WIRELESS MESH NETWORK THROUGHPUT ANALYSIS USING PETRI NETS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Lütfü Melih Buğra OĞUZER

July 2022
IZMIR

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude and special thanks to my supervisors Assoc. Prof. Dr. Tugkan Tuglular and Assoc. Prof. Dr. Fevzi Belli, for their guidance, patience and support, as well as their understanding. I appreciate their time and effort in helping the writing process of this thesis and greatly value their assistance. A special thanks to my family, my future wife Nagihan Akan, and my friends, for their ongoing love, support, and for reminding me of my dreams when I felt like giving up.

ABSTRACT

WIRELESS MESH NETWORK THROUGHPUT ANALYSIS USING PETRI NETS

Evolving technology has made the understanding of quality perception in software processes more difficult. Unlike other sectors, rapid adaptation and software development processes have become a critical issue. This issue can especially be observed in the service, telecommunication, and high technology sectors. User demands and competition are quite high and with this competition, the need to subject the customized or developed software to rapid testing processes has formed.

Undoubtedly, this process implies a great responsibility for the "quality assurance" teams. This responsibility has reached a level that can only be handled by the quality assurance departments that automate the testing cycles. However, it is also important that these cycles are very efficient.

Our research is concerned with modeling test processes with Petri nets and creating test scenarios based on this modeling to make automation processes in the telecommunications industry more efficient. In this research, the performance analysis of wireless mesh networks is executed through place/transition petri-net modeling. Through this modeling, reusable test scenarios which were compared and analyzed with traditional automation processes were created for performance tests.

The research also addresses another topic which is the shortening of the modeling processes created with Petri nets and how to make them more efficient. In this context, a tool has been developed in order to shorten the modeling process and analyze the reusable test scenarios.

Finally, ten test engineers were interviewed about reusable test processes. In these interviews, feedback was provided on reusable test scenarios in test automation processes.

ÖZET

PETRİ AĞLARINI KULLANARAK KABLOSUZ AĞ VERİ AKIŞ ANALİZİ

Gelişen teknoloji ile yazılım süreçlerindeki kalite algılısını algılayabilmek her geçen gün daha da güçleşmektedir. Diğer sektörlerden farklı olarak özellikle hizmet, telekomünikasyon ve yüksek teknoloji içeren sektörlerde hızlı adaptasyon ve yazılım geliştirme süreçleri kritik bir husus haline gelmiştir. Kullanıcı isterleri ve rekabet çok fazladır. Bu rekabet ile birlikte, adapte edilen veya geliştirilen yazılımların çok hızlı bir test sürecine tabi tutulması gerekliliği ortaya çıkmıştır. Bu süreç hem hızlı bir şekilde ilerlemeli hem de efektif olmalıdır.

Hiç kuşkusuz bu süreç, “Kalite Güvence” takımlarına çok fazla sorumluluk yüklemektedir. Bu sorumluluklar, sadece kalite güvence departmanlarının test süreçlerini otomatize etmesi ile karşılanabilecek bir boyuta ulaşmıştır. Ancak gelinen noktada bu süreçlerin de verimliliği yüksek olmalıdır.

Araştırmamız, telekomünikasyon sektöründeki otomasyon süreçlerinin daha verimli bir hale getirilmesi için petri-ağları ile test süreçlerinin modellenmesi ve bu modelleme üzerinden test senaryolarının oluşturulmasını konu almaktadır. Araştırma kapsamında kablosuz örgü ağlarının performans analizi petri-ağı modellemesi ile yapılmaktadır. Bu modelleme üzerinden tekrar kullanabilir test senaryoları oluşturulmuştur. Bu senaryolar geleneksel otomasyon süreçleri ile karşılaştırılıp analiz edilmiştir.

Araştırmanın değindiği bir diğer konu ise, Petri-Ağları ile oluşturulan modelleme süreçlerinin kısaltılıp daha verimli bir hale getirilmesidir. Bu kapsamda modelleme sürecini hızlandıran bir araç geliştirilmiştir. Bu araç ile kısaltılan modelleme süreci ve tekrar kullanılabilir test senaryoları yine aynı şekilde analiz edilmiştir.

Son olarak, tekrar kullanılabilir test süreçleri ile ilgili olarak on test mühendisi ile görüşme yapılmıştır. Bu görüşmelerde test otomasyon süreçlerindeki model bazlı test süreçleri ile tekrar kullanılabilir test senaryoları üzerine görüşler alınmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

PN	Petri-NET
MISTA	Model-based integration and system test automation
QA	Quality assurance
AP	Access point
GW	Gateway
Topology	Repository of all devices used during testing
DUT	Device under test
Wi-Fi	Wireless fidelity
WLAN	Wireless local area network
IOT	Internet of things
STB	Set top box
MBT	Model based testing
IPERF	Network performance tool
Uplink	Data transfer from station to access point
Downlink	Data transfer from access point to station
TCP	Transmission control protocol
UDP	User datagram protocol
RSSI	Received signal strength indication
FSM	Finite-state machine

CHAPTER 1

INTRODUCTION

A software product's life cycle is composed of conception, design, development, testing, and maintenance to a product (Sommerville, 2016). This process becomes a continuous cycle with maintenance and new integrations. This cycle continues until the product dies completely. At this point, "Testing" is of critical importance for the product throughout the entire cycle. However, testing is one of the most expensive and time-consuming processes in the software development lifecycle, accounting for 40% to 70% of all system development costs. (Devi, 2012).

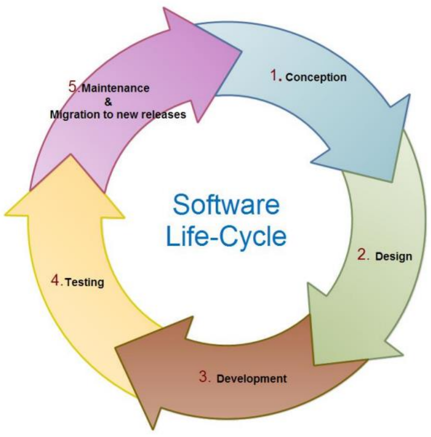


Figure 1.1. Software product life cycle

The term "Quality" has generated for institutions and companies that can perceive the criticality of the testing process and its costs. Product upgrade cycles have shortened and support times have increased mainly because of the lifecycle and performance requirements of the companies working on the software.

In addition to these responsibilities, companies are expected to have a very short time to adapt to new technologies. Perhaps one of the most important sectors facing this situation is the "telecommunications sector". These companies are expected to compete and adapt very quickly (Lopez & Viela, 2012). This expectation especially influences the devices owned by home users. In addition to needing bandwidth with evolving technology, these products are also expected to be secure and stable. These expectations that users have of Internet service providers also encumber responsibilities on companies that manufacture gateways, access points, and other wireless technology products for home use. At this point, manufacturers invest in test automation departments to increase the quality and efficiency of QA processes, which is one of the cornerstones of these processes.

However, even if these processes are automated, it is very important that they prove to be efficient. Minimizing human errors in automation processes and reducing human needs are some of the most important factors affecting efficiency.

At this point, our research proposes a reusable Petri net model structure that can minimize human errors in test automation processes and increase efficiency. For this idea, the test process for the wireless steering was modeled via Place/transition PN. With the created model, test cases were generated and compared with the requirement-based testing and experience-based testing automation processes. At the same time, the research also created a tool that generates an automatic PN model with the given inputs to increase efficiency.

1.1. Motivation

Given the speed of today's wireless technologies, it is clear that the only way to keep up with that speed is to accelerate the QA cycle. Accelerating these cycles is possible by integrating the process with automation. However, due to the large financial burden of setting up an automation system in the Wi-Fi area, the efficiency of this process should also be increased. In this context, it is very important to increase automation efficiency and reduce the cost of the process. For this process, it is necessary to gain different perspectives. The data to be obtained as a result of the analysis of this structure will undoubtedly increase the efficiency of the process. Modeling techniques, perhaps the most popular of these analysis options, can both improve the process and identify the missing points in the process. In line with these considerations, we think that the analysis of a steering test process to be modeled with Petri-Nets will give us many new perspectives.

In addition, we also think that the automation process should be dynamic. Undoubtedly, if this structure is static, some problems will arise. It is the detection of critical test bugs at the result end of the test automation processes that proceed with the static requirement-based testing or experience-based testing processes. However, at the end of the processes, the existence of undiscovered bugs is an undeniable fact. It is evident that the same testing process will repeatedly bare the same results.

Even if standard manual testing and automation scripts reduce the level of bugs by a certain amount, the occurrence of the "pesticide paradox" would prove inevitable since the same things are examined in the tests. This situation may affect the quality of the product.

Definition Pesticide Paradox: Each method you use to prevent or find bugs leaves a residue of subtler bugs against which those methods are inactive (Beizer, 1990).

In organizations with extensive QA cycles, test cases with discrete events further complicate the cycles. This difficulty in the cycles also reduces the coverage rate of the test cases.

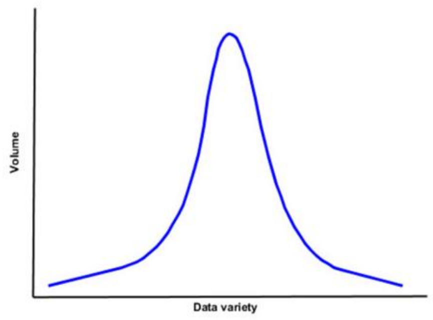


Figure 1.2. Pesticide Paradox Graph

In Figure 1.2. it can be seen that the bug detection rate of the tests decreased after a certain time. In our research, we introduce our model-based testing via Petri nets to improve this process and accelerate and differentiate the test automation process. This research also includes the analysis of the testing process and the improvement of the testing processes' scope. The analysis and development in this context will increase operational quality and efficiency.

1.2. Major Contributions of the Thesis

This paper aims to develop a petri-net based approach to test generation for enterprises and addresses the following questions:

1. How can the discrete events that occur in automation test cycles be better structured?
2. Which petri-net type is appropriate?
3. How can a created Petri-Nets Modeling be reused?
4. How can the learning costs of Petri nets be reduced?
5. How efficient is the MISTA test generator system?
6. What do testers think of the structure we created as part of the research?

In order to answer the first question, we can say that many discrete events in these tests have been found while studying end-to-end testing in automation process. As part of the research, the "Steering Performance" suite, which has many discrete events in the test automation processes within the "Mesh Wireless", was selected as the pilot suite. During the modeling phase, the process was analyzed and the necessary updates were added in accordance with the chains of opportunities. Petri nets were used for model-based testing and analyzed using the "incidence matrix," one of the Petri net analysis tools. As a result of the analysis, the critical discrete events were prioritized and detected that in test scenarios some of these events may be executed more than once.

