

Chapter *17*

Petri Net Analysis

17.1 INTRODUCTION

Petri net analysis (PNA) is an analysis technique for identifying hazards dealing with timing, state transitions, sequencing, and repair. PNA consists of drawing graphical Petri net (PN) diagrams and analyzing these diagrams to locate and understand design problems.

Models of system performance, dependability, and reliability can be developed using PN models. PNA is very useful for analyzing properties such as reachability, recoverability, deadlock, and fault tolerance. The biggest advantage of Petri nets, however, is that they can link hardware, software, and human elements in the system.

The PNA technique may be used to evaluate safety critical behavior of control system software. In this situation the system design and its control software is expressed as a timed PN. A subset of the PN states are designated as possible unsafe states. The PN is augmented with the conditions under which those states are unsafe. A PN reachability graph will then determine if those states can be reached during the software execution.

17.2 BACKGROUND

The PNA technique falls under the system design hazard analysis type (SD-HAT) and should be used as a supplement to the SD-HAT analysis. Refer to Chapter 3 for a description of the analysis types. The purpose of the PNA is to provide a technique to graphically model systems components at a wide range of abstraction levels

in order to resolve system reliability, safety, and dependency issues. The graphical model can then be translated into a mathematical model for probability calculations.

Petri nets can be used to model a system, subsystem, or a group of components. PNA can be used to model hardware or software operation or combinations thereof. To date, the application of PNA for system safety use has been limited to the examination of software control systems. Its use has rarely been applied to large systems. PNA can be used to develop reliability models of system operation.

The significant advantage of the technique is that it can be applied to a system very early in development and thereby identify timing issues that may effect safety early in the design process. PNA application helps system developer's design in safety and system quality during early development, eliminating the need to take corrective action after a test failure or mishap.

The PNA method is somewhat difficult to learn, understand, and master. A graduate-level understanding of mathematics and computer science is needed for the application of PNA. The analyst must master the technique and have a detailed knowledge of the process being modeled. The PNA technique is suited for use by theoretical mathematicians. The PN model quickly becomes large and unwieldy as system size increases and is therefore usually only used on small system applications.

The use of PNA is not widespread in the system safety discipline because of its difficulty to use, its limitation to smaller problems, and its limitation in scope to timing-type problems. PNA is highly specialized and only assists in a certain niche of potential safety concerns dealing with timing and state reachability. The technique is only recommended for special design safety concerns.

17.3 HISTORY

The concept of Petri nets has its origin in Carl Adam Petri's doctoral dissertation *Kommunikation mit Automaten* submitted in 1962 to the faculty of Mathematics and Physics at the Technische Universität Darmstadt, Germany. His thesis developed this graph-based tool for modeling the dynamics of systems incorporating switching. Subsequent research by many individuals has provided the means for using Petri's concepts as the basis for system modeling in many different applications, such as reliability, dependency, safety, and business models.

17.4 DEFINITIONS

In order to facilitate a better understanding of PNA, some definitions for specific terms are in order. The following are basic PNA terms:

Transition Represents a system event that must occur. Transitions contain a switching delay time. When all of the inputs to the transition have a token, the transition event is *enabled*, and it occurs or *switches* after the given delay time. The delay time represents the actual system operational design. A delay time

is placed in each transition node. Immediate transitions have a delay time equal to zero ($D = 0$). When the transition node *switches* or *fires*, all input nodes lose their token, and all output nodes receive a token.

Place Used to represent the input and output nodes to a transition. Places contain the tokens.

Token Represents timing in the system logic. As a PN model develops, the tokens build sequencing into the model. Tokens are analogous to currency; they are used for transactions.

Connecting edge Connects the places and transitions together to build the logical model.

State Static condition (or state) of the PN model before and after the firing of a transition. A PN model will have a finite number of states, based on the model design.

Reachability System can have many different possible states; reachability refers to the systems capability to reach any or all of those states during operation. As designed, the system may not be able to reach some states.

Repair Refers to the capability to physically repair a failed component and restore it to an operational state.

17.5 THEORY

The PNA method utilizes a diagram or directed graph that portrays in a single diagram the operational states of the system. The state diagram is flexible in that it can serve equally well for a subsystem or an entire system. The diagram provides for representation of system states, transitions between states, and timing. Figure 17.1 illustrates the overall PNA process.

