like "okay" were information dumps.
-Penelope over explained in cases where a shorter answer was more appropriate
-Lacking context and memory of chat history
-Messages appeared in bulky paragraphs, making them harder to read
-Answers questions that don't pertain to Petri nets

Improvements made

-Introduced logic to detect short or unclear inputs and respond accordingly
-Reformatted messages for better readability using line breaks
-Styled chat bubbles and interface with CSS to improve visual clarity
-Refined content handling so Penelope now responds more naturally, using simpler phrases when needed
-Only answers questions about Petri nets, and politely directs users back to Petri nets if they get off topic

Penelope now performs more effectively as an educational chatbot. She responds to a variety of question types, handles ambiguity, and communicates Petri net concepts with clarity and consistency. These improvements have positioned her as a promising learning tool with strong potential for classroom or self-guided use once formal testing is conducted.

1. C.A. Petri, “Kommunikation mit Automaten,” Technical Report, University of Bonn, 1962.
2. S. Budko and W. Leister, “Treatment Pathways as Petri Nets in Patient Workflow Management,” Proceedings of the 2017 International Conference on Health Informatics and Medical Systems, 2017.
3. W.M.P Van der Aalst, “The Application of Petri Nets to Workflow Management”, Journal of Circuits, Systems, and Computers, vol.8, no. 1, pp. 21-66, 1988.
4. Chang-Pin Lin and Hung-Lin Dai, “Applying Petri Nets on Project Management,” Proceedings of the 2004 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1428-1433, 2004.
5. Todao Murata, “Petri Nets: Properties, Analysis and Applications,” Proceedings of the IEEE, vol. 77, no. 4, pp. 541-580, 1989.
6. L. Recalde, M. Silva, J. Expleta, and E. Teruel, “Petri Nets and Manufacturing Systems: An Examples-Driven Tour,” in Applications and Theory of Petri Nets, Lecture Notes in Computer Science, vol.. 1769, Springer, 2000, pp. 64-88.
7. W.Reisig,” Understanding Petri Nets,” Springer, 2013.
8. J. Wang,” Petri Nets for Dynamic Event-Driven System Modeling”, Monmouth University, 2007.
9. Halder, A.” A Study of Petri Nets Modeling, Analysis and Simulation”, Department of Aerospace Engineering Indian Institute of Technology Kharagpur, 2006.
10. Pommereau, Franck. SNAKES: A flexible high-level Petri nets library. 9115. (2015).

11. Westergaard, M., Kristensen, L.M. The Access/CPN Framework: A Tool for Interacting with the CPN Tools Simulator. In: Franceschinis, G., Wolf, K. (eds) Applications and Theory of Petri Nets. PETRI NETS 2009. Lecture Notes in Computer Science, vol 5606. Springer, Berlin, Heidelberg.(2009)
12. N.J. Dingle, W.J. Knottenbelt and T. Suto. PIPE2: A Tool for the Performance Evaluation of Generalised Stochastic Petri Nets (PDF format). ACM SIGMETRICS Performance Evaluation Review (Special Issue on Tools for Computer Performance Modelling and Reliability Analysis), Vol. 36(4), March 2009, pp. 34-39.
13. P. Bonet, C.M. Llado, R. Puijaner and W.J. Knottenbelt. PIPE v2.5: A Petri Net Tool for Performance Modelling (PDF format). Proc. 23rd Latin American Conference on Informatics (CLEI 2007), San Jose, Costa Rica, October 2007.