A tool called “EIN” was developed as part of the research in order to increase the reuse of the testing process modeled with the Petri net for different devices. It is possible to create new sub-test scenarios from the model once created according to the desired configuration types with this tool. This ensures that tests intended for different purposes or configurations can be used effortlessly and answers our second question.

To answer the third question, it was found during the research that Petri nets are used differently in different areas. It was also determined that it is used less than other modeling tools. In order to be able to use different domains and to reduce the need for users to know how to model Petri nets, an infrastructure was created in "EIN Tool" to meet users' needs for test automation in a simple way and without modeling knowledge.

To answer the fourth question, the study also analyzed the test-generating function of MISTA. For this analysis, an alternative to MISTA's test generator was created. The test case scenarios generated with all configuration and product options were manipulated in the file and compared individually with the modeling. As a result of this comparison, it was found that there were duplicate results, especially when generating tests from models with high combination.

The negative and positive aspects of using the created model and tool were analyzed which answers the fifth question. In this context, the feedback of the companies QA department employees was obtained. The feedback is based on a survey and was created with a total of 10 test engineers and technicians. In the future work, necessary updates will be made based on this feedback.

1.3. Outline of Thesis

This thesis is organized as follows; the next chapter provides a review of the literature, chapter 3 provides background information on wireless mesh networks, Petri nets, and MISTA while chapter 4 explains our predictive approaches. In chapter 5, we explain the dataset and the experiments we conducted. Finally, chapter 6 provides concluding remarks and an outlook on future research.

CHAPTER 2

RELATED WORK

Model-based testing is a well-known common testing approach. In this approach, test cases are developed using an extracted model of a system or device under test (Utting, 2006). Research has shown that many different approaches and different modeling techniques are used in analyzing or testing. In this context, similar studies have been identified and made.

One of the most important studies on the performance evaluation of 802.11 was done by Bianchi (Bianchi, 2000). Bianchi presented a simple analytical Markov model to calculate the saturation throughput performance of 802.11. This research is not about a test generator, but about comparing the throughput obtained by modeling with the actual throughput.

Hessel and Petterson (Hessel & Petterson, 2008) suggested a case study where a model-based approach to black-box testing is applied to verify that a Wireless Application Protocol (WAP) gateway conforms to its specification. The WAP gateway used in the research was developed by Ericsson. In the research, which used "timed automata" as a modeling technique, the tool that generates the tests takes this model as input. Even if the research was successful, Petri nets, are more suitable for modeling discrete systems than automata. In addition, petri-nets have precise mathematical definitions. Also, there are many analysis methods related to the implementation of the Petri net. (Unal, 2008)

Fragal et al, have used FSM with constraints to model web application behavior using forward engineering. Higher level tests sequences are formed by combining test sequences from lower-level FSMs. Also, a test reuse strategy to minimize the concretization effort has been proposed. The results showed that the reusability of concretized test cases not only provided a reduction in effort but also reducing the cost of testing. (Nouman Zafar et al, 2021)

Another research paper belongs to Thummala et al, which developed a technique to use Petri-Nets to model web applications for testing purposes. This technique automatically generates test cases from the model. However, on the scope of the research, there has not been any suggestions regarding the reuse of the model produced with petri-nets. Our research is differentiated by a system that automatically generates models for simple events without requiring modeling knowledge of the automation process. Thummala et al tried to make possible the effortless reuse of the modeled structures with this development.

Another research paper belongs to Sergio et al and in this research, conflicting interactions between pilot/autopilot in aviation were analyzed with petri-nets. With these analyzes, interactions that may cause deadlocks in the system were examined. (Sergio et al, 2014)

Anbunathan and Basu suggested a test case generated based on over the UML sequence diagram. The experimental evaluation showed a significant improvement in bug detection along with a better effect in terms of test coverage when compared with other methods. (Nouman Zafar et al, 2021)

CHAPTER 3

FUNDAMENTALS

3.1. Wireless Mesh Network Automation Test

Wi-Fi test automation systems are more complex and difficult to understand than other automation systems. The main reason for this difficulty is that the system works with radio waves.

3.1.1. Environment

Wi-Fi test automation systems work with radio waves and therefore these systems surpass any other automation system when it comes to complexity and difficulty. This situation may cause the system to behave erratically from time to time in performance measurements. In addition, other subsystems that interact with the system are also extremely complex. Mobile devices, IOT devices, STB devices and more connecting to the fixed wireless network. It is becoming increasingly difficult to test this complex environment, which is increasing by the day.

In the mesh network, the situation becomes much more complex. The reason is the increased probability of bugs caused by more than one source making the same broadcast. Finding out the which device with which settings and at what distance was the one where the problem was detected is extremely complicated.

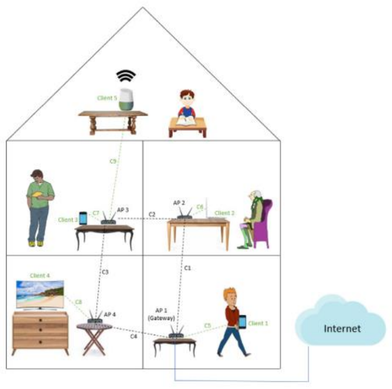


Figure 3.1. An example mesh network
(Source: Kuskonmaz et al, 2011)

Figure 3.1 illustrates how the system works. As it can be seen, large areas may be required to test such systems. Because of this situation, it is expected that this can be done quickly and dynamically, even if the required environment is set up.

3.1.2. Device Under Test

During the study, 3 different gateways and 2 different access points were used. In total, 3 different Wi-Fi generations were used. Each gateway is paired with the corresponding access point. During this pairing, only the paired topology is accessible in the environment and other devices are turned off due to interference issues.

Table 3.1.Devices used during the research

Device Under Test		
Label	Device	Generation
GW_A	Gateway	Wi-Fi 4
GW_B	Gateway	Wi-Fi 5
GW_C	Gateway	Wi-Fi 6
AP_A	Access Point	Wi-Fi 5
AP_B	Access Point	Wi-Fi 6

During the tests used two mobile devices and one STB were used for performance measurement. In addition, in order to provide a stable environment in the test environment, updates to the devices were turned off.

Table 3.2. Clients used during the research

Mobile Clients			
Client	Wi-Fi 6	Wi-Fi 5	Wi-Fi 4
iPhone 12	✓	✓	✓
Samsung S10	✓	✓	✓
PS5	✓	x	x

3.1.3. Wireless Mesh Network Throughput

The most important difference between wireless mesh networking devices and other home network devices is performance and flexibility. This distributed network system should work consistently and with high performance at any point of the local area. At this point, it is necessary to look at what the throughput is and the factors that affect the throughput performance in mesh networks.

Definition of Throughput: Measurement of the average rate that data can be sent between one client and another and is typically reported in kilobits per second or megabits per second, where kilobits is 10^3 and megabits is $10^{6.}$ (Henty, 2001)

According to the explanation of the term throughput, an average value over a period of time is calculated in wireless mesh networks. However, there is more than one complex factor that affects this value.

- When independent Wi-Fi devices operate with the values of the adjacent channel. This results in the bandwidth being occupied by different Wi-Fi sources. As a result, it can be observed that the throughput value of the bandwidth occupied by different sources is sharply reduced. (Henty, 2001)

- There is a contrast between distance and throughput. It can be observed that the data transfer rate of Wi-Fi devices decreases as the distance between sources increases.

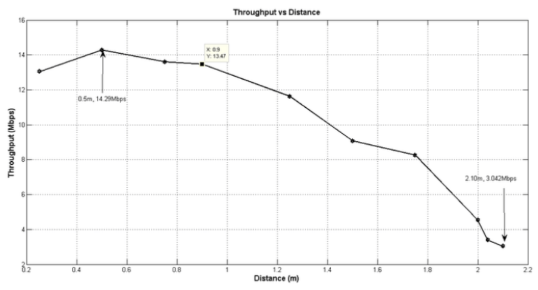


Figure 3.2. Throughput and Distance Contrast
(Source: Ayub et al, 2013)

- Steering actions in wireless mesh networks are the passing between access points to take better advantage of the mesh network and obtain higher quality connections (Kuskonmaz et al. 2011). Stable and accurate passing between clients and access points will provide more throughput to the client. For example, if you look at Figure 5, it is not possible for the client to get stable throughput in "ROOM _6" without steering. However, after a certain period of time, it is seen that the client switches to another access point. This leads to a noticeable increase in throughput.

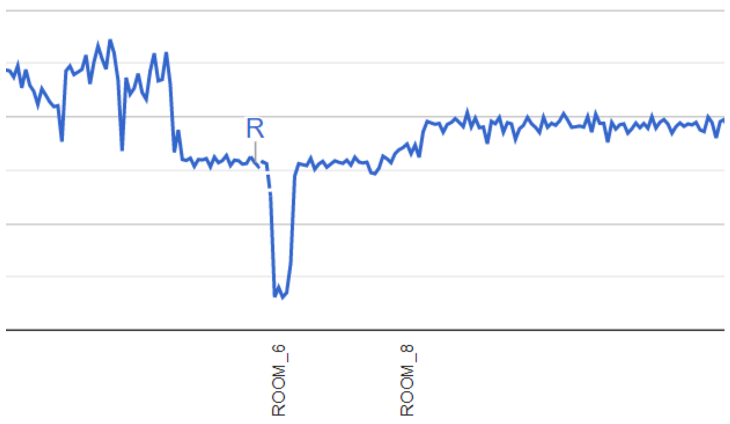


Figure 3.3. Throughput values of the connected client before and after steering

3.1.4. Throughput Monitoring

Various tools have been developed to simulate, analyze and observe throughput values in wireless networks. The common features of these tools are that they can simulate various types of communication protocols at different durations and on different devices.

In mesh wireless performance analysis, the most commonly used protocols are "TCP" and "UDP" protocols. In simulation applications, throughput obtained by changing factors such as various config settings and traffic direction can vary. These variables are specified in the event options that we created within the scope of the research. The simulation configs for these events are detailed in Table 3. The action is visualized in Figure 6.