17.6 METHODOLOGY

A Petri net is a graphical and mathematical modeling tool. It consists of places, transitions, and arcs that connect them. Input arcs connect places with transitions, while

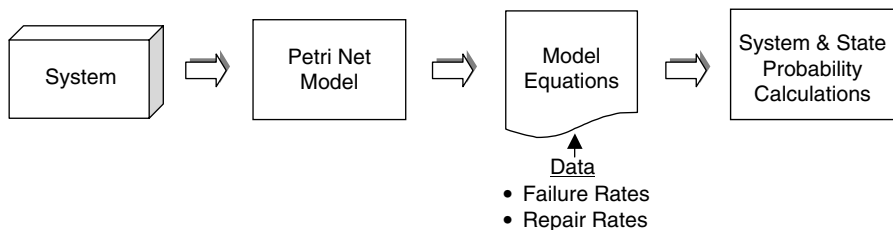


Figure 17.1 PNA process.

output arcs start at a transition and end at a place. There are other types of arcs, for example, inhibitor arcs. Places can contain tokens; the current state of the modeled system (the marking) is given by the number (and type if the tokens are distinguishable) of tokens in each place. Transitions are active components. They model activities that can occur (the transition fires), thus changing the state of the system (the marking of the Petri net). Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled (there are enough tokens available in the input places). When the transition fires, it removes tokens from its input places and adds some at all of its output places. The number of tokens removed/added depends on the cardinality of each arc. The interactive firing of transitions in subsequent markings is called *token game*.

Figure 17.2 shows the symbols utilized in comprising a PN model. All PN models can be constructed with just these components and using the firing rules listed below. A PN is a bipartite directed graph (digraph). It consists of two types of nodes: places (drawn as circles), which can be marked with tokens (drawn as a dot), and transitions (drawn as squares or bars), which are marked by the time, D , it takes to delay the output of tokens. If $D = 0$, the transition time is immediate; otherwise, it is timed. PNs dealing with transition times are often referred to as timed PNs. Timed Petri nets are models that consider timing issues in the sequencing. The practice of integrating timed Petri nets with software fault tree analysis has recently become popular.

The movement of tokens is governed by the *firing rules* as follows:

1. A Transition is enabled when all of the places with edges pointing to it are marked with a token.
2. After a delay $D \geq 0$ the transition switches or fires.
3. When the transition fires, it removes the token from each of its input places and adds the token to each of its output places.
4. A place can have multiple tokens.
5. The number of tokens in a PN is not necessarily constant.

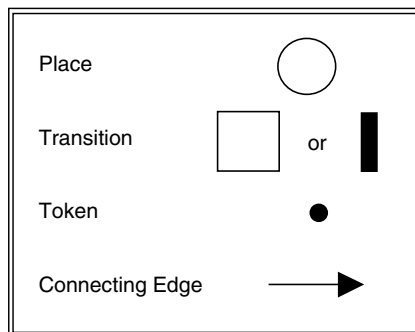


Figure 17.2 PN model symbols.

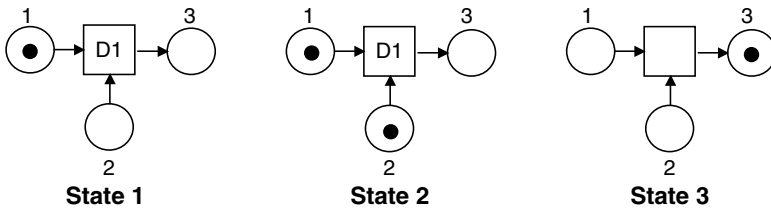


Figure 17.3 Example PN model with three transition states.

6. Tokens move along the edges at infinite speed.
7. The transition time D can be either random or deterministic.

The PNA method has a static part and a dynamic part. The static part consists of the places, transitions, and edges. The dynamic part involves the marking of places with the tokens when the transition firing occurs. In the PN model, places represent events that correspond to discrete system states. The transitions represent logic gates. The marking of a PN model at a given moment represents the state at that moment in time.

Figure 17.3 shows an example PN model with three transition states. In state 1, place 1 has a token but place 2 does not. Nothing can happen until place 2 receives a token. In state 2, place 2 receives a token. Now transition D1 has both inputs fulfilled, so after delay D1 it fires. State 3 shows the final transition, whereby D1 has fired, it has removed the two input tokens (places 1 and 2) and given an output token to place 3.

Figure 17.4 shows another example PN model with two transition states. This example shows how places can have multiple tokens, and how tokens are added and subtracted depending on the model. In this example, transition D1 has tokens for all input places in state 1, so it therefore fires and transitions into state 2. State 2 shows that each input place loses a token and each output place receives a token. Multiple tokens can be used for modeling counting loops or redundant components in a

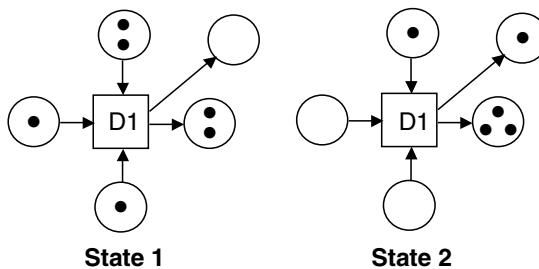


Figure 17.4 Example PN model with multiple tokens.