Table 3.3 Clients used during the research

Event	Protocol	Source	Destination
UDP_Uplink	UDP	Client	PC1
UDP_Downlink	UDP	PC1	Client
TCP_Uplink	TCP	Client	PC1
TCP_Downlink	TCP	PC1	Client

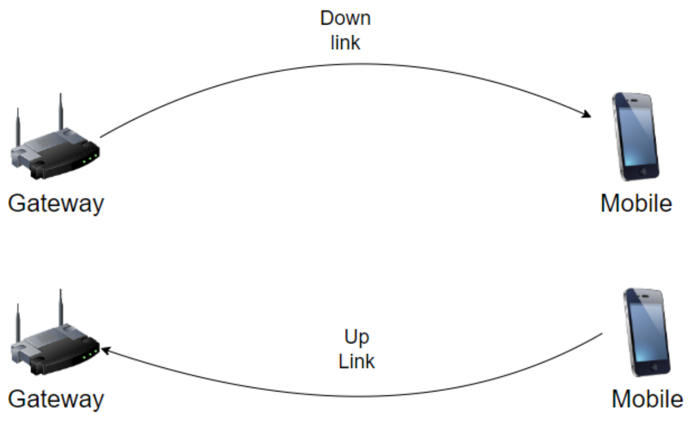
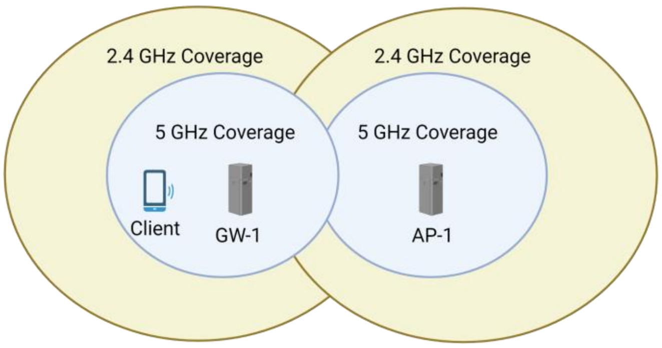


Figure 3.4. Direction information in traffic simulation tool

3.1.5. Test Suite

Due to the number of discrete events during the research, the "steering" function of the system was used in the automation test. First of all, we need to understand the concept of steering. Steering actions in wireless mesh networks refer to requesting clients to change their access points (AP) in order to better exploit the mesh network and achieve higher link quality. (Kuskonmaz et al, 2011)



.

Figure 3.5. Mesh Wireless Coverage

As can be seen in the Figure 7, the highest priority of the steering function is to keep the user at 5 GHz, which always means high bandwidth. During test execution, "Steering", one of the most important functions of mesh network systems, must have more than one discrete event and each event must be tracked regularly. To work out the steering, the client that is within the coverage area of the gateway and access points should be able to switch between the gateway and access point to achieve better performance or stability. As part of the study, as seen in Figure 8, the basis for the steering tests was established on 3 different discrete events that began at the same time.

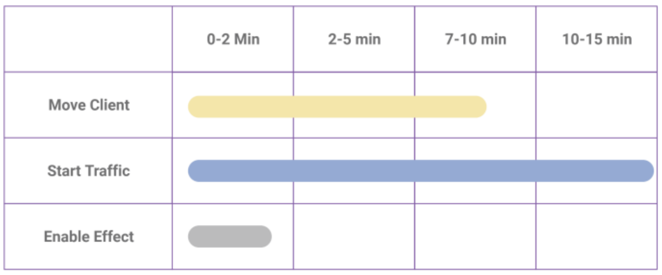


Figure 3.6. Discreate events of steering suite

Looking at the Figure 3.6, the following is examined, respectively.

- **Move Client:** The client should start moving in the test environment. The reason for this action is to change the position of the device and trigger the steering function.
- **Start traffic:** The reason for this action is that there is client traffic while the client is in move. This way, the performance values given by the client during steering can be determined.
- **Enable Effect:** The purpose of this function is to increase the variability of the environment while the client is moving and traffic is simulated.

3.2. Modelling

During the research, when the modeling techniques were examined, it was determined that the Petri-nets modeling technique was suitable for discrete events.

3.2.1. Basic Properties of Petri nets

A Petri is net may be identified as a particular kind of bipartite directed graphs populated by three types of objects. These objects are places, transitions, and directed arcs connecting places to transitions and transitions to places. Pictorially, places are depicted by circles, and transitions are depicted by bars or boxes. In its simplest form, a Petri net may be represented by a transition together with its input and output places. This elementary net may be used to represent various aspects of the modeled systems. Petri Nets (Petri, 1998)

Definition 3.1.1: A State (Place)-Transition is defined as a five-tuple N = (P, T, I, O, M0), where

- P = {p1, p2, …, pm} is a finite set of places;
- T = {t1, t2, …, tn} is a finite set of transitions, $P \cup T \neq \emptyset$, and $P \cap T = \emptyset$;

- I: P × T → N is an input function that defines directed arcs from places to transitions
- O: T × P → N is an output function that defines directed arcs from transitions to places
- M0: P → N is the initial marking.

"Petri-nets", which have a mathematical background, also have some features. Some of these features formed complementary elements in the context of this research. If we consider these few features, we can get the following;

- **Definition of Safeness**: Every local state of a concurrent, hierarchical digital controller can be active or inactive. In the Petri nets a local state corresponds to a place. A place is active, provided it has at least one token. (Miczulski, Adamski, 2006). For example, when we look at the 302nd test case created, it is seen that 20 is selected as the bandwidth value. Until the Bandwidth 20 option is exhausted in combinations, it is not seen other bandwidth selection in any case.
- **Definition of Liveness**: In Petri nets, liveness preserves the possibility to fire any transition after a finite number of other firings, non-deadlock ability–a weak form of liveness, also known as deadlock-freeness–states the absence of reachable deadlocks (Hujsa & Devillers, 2018). This function of the Petri net states that the event in question will never occur unless the transition comes to life. This function prevents the modeled system from entering the deadlock.
- **Definition of Conservation**: In petri nets, conservation is the weighted sum of the tokens in in the net is constant. This rule is very important in such systems where events are activated with tokens. When events are triggered or put on hold, the number of tokens created at the beginning and the number of tokens in the final must be equal. For example, in Figure 9 which starts with 1 token, a token is formed in the place waiting for other discreate events to finish. However, in Figure 10 when this process is finished, our token count drops to 1 again.

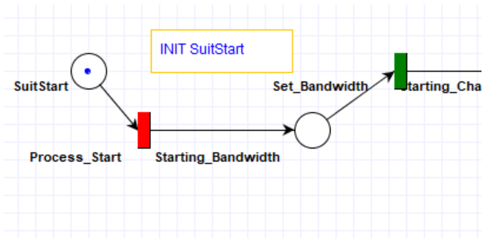
  Definition 3.1.2: If the number of tokens in the initial state x0 and the number of tokens in the ending state x_n are equal, it is called "Strictly Conservative".

$$\sum_{p=1}^{P} m_p = \sum_{p=1}^{P} m_{0p}, \qquad \forall M \in RS(M_0)$$

$M_0 = (1, 0, 0, 0, 0, 0\ldots.)$

$M_n = (0, 0, \ldots ,0, 0, 1)$

$M_0T_W = M_1T_W = M_nT_W = 1$



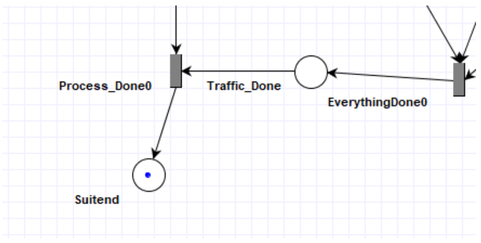


Figure 3.7. First Place Figure 3.7. Last Place

- **Definition of Reachability**: One of the important features of Petri-nets is that the modeled system also has mathematical properties. A reachability graph corresponds to an interleaving model of the system behavior. In the interleaving model all events of a single execution are arranged in a linear order which is called an interleaving sequence. Concurrently executed events appear arbitrary ordered with respect to one another (Djemame, 1999). At this point, deadlocks and all reachable states are obtained.

An interpreted Petri net is such that conditions and events are associated with places and transitions. When the conditions corresponding to some places are satisfied, tokens are assigned to those places and the net is said to be marked. The evolution of tokens within the net follows transition firing rules, With Petri-nets, it can be seen that sequential or discrete events are visualized. (David & Alla, 2005). In our research, we have found that Petri nets can use "conditional features" for the input values. This allows us to determine the required inputs for the cases executed in the automation framework.

Example 3.1: For instance, in Figure 11 of the modeling of the steering test suite

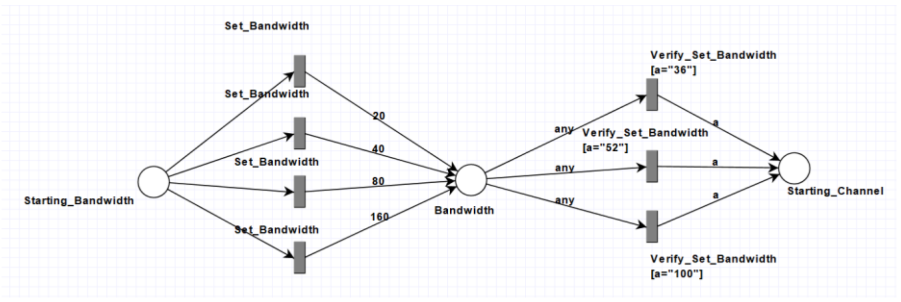


Figure 3.8. Petri-Net of Bandwidth Function

When we examine the Figure 11, the Set Bandwidth value determined as a transition represents what we will call a function. In addition, it is seen that any of the "20, 40, 80, 160" values needed for the test case can be given with the arc. Each time during test generation, the "20,40,80,160" options specified by the user are automatically modeled on the left. At this point during test generation, it will enter the bandwidth values determined.

3.2.2. Analysis Techniques

The ability to analyze Petri nets is often considered the most important activity. By analyzing a Petri net, the behavior and outcomes of the modeled system can be predicted. There are two main approaches to the analysis of Petri nets. These methods are chosen depending on the expectation and purpose of the study.

3.2.2.1. Incidence matrix equation

The incidence matrix is based on a mathematical matrix approach to Petri nets. This approach essentially uses matrix equations to analyze the dynamic behavior of a Petri net. The reachability tree analysis is more efficient in terms of visual representation and tractability. However, as the number of places increases, the tree becomes larger. For this reason, the "incidence matrix" is used in more extensive projects. In many cases, however, they can only be used for special subclasses of Petri nets.

With this mathematical infrastructure, which is one of the advantages of Petri nets, the processing, analysis and verification of the PN modeling structure can be performed in the computer environment. In the research, incidence matrix and state equation were used to verify the model created visually via MISTA.

Definition of Incidence Matrix: For a Petri net N with n transitions and m places, the incidence matrix A = [a_{ij}] is an n x m matrix of integers and its typical entry is given by (Murata, 1989):

$a_{ij=}$ a_{ij}^{+} - a_{ij}^{-}

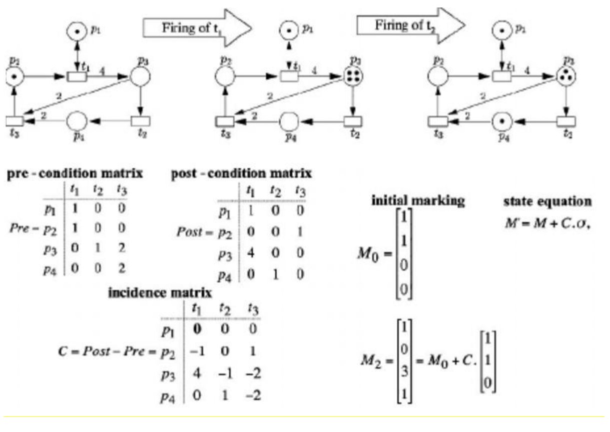
a_{ij}^{+} = w(i,j) is the weight of the arc from transition i to its output place and

a_{ij}^{-} = w(j, i) is the weight of the arc to transition i from its input place j

Definition of state equation: In writing matrix equations, we write a marking M_k as an m × 1 column vector. The ~~*j*~~th entry of M_k denotes the number of tokens in place *j* immediately after the ~~k~~th firing in some firing sequence. The ~~k~~th firing or control vector u_k is an n × 1 column vector of n – 10’s and one nonzero entry, a 1 in the jth position indicating that transition *i* fires at the kth firing. Since *i*th row of the incidence matrix A denotes the change of the marking as the result of firing transition *i*, we can write the following state equation for a Petri net (Murata, 1989).

$$M_k = M_{k-1} + A^T u_k \quad k= 1,2,3….$$

Example 3.2: In Figure 12, each token from places to transitions assigns a value to the precondition matrix. Each token going from transitions to places adds a value to the postcondition matrix. Subtracting these two matrixes gives the "incidence matrix". After this process, the current positions of the tokens in the model are determined by the state equation.



pre - condition matrix

$$Pre = \begin{array}{c|ccc} & t_1 & t_2 & t_3 \\ \hline p_1 & 1 & 0 & 0 \\ p_2 & 1 & 0 & 0 \\ p_3 & 0 & 1 & 2 \\ p_4 & 0 & 0 & 2 \end{array}$$

post - condition matrix

$$Post = \begin{array}{c|ccc} & t_1 & t_2 & t_3 \\ \hline p_1 & 1 & 0 & 0 \\ p_2 & 0 & 0 & 1 \\ p_3 & 4 & 0 & 0 \\ p_4 & 0 & 1 & 0 \end{array}$$

initial marking

$$M_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

state equation

$$M' = M + C.\sigma,$$

incidence matrix

$$C = Post - Pre = \begin{array}{c|ccc} & t_1 & t_2 & t_3 \\ \hline p_1 & 0 & 0 & 0 \\ p_2 & -1 & 0 & 1 \\ p_3 & 4 & -1 & -2 \\ p_4 & 0 & 1 & -2 \end{array}$$

$$M_2 = \begin{bmatrix} 1 \\ 0 \\ 3 \\ 1 \end{bmatrix} = M_0 + C.\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 3.9. Incidence Matrix Example

(Source: Sun et al, 2012)

3.2.2.2 Reachability Tree

In the reachability tree, nodes represent places and arcs represent transitions. The main use of the reachability tree is to check safeness, boundedness and conservation properties. In principle, this method can be used in all Petri net models. However, as the modeling depth increases, the complexity of this analysis increases, and it reaches a point where it is no longer applicable. Therefore, this method of analysis was widely used in the past for models with less complex depth.

However, with evolving computer technologies, this analysis technique can also be used with greater-depth models. The mesh network steering test structure modeled in this context could be analyzed using the "reachability tree" technique. This analysis was performed using the MISTA tool because it could be performed faster and more effectively in the computer environment.

Definition of Reachability Tree: Given a Petri net (N,M_0) from the initial marking M_0 we can obtain as many “new” markings as the number of the enabled transitions. From each new marking, we can again reach more markings. This process results in a reachability tree representation of the markings. Nodes represent markings generated from M_0 (the root) and its successors, and each arc represents a transition firing, which transforms one marking to another. (Murata, 1989).

For example, in Figure 3.11. All places within reach of the fired tokens are shown in the tree. As can be seen from the example figure, this type of analysis is inefficient for models above a certain size

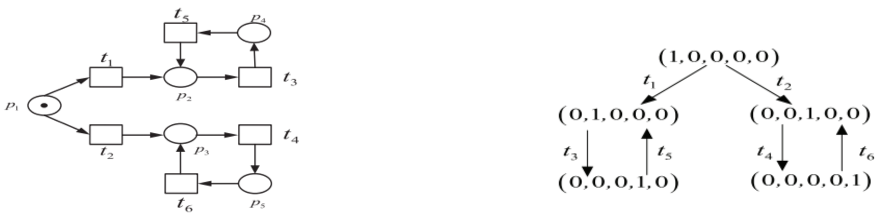


Figure 3.10. Reachability Tree Example
(Source: Ren et al, 2012)

3.2.3. MISTA

As part of the research, tools that can create tests with Petri nets were searched for. As a result of the research, the tool "MISTA" was selected as the application that meets the criteria. The application was developed by "Dianxiang Xu" and his team.

Definition of MISTA: Application for automated test generation and execution by using high-level Petri nets as finite state test models. (Xu, 2011)

The purposes of use of the application.

- Modeling of wireless performance test suite.
- Generating test cases from the designed petri-net model.

The application is written in an open source via JAVA. The application can create test outputs of the designed model in different formats for different frameworks or programming languages.

Frameworks or programming languages that can be output.

- JAVA
- C++
- C#
- PHP
- VB
- HTML
- C
- KBT
- SELENIUM
- KBT

In the application, the desired model can be created according to the settings in the menu. After the modelling process is complete, test cases can be generated to run on the desired platform. Figure 3.12 shows the model created during the research and the interface of the MISTA application.

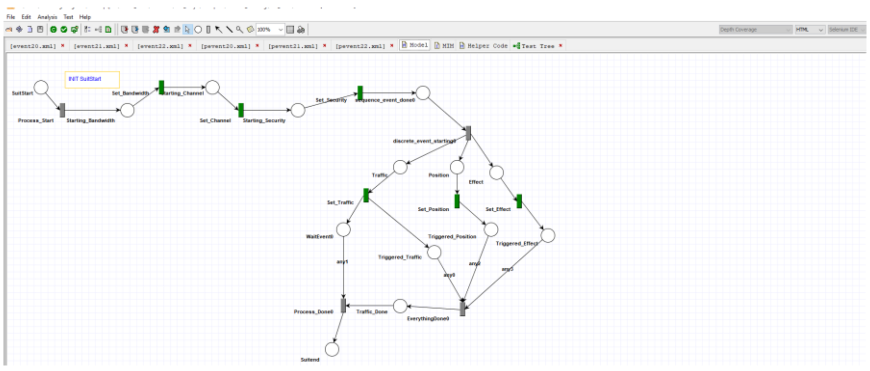
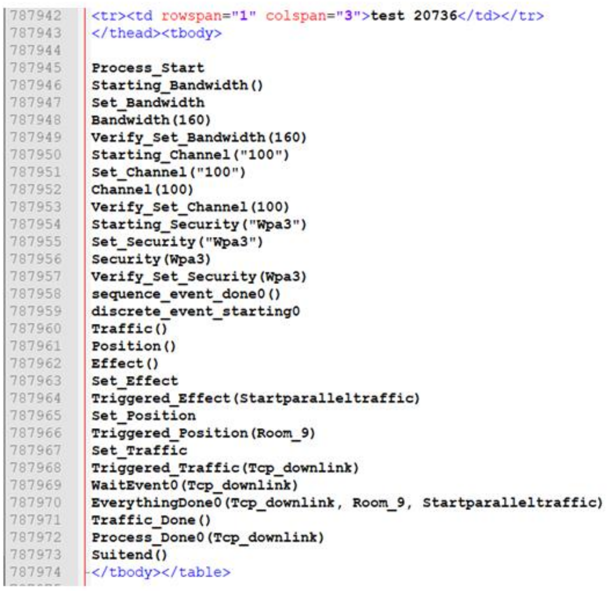


Figure 3.11. A petri-net modeled via MISTA

The "steering" test suite is modeled with the app, the possible combinations are still calculated by the app. The application writes these calculations to files in xml format. A test management framework that run this html output test is also used. In this way, a fast and organic link between the model and the test cases can be established. An example of html output is shown in Figure. 3.13. As can be seen in the output, the particular functions and inputs are presented pre-built.



```
<tr><td rowspan="1" colspan="3">test 20736</td></tr>
</thead><tbody>

Process_Start
Starting_Bandwidth()
Set_Bandwidth
Bandwidth(160)
Verify_Set_Bandwidth(160)
Starting_Channel("100")
Set_Channel("100")
Channel(100)
Verify_Set_Channel(100)
Starting_Security("Wpa3")
Set_Security("Wpa3")
Security(Wpa3)
Verify_Set_Security(Wpa3)
sequence_event_done0()
discrete_event_starting0
Traffic()
Position()
Effect()
Set_Effect
Triggered_Effect(Startparalleltraffic)
Set_Position
Triggered_Position(Room_9)
Set_Traffic
Triggered_Traffic(Tcp_downlink)
WaitEvent0(Tcp_downlink)
EverythingDone0(Tcp_downlink, Room_9, Startparalleltraffic)
Traffic_Done()
Process_Done0(Tcp_downlink)
Suitend()
</tbody></table>
```

Figure 3.12. Test case outputs obtained via MISTA.

CHAPTER 4

APPROACH

4.1. Performance Analysis

In this section, we first explain the modeling process of the performance analysis of the "steering" function, which is one of the most important features of mesh systems, with Petri nets. The problems identified by the generic structure created by this process and the new approaches discovered in the process are explained. Secondly, the application "EIN Tool" is presented, which was developed to speed up this modeling process and increase its efficiency. In the last part, the stability of the test generator system created by MISTA was analyzed and several comparisons were made.

4.1.1. Modeling of the Test Suite

Within the scope of the research, the wireless steering test process was modeled as end-to-end. Both start and end events of the process are included in the model. In this way, all possibilities of test combinations were analyzed.