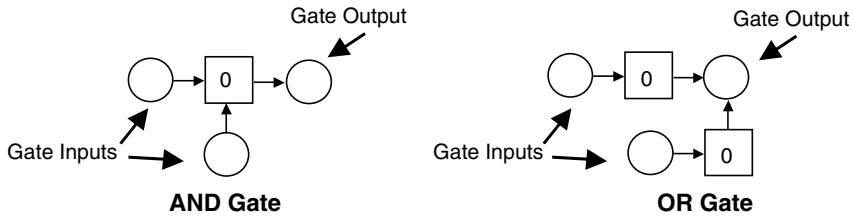


Figure 17.5 PN model of AND gate and OR gate.

system design. Transition D1 cannot fire again because all of its inputs do not have tokens now.

Figure 17.5 shows how AND gates and OR gates are modeled via PN models. Note that the transitions have a delay time of zero and are therefore immediate.

A powerful feature of PNs is their corresponding state graphs, otherwise referred to as reachability graphs (RG). Reachability graphs show all of the possible system states that can be reached by a PN model. Knowing which states can be reached and which cannot is valuable information for reliability and safety evaluations. For example, it is important to know if a system can reach a suspected high-risk state.

Figure 17.6 shows an example PN model along with its corresponding reachability graph. In this RG state 1 is the initial state, with places 1, 2, and 4 each having a token. State 2 shows the result of transition D1. State 3 shows the result when transition time $D2 < D3$, while state 4 shows the result when transition time $D2 > D3$.

With five places in this model, one might assume that with each place having a binary value of 0 or 1 that there would be $2^5 = 32$ possible states. But, as shown by the reachability graph only four states are actually reachable.

Figure 17.7 shows a PN model for a system design with two redundant components with repair. This system is comprised of two redundant components operating simultaneously. Successful operation requires that only one component remain functional. When a component fails, it undergoes repair. If the second component fails while the first is being repaired, then the system is failed. Note the special PN terminology in this PN model. L1 indicates life operation until random failure of component 1, while R1 indicates component 1 during repair duration. The /2 notation indicates that two tokens are required for this transition to occur.

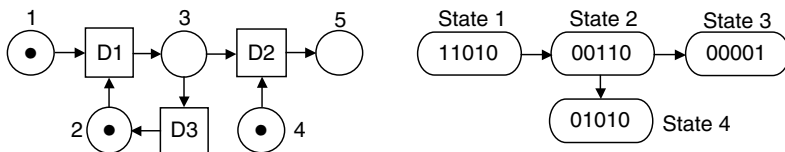


Figure 17.6 PN with reachability graph.

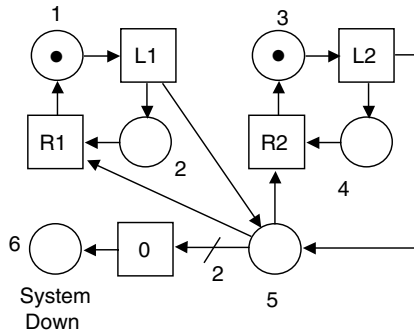


Figure 17.7 PN of two-component system with repair.

Figure 17.8 shows the RG for the PN model in Figure 17.7. This type of RG is referred to as a cyclic RG with an absorbing state. Note that with two components in this system, four possible states are expected, but in this case the eventual marking of place 6 defines a fifth state. Note also, that since place 5 requires two tokens for switching these state numbers are nonbinary.

The PNA method may be used to evaluate safety critical behavior of control system software. In this situation the system design and its control software is expressed as a timed PN. A subset of the PN states are designated as possible unsafe states. The PN is augmented with the conditions under which those states are unsafe. A PN reachability graph will then determine if those states can be reached. If the unsafe cannot be reached, then the system design has been proven to not have that particular safety problem.

17.7 EXAMPLES

The hypothetical missile fire control system shown in Figure 17.9 will be used to demonstrate a PN model. System operation is as follows:

1. The operator presses the arm button to initiate arming and firing process.

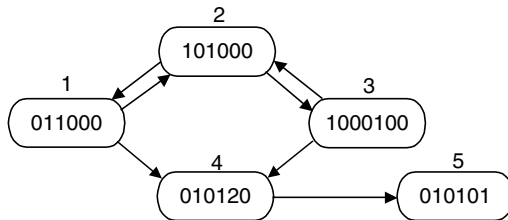


Figure 17.8 RG for two-component system with repair.