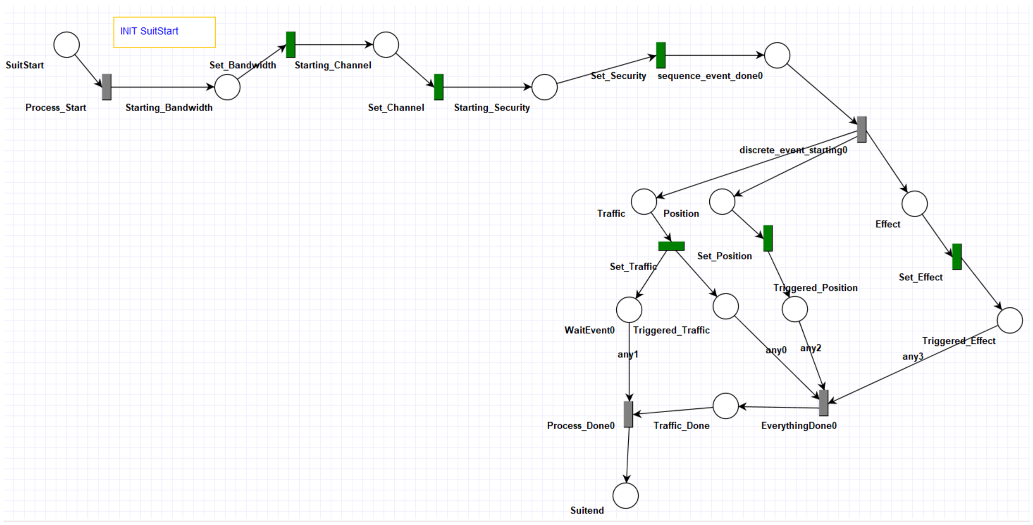


Figure 4.1. PN model of steering suite model

To reduce the visual complexity of the model, a sub-model is created for each of the events on the model. Each green-colored event in the image shown in Figure 4.1 has its own sub-model. Each of these events will be addressed in the next titles. The model created in the research was based on the end-to-end modeling of the steering tests. The events are explained in Table 4.1.

Table 4.1. Combination table of model

Event Name	Event Type	Combination	Description
Bandwidth	Sequence Event	20,40,80,160	Set bandwidth of the device.
Channel	Sequence Event	36,52,100	Set channel of the device.
Security	Sequence Event	WPA,WPA2,WPA3	Set security type of the device.
Traffic	Discreate Event	UDP_Uplink, UDP_Downlink, TCP_Uplink,TCP_Downlink	Set packet type and direction to be simulated during testing on the device
Move	Discreate Event	ROOM_3,ROOM_6, ROOM_7,ROOM_10	Set the position of the client during testing on the device
Effect	Discreate Event	Start_Parallel_Traffic, Add_Client	Set the variability of the environment during testing on the device

During the test, all scenarios that may affect the steering tests were created, along with this content that may cover all end-to-end tests. In the following subheadings, the sub-models are mentioned in order.

4.1.1.1. Bandwidth

This is the amount of data that a device connected to the Internet can receive in a given period of time. In other words; it is the transmission and transport of the maximum amount of data that can be transmitted over a channel. The combinations in this option are "20, 40, 80, 160" respectively for this research.

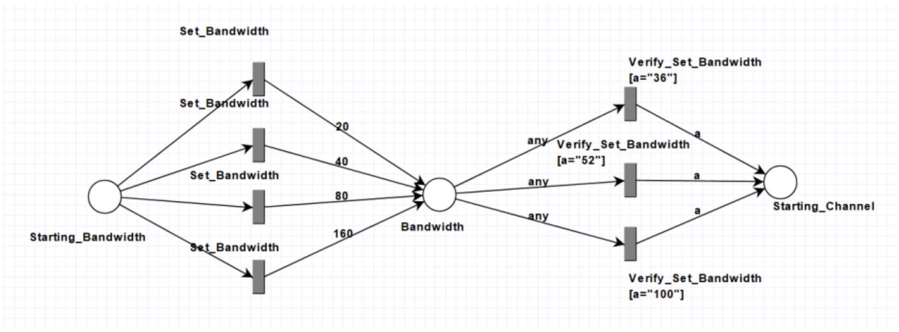


Figure 4.2. Sub PN model of Bandwidth

In the bandwidth sub modeling, the combination of bandwidths that the gateway or AP can use is created. These combinations appear in the xml output by entering one of the "Set_Bandwidth" transitions. Test case variations created according to these combinations are processed by the test management tool and set the gateway or access point to the required bandwidth value.

This value, which is then set, verifies whether the value is set properly with the "Verify_Set_Bandwidth" place. This process is also done via test-management-tool in the same way. Another feature of “Verify_Set_Bandwidth” is it sets the necessary values for the next event. The bandwidth is formed in the xml file created by MISTA as a result of the combinations from the partial model shown in Figure 4.2, in xml output.

4.1.1.2. Channel

Wireless channels are the frequency ranges used to send and receive data between the gateway/access point and your wireless device. The combinations in this option are "36, 52, 100" respectively for this research.

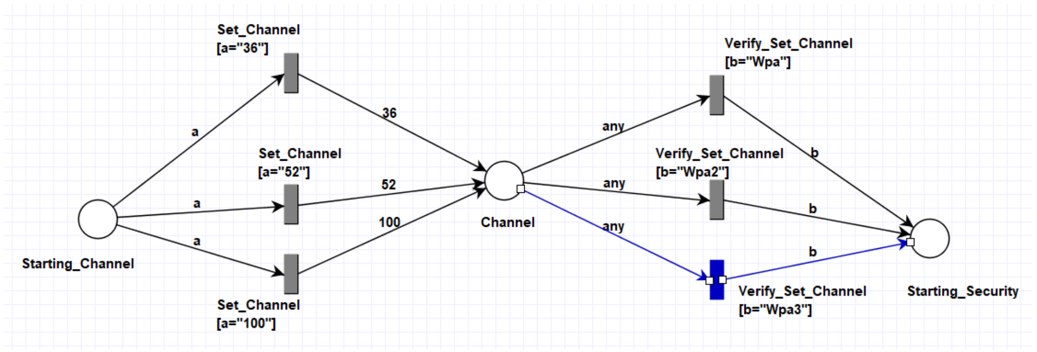


Figure 4.3. Sub PN model of Channel

In the channel sub modeling, the combination of channels that the gateway or AP can use is created. These combinations appear in the xml output by entering one of the "Set_Channel" transitions. Test case variations created according to these combinations are processed by the test management tool and set the gateway or access point to the required channel value.

This value, which is then set, verifies whether the value is set properly with the "Verify_Set_Channel " place. This process is also done via test-management-tool in the same way. Another feature of “Verify_Set_Channel” is it sets the necessary values for the next event. The channel is formed in the xml file created by MISTA as a result of the combinations from the partial model shown in Figure 4.3, in xml output.

4.1.1.3. Security Type

Wi-Fi security is designed to prevent unauthorized access to devices on the wireless network. Most home routers provide multiple security modes that change according to their protection level. The combinations in this option are "WPA, WPA2, WPA3" respectively for this research.

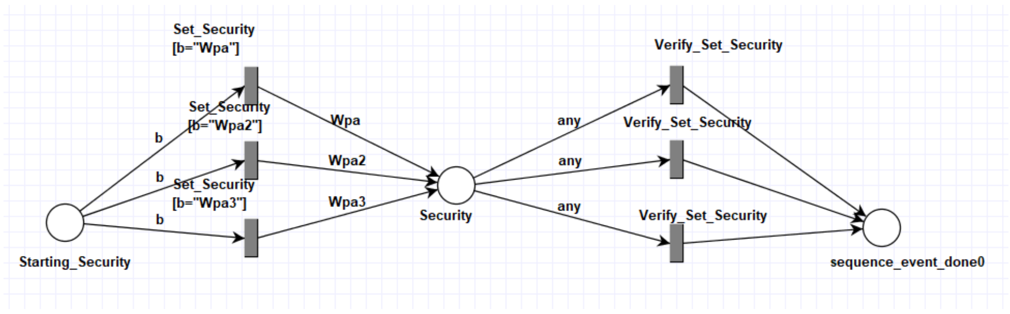


Figure 4.4. Sub PN model of Security

In the security sub modeling, the combination of security types that the gateway or AP can use is created. These combinations appear in the xml output by entering one of the "Set_Security" transitions. Test case variations created according to these combinations are processed by the test management tool and set the gateway or access point to the required channel value.

This value, which is then set, verifies whether the value is set properly with the "Verify_Set_Security " place. This process is also done via test-management-tool in the same way. The security type is formed in the xml file created by MISTA as a result of the combinations from the partial model shown in Figure 4.4, in xml output.

4.1.1.4. Traffic Type

During testing, it is necessary to analyze how much throughput the client connected to the GW or AP can experience. Applications such as iperf are used for this analysis. The combinations were created by selecting the traffic type and direction in the modeling. The process of IPERF is shown in Figure 4.5. The throughput value between the two devices, considered client and server, is measured.

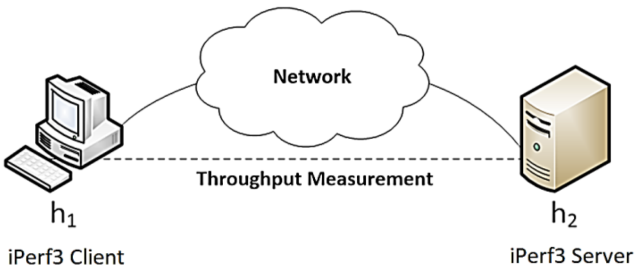


Figure 4.5. Throughput measurement with iPerf3

The traffic simulation process is started by executing the test cases created by MISTA as a result of the model's combination via the test management tool. When modeling the traffic event, the required combinations were created one by one. At the same time, the event was created at a "place" waiting for its completion, since it is a primary discrete event. This "places" is called "WaitEvent" in Figure 4.6.

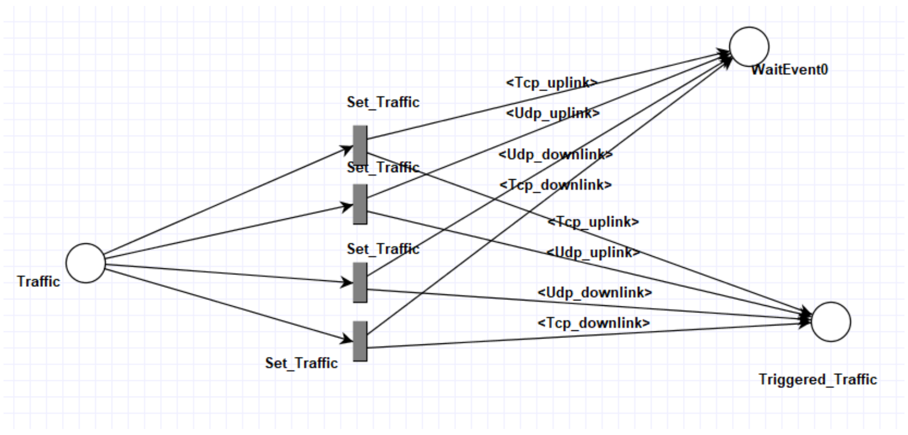


Figure 4.6. Sub PN model of traffic event

4.1.1.5. Position

The position event is an event generated during "Steering" tests to allow the mobile client to switch between the gateway and the access points. The main purpose of this event is to allow the mobile clients to move so that they can switch between the appropriate GHz bands. As a result of this action, the client moves to the most stable and powerful GHz band. These steering probabilities are explained in Table 4.2

Table 4.2 Steer probabilities

Current GHz	Current Device	Target Band	Target Device
5 GHz	Gateway	5 GHz	Access Point
5 GHz	Access Point-1	5 GHz	Gateway
5 GHz	Gateway	2.4 GHz	Gateway
2.4 GHz	Gateway	5 GHz	Gateway
5 GHz	Access Point-1	5 GHz	Access Point-2
5 GHz	Access Point-2	5 GHz	Access Point-1
5 GHz	Access Point-1	2.4 GHz	Access Point-1
2.4 GHz	Access Point-1	5 GHz	Access Point-1

All probabilities of these steer are seen at different "RSSI" levels. To change on these levels, the mobile clients must be moved, of course. For all these reasons, different combinations of the position event have been created for the spaces where the client is located. In this way, the RSSI given by the mobile client in "ROOM _7" and "ROOM _2" is different. Figure 4.7. shows the possible combinations for this discrete event.

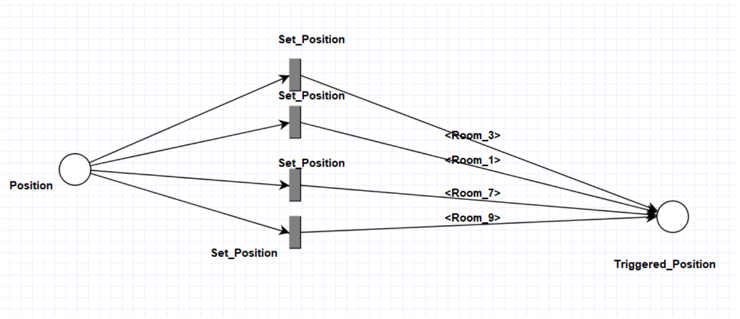


Figure 4.7. Sub PN model of position event

4.1.1.6. Effect

Definition of Effect: This discrete event, added to the steering test process in the context of research. increases the variability of the test process, is intended to bring a different perspective to the test process. The main purpose is to vary the static test setup at regular intervals. This dynamic process was developed specifically for the "pesticide paradox" problem. This discrete event was experimental. It was created with two combinations. These combinations are;

- **Definition of Add Client:** While the client is moving during the steering test, new clients are added to the setup in another environment. Since such user behavior was not verified in the steering tests prior to this research, the behavior of the steering algorithm was analyzed. The clients added to the active steering device are STBs or mobile clients in the test environment. In addition, this addition process is performed multiple times.

- **Definition of Start parallel traffic**: During the Mobile Client Steering Test, IPERF is simulated too with another device connected to the network, not through the single client. In this way, the set bandwidth is not used with a single device. Since such user behavior was not verified in the steering tests prior to this research, the behavior of the steering algorithm was analyzed.

4.1.2. Modeling of the Discreate Events

The main reason why the Steering test process is included in the modeling is that the test process contains too many discrete events. Detailed information about this process can be found in the background chapter. There are 2 events that are discretized during the commonly used test, these are;

- Simulation of the traffic between the controlling client and the device it is connected to. This event is referred to as "traffic" in this research.

- Moving the client so that it can steer. This event is also referred to by the term "move" in this study.

In order to better understand this process, the activity diagram of the process was created in Figure 4.8. When the diagram is examined, two events that are carried out discreate of each other stand out.

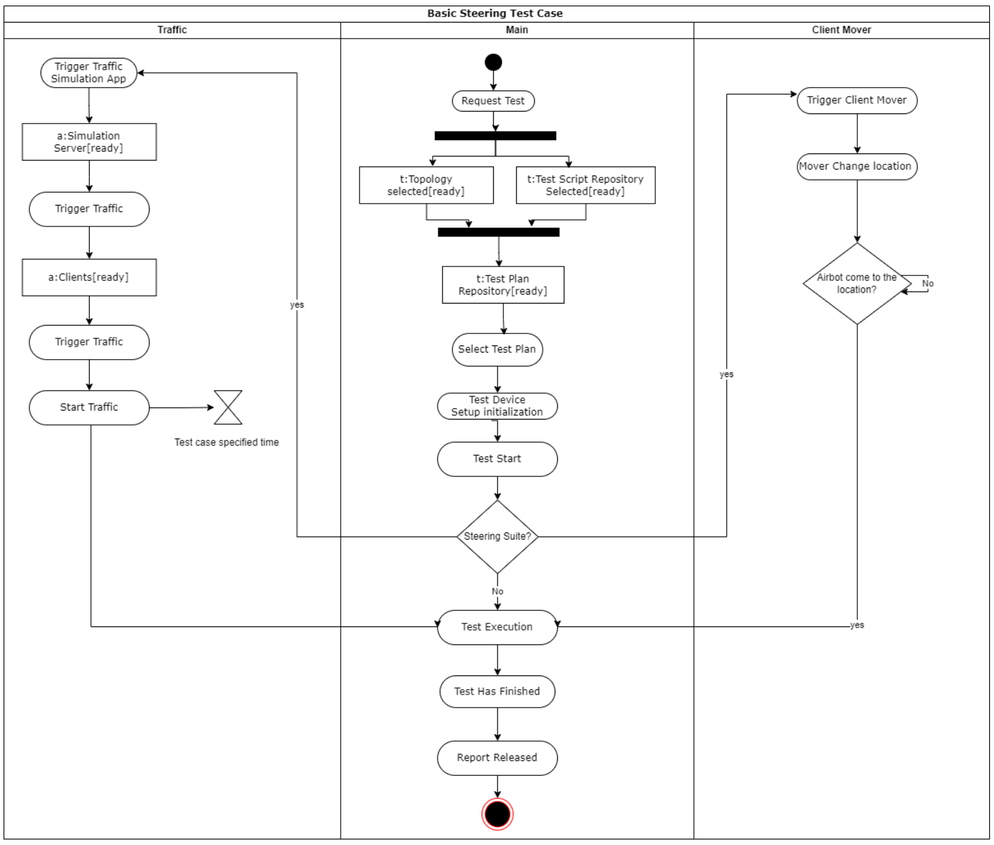


Figure 4.8. Activity Diagram of Generic Steering Test Case

In order to better understand this process, the activity diagram of the process was created in Figure 4.8. When the diagram is examined, two events that are carried out discreate of each other stand out. Our primary goal was to model the activity diagram in Figure 4.8 with Petri nets. In modeling this test process with Petri nets, we also created a "waiting area" for discrete events. In this modeling, the other discrete action is put on hold until the primary "traffic" event is completed.

As seen in Figure 4.9., other discreate events continue to run until the Traffic event ends. This waiting process is provided by the place named "WaitEvent0". When the traffic discrete event is finished, the other events are finished one after the other.

When the traffic discrete event is finished, the other events are completed one at a time. However, it should be noted that the waiting process here is done by the test management framework.

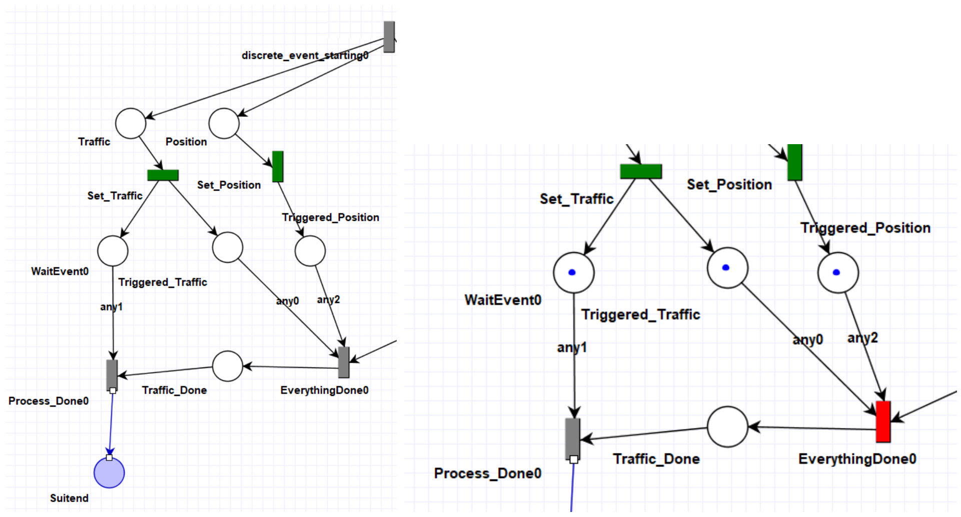


Figure 4.9. Petri-Nets model of steering test part

As part of the research, another event was added to these discrete events to provide a better structure for error detection. The most important feature of this discrete event is the change of the environment during the test. In this way, the steering test becomes more variable. Figure 4.10. shows part of the model with this newly added event.