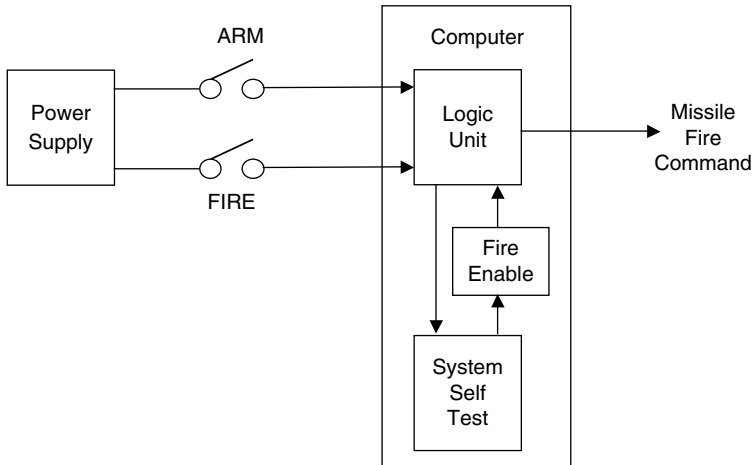


Figure 17.9 Example missile fire sequence system.

2. When the computer receives arm button signal, the computer initiates a system self-test to determine if the system is in a safe state and all arming and firing parameters are met.
3. If the self-test is passed, the computer generates a fire signal when the fire button is depressed; if the self-test does not pass, then firing is prohibited by the computer.

Figure 17.10 shows how a PN model might be applied to analyze an actual weapon system design.

The PN model for the missile fire sequence reveals the following safety information:

1. A valid fire signal requires three interlocks: arm command, fire command, and fire enable signal.

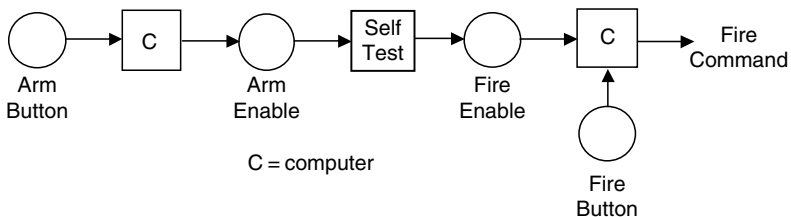


Figure 17.10 PN for missile fire sequence system.

2. The computer is a single-point item that processes all three interlocks and, therefore, is susceptible to single-point failures, both in hardware and software.

17.8 ADVANTAGES AND DISADVANTAGES

The following are advantages of the PNA technique:

1. PNs can be used to model an entire system, subsystem, or system components at a wide range of abstraction levels, from conceptual to detailed design.
2. When a PN model has been developed for analysis of a particular abstraction, its mathematical representation can support automation of the major portions of the analysis.
3. PNA is a good tool for modeling and understanding system operation.

The following are disadvantages of the PNA technique:

1. PNA only identifies system hazards dealing with timing and state change issues.
2. PNA is a limited hazard analysis tool because it does not identify root causes.
3. PNA requires an analyst experienced in PN graphical modeling.
4. PNA models quickly become large and complex; thus, it is more suitable to small systems or high-level system abstractions.

17.9 COMMON MISTAKES TO AVOID

When first learning how to perform a PNA, it is commonplace to commit some traditional errors. The following is a list of typical errors made during the conduct of a PNA:

1. Not obtaining the necessary training.
2. Using the complex PNA technique when a simpler technique might be more appropriate.

17.10 SUMMARY

This chapter discussed the PNA technique. The following are basic principles that help summarize the discussion in this chapter:

1. PNA models the timing and sequencing operation of the system.
2. PNA is a tool for identifying a special class of hazards, such as those dealing with timing, state transitions, and repair.

3. PNA provides both a graphical and mathematical model.
4. PNA only requires places, tokens, transitions, and connecting edges to model a system.
5. PNA can easily become too large in size for understanding, unless the system model is simplified.
6. For system safety applications, PNA is not a general-purpose hazard analysis tool and should only be used in situations to evaluate suspected timing, state transition, sequencing, and repair hazards.

BIBLIOGRAPHY

- Agerwala, T., Putting Petri Nets to Work, *IEEE Computer*, Dec., 85–94 (1979).
- Malhotra, M. and K. Trevedi, Dependability Modeling Using Petri Nets, *IEEE Trans. Reliability*, **44**:428–440 (1995).
- Petri Nets: Properties, Analysis and Applications, *Proc. IEEE*, **77**:541–580 (1989).
- Schneeweiss, W. G., *Petri Nets for Reliability Modeling*, LiLoLe, 1999.
- Schneeweiss, W. G., Tutorial: Petri Nets as a Graphical Description Medium for Many Reliability Scenarios, *IEEE Trans. Reliability*, **50**(2):June, 159–164 (2001).