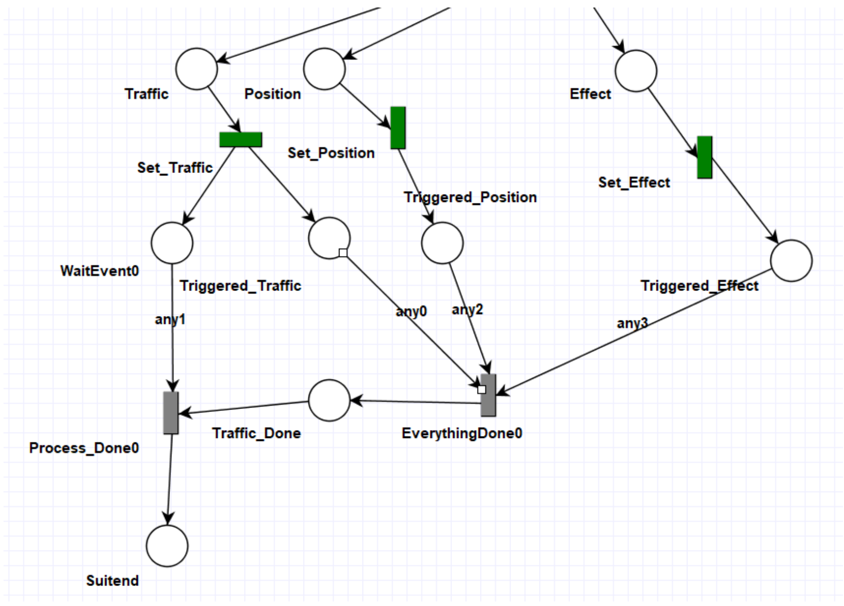


Figure 4.10. Petri-Nets Model of New Steering Test Part

To make this analysis process healthier and reduce complexity, a new log script was created to be used in the research. This script will be analyzed in more detail in next outline. This discrete event can be removed or replaced with another discrete event if needed.

4.1.3. Analyzing of Steering Model

The analysis techniques, which are one of the biggest advantages of conducting the research using the Petri net modeling technique, were tested in the model we created as part of the research.

One of these analysis techniques, the "Reachability Tree", was tested in the model with the "MISTA" application. The "Incidence Matrix" technique, another analysis technique, was also tested on some of the sub models prepared in the model. The details of these analysis results are described in the next subsections.

4.1.3.1. Reachability Tree Analysis

The "Reachability Tree" technique, one of the analysis techniques, was applied to the model created with the EIN tool. The current modeled PN model can be created using the MISTA tool in Reachability Tree format. As a result of these checks, "151" different branches were observed. In none of these branches was a situation found to contradict the "safety", "boundedness" and "conservation" features of the PN modeling.

Another point is, that a dead-lock analysis was performed for the model prepared with the EIN tool. The result of this analysis was that no deadlock condition was detected

4.1.3.2. Incidence Matrix Analysis

To verify the accuracy of the visually created model, a small part of the created model was used. The details of this part can be seen in Figure 26.

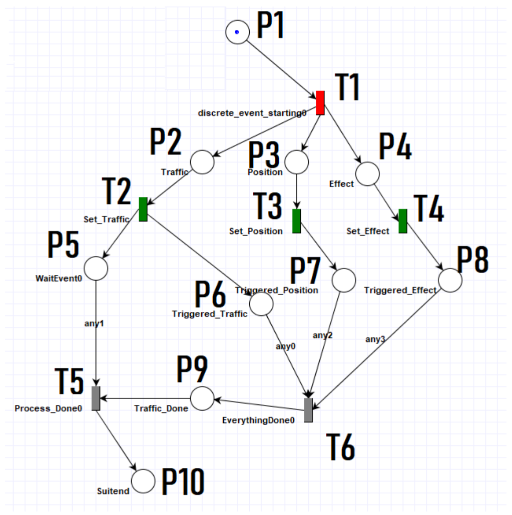


Figure 4.11. Model Part of Wireless Mesh Steering

The place and transitions of this image taken on the model are given an ID as shown in Figure 4.11. To make the mathematical transition from modeling, fire able tokens from places to transitions assigns as a "1" to the precondition matrix. This process is turned into a matrix as seen in Figure 4.12.

$$\textbf{Precondition} = \begin{array}{c|cccccc} & T1 & T2 & T3 & T4 & T5 & T6 \\ P1 & 1 & 0 & 0 & 0 & 0 & 0 \\ P2 & 0 & 1 & 0 & 0 & 0 & 0 \\ P3 & 0 & 0 & 1 & 0 & 0 & 0 \\ P4 & 0 & 0 & 0 & 1 & 0 & 0 \\ P5 & 0 & 0 & 0 & 0 & 1 & 0 \\ P6 & 0 & 0 & 0 & 0 & 0 & 1 \\ P7 & 0 & 0 & 0 & 0 & 0 & 1 \\ P8 & 0 & 0 & 0 & 0 & 0 & 1 \\ P9 & 0 & 0 & 0 & 0 & 1 & 0 \\ P10 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Figure 4.12. Precondition matrix

Secondly fire able tokens from transitions to places assigns as a "1" to the precondition matrix. This process is turned into a matrix as seen in Figure 4.13.

$$\textbf{Postcondition} = \begin{array}{c|cccccc|} & T1 & T2 & T3 & T4 & T5 & T6 \\ P1 & 0 & 0 & 0 & 0 & 0 & 0 \\ P2 & 1 & 0 & 0 & 0 & 0 & 0 \\ P3 & 1 & 0 & 0 & 0 & 0 & 0 \\ P4 & 1 & 0 & 0 & 0 & 0 & 0 \\ P5 & 0 & 1 & 0 & 0 & 0 & 0 \\ P6 & 0 & 1 & 0 & 0 & 0 & 0 \\ P7 & 0 & 0 & 1 & 0 & 0 & 0 \\ P8 & 0 & 0 & 0 & 1 & 0 & 0 \\ P9 & 0 & 0 & 0 & 0 & 0 & 1 \\ P10 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Figure 4.13. Postcondition matrix

To obtain the incidence matrix, the matrices of each postcondition must be subtracted to the precondition matrix, as mentioned in 3.2.0.1. This mathematical operator is “$a_{ij=}$ $a_{ij}^{postcondition}$ - $a_{ij}^{precondition}$”. The incidence matrix after this operation is shown in Figure 4.14.

$$\textbf{Incidence Matrix} = \begin{array}{c|cccccc|} & T1 & T2 & T3 & T4 & T5 & T6 \\ P1 & -1 & 0 & 0 & 0 & 0 & 0 \\ P2 & 1 & -1 & 0 & 0 & 0 & 0 \\ P3 & 1 & 0 & -1 & 0 & 0 & 0 \\ P4 & 1 & 0 & 0 & -1 & 0 & 0 \\ P5 & 0 & 1 & 0 & 0 & -1 & 0 \\ P6 & 0 & 1 & 0 & 0 & 0 & -1 \\ P7 & 0 & 0 & 1 & 0 & 0 & -1 \\ P8 & 0 & 0 & 0 & 1 & 0 & -1 \\ P9 & 0 & 0 & 0 & 0 & -1 & 1 \\ P10 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Figure 4.14. Incidence Matrix

Finally, the "State Equation" is calculated so that we can see the changes in the model as a result of the next token firing. The details of this calculation are explained in Title 3.2.0.1. The calculation is shown in Figure 4.15.

- State equation formula for find next situation

$M_1 = M_0 + A_v$

$$M^1 = \begin{array}{c|c|} P1 & 1 \\ P2 & 0 \\ P3 & 0 \\ P4 & 0 \\ P5 & 0 \\ P6 & 0 \\ P7 & 0 \\ P8 & 0 \\ P9 & 0 \\ P10 & 0 \end{array} + \begin{array}{c|cccccc|} & T1 & T2 & T3 & T4 & T5 & T6 \\ P1 & -1 & 0 & 0 & 0 & 0 & 0 \\ P2 & 1 & -1 & 0 & 0 & 0 & 0 \\ P3 & 1 & 0 & -1 & 0 & 0 & 0 \\ P4 & 1 & 0 & 0 & -1 & 0 & 0 \\ P5 & 0 & 1 & 0 & 0 & -1 & 0 \\ P6 & 0 & 1 & 0 & 0 & 0 & -1 \\ P7 & 0 & 0 & 1 & 0 & 0 & -1 \\ P8 & 0 & 0 & 0 & 1 & 0 & -1 \\ P9 & 0 & 0 & 0 & 0 & -1 & 1 \\ P10 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \; \mathbf{x} \; \begin{array}{c|c|} T1 & 1 \\ T2 & 0 \\ T3 & 0 \\ T4 & 0 \\ T5 & 0 \\ T6 & 0 \\ T7 & 0 \\ T8 & 0 \\ T9 & 0 \\ T10 & 0 \end{array}$$

Figure 4.15. State equation result

When we check the token fired visual model in Figure 4.17. with the resulting matrix in Figure 4.16. it is seen that they support each other.

$$M^1 = \begin{array}{c} P1 \\ P2 \\ P3 \\ P4 \\ P5 \\ P6 \\ P7 \\ P8 \\ P9 \\ P10 \end{array} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Figure 4.16. M^1 Matrix

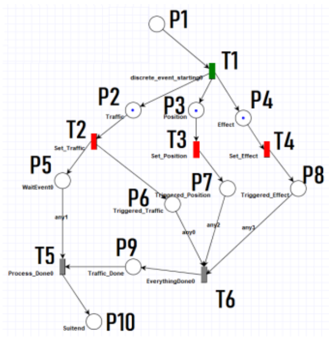


Figure 4.17. New Model Visual

With this example check, the MISTA tool and the mathematical results of the structure we are modeling are matched. In this way, the model visualized by the MISTA tool is validated with the PN model.

4.2. Reuse of the PN Models

Reusable test automation systems are becoming increasingly important in software testing. Reusable test automation structures created in this way have many benefits for organizations.

- Work effort is not lost. The next test cycle can start where this one left off. When the product gets new features or when product change, they can be incrementally added or removed this feature to the model. (Apfelbaum & Doyle, 1997)
- Depending on the criticality of the product, it can be included in a longer or shorter automation cycle.
- If there are people who are newly included on the team, they can accelerate quickly by examining the model. (Apfelbaum & Doyle, 1997)

This part of research introduces a reusable testing approach for Petri-nets. The next section will explain the application we created for our approach and how it works.

4.2.1. EIN Tool

The name of the application we developed as part of the research is "EIN Tool". The application is written in Python programming language. The application has some advantages and disadvantages. If we look at them;

Advantages

1) People using the application can create models without having PN modeling skills.
2) The structure prepared in the application can be changed without much effort.

Disadvantages

1) While designing application models, it acts according to a certain order. Due to this situation, it is not as flexible as a model to be designed by an individual.

4.2.1.1. GUI

The application was designed to be simple and easy to understand. In line with the application goals, there are two usage patterns in total. One is the "New Project" option to create a new test-suite. The second is the "Edit Project" section, where we select our test options to be reused in a previously created test-suite. The main screen of the application can be seen in Figure 4.18.

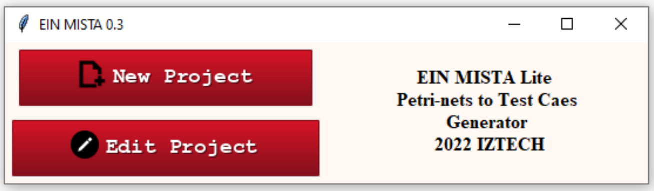
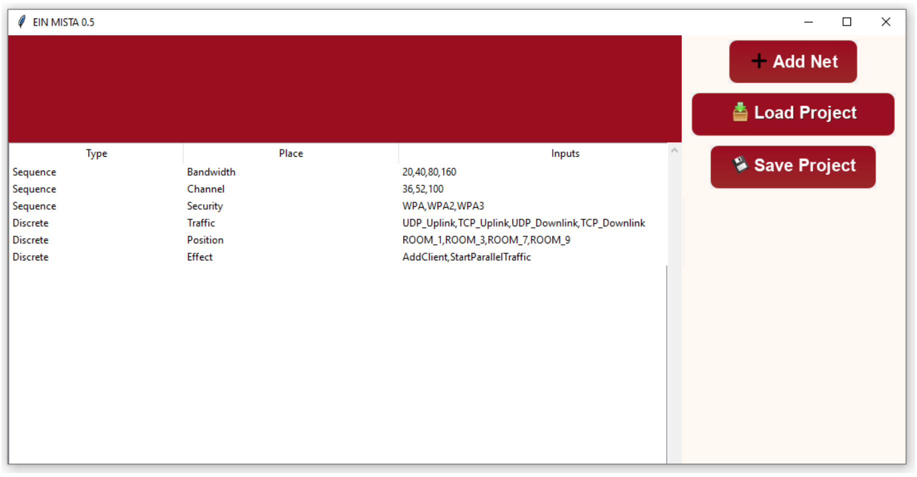


Figure 4.18. Main Screen of EIN Tool

If the tester wants to create a new test-suite. A screen like the one shown in Figure 4.19. will be displayed. user can add new events if necessary or save the new test suite.



Type	Place	Inputs
Sequence	Bandwidth	20,40,80,160
Sequence	Channel	36,52,100
Sequence	Security	WPA,WPA2,WPA3
Discrete	Traffic	UDP_Uplink,TCP_Uplink,UDP_Downlink,TCP_Downlink
Discrete	Position	ROOM_1,ROOM_3,ROOM_7,ROOM_9
Discrete	Effect	AddClient,StartParallelTraffic

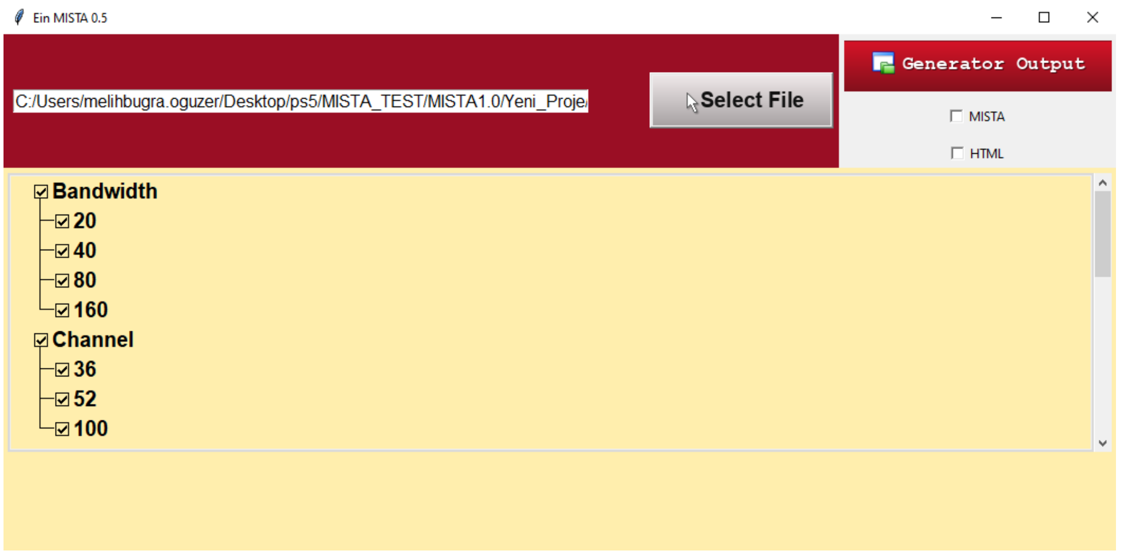
Figure 4.19. New Project Screen of EIN Tool

The user can add new events to the test-suite by clicking on the "Add Net" button. These events must be sequential or discrete events. The "Event" information prompted on the same screen determines the name of this event in the model. Finally, it determines the inputs that the event will use in the model. As a result, each "Add Net" request can create as many discrete and sequential events and output a model.

After preparing the desired suite with "EIN Tool", the user saves the suite with the "Save Project" in Figure 4.19. The recorded test suite can be reused for further editing.

If it is desired to reusable a prepared test suite, user should open the prepared file on the "Edit Project" screen. For this operation, the file prepared via "EIN Tool" is selected with the "Select file" button in Figure 4.20.

A screen like the one shown in Figure 4.20. is displayed where the user can edit the structure of the suite. In this way, events that are not supported by the device or whose execution is not important at this time can be removed from the suites that have already been prepared.


Ein MISTA 0.5
C:/Users/melihbugra.oguzer/Desktop/ps5/MISTA_TEST/MISTA1.0/Yeni_Proje
Select File
Generator Output
MISTA
HTML

☑ Bandwidth
- ☑ 20
- ☑ 40
- ☑ 80
- ☑ 160

☑ Channel
- ☑ 36
- ☑ 52
- ☑ 100

Figure 4.20. Edit Project Screen of EIN Tool

Test cases can be generated in two different ways after the change has been made in the currently used test suite on the Edit Project screen.

- MISTA
- HTML

The first option, "MISTA", remodels the suite the user is working on. With this modeling, the user can generate test cases in the MISTA application. The second option "HTML", the user can update the test case file user has created via MISTA in the past, according to the new suite they have edited.

4.2.1.2. Test generator via MISTA

It is one of the ways to create the test scenario using the test suite prepared with "EIN Tool". To use it, it is enough to check the MISTA checkbox and click on the "Generator Output" button (see Figure 4.20.). With this option "EIN Tool" creates a new PN model from the updated test suite. With this created model, it can generate tests from the "MISTA" as it wishes. The process is visualized with flowchart in Figure 4.21.

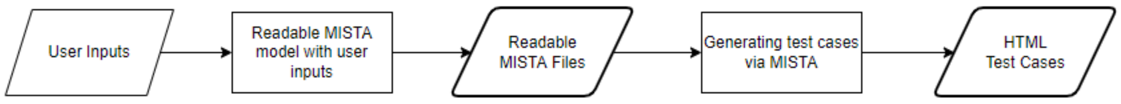

User Inputs
Readable MISTA model with user inputs
Readable MISTA Files
Generating test cases via MISTA
HTML Test Cases

Figure 4.21. Creating Test Cases via MISTA

4.2.1.3. Test generator via HTML

This is another way to create a test case using the test-suite prepared with "EIN". To do this, simply check the "HTML" checkbox and click the "Generator Output" button (see Figure 35). With this option "EIN" will update the previously created test case file according to the updated test-suite. An update is the removal of unwanted step combinations from a fully prepared "HTML" file. This ensures that the "HTML" file, which is created only once, is used reusable. The process is visualized with flowchart in Figure 4.22.

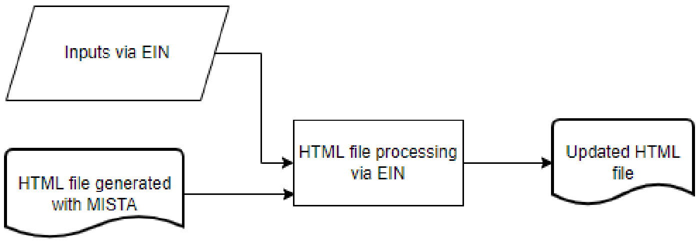


Figure 4.22 Creating Test Cases via EIN

4.2.1.4. Test Execution

To execute the test cases in the created html file, a test management framework should be prepared accordingly. For the test cases performed in the research, function was created by the test management framework and various parsing mechanisms were used. In Figure 4.23. a flowchart of how the process progresses is drawn.

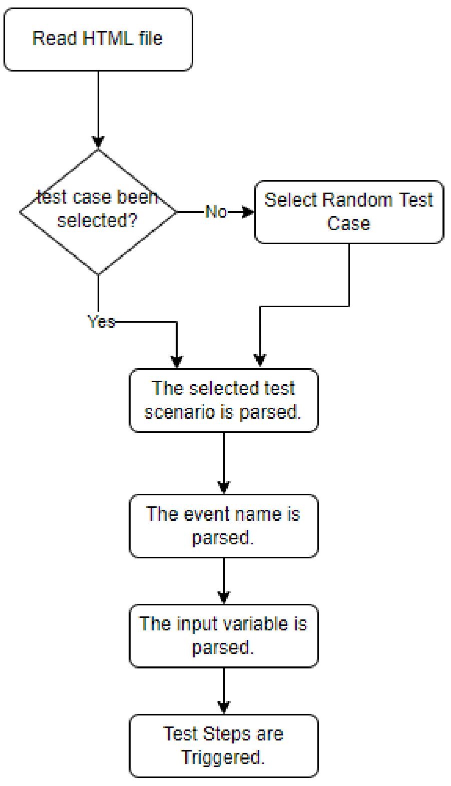
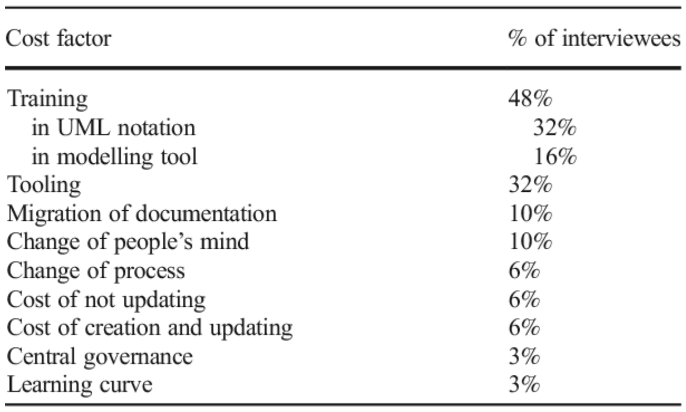

Read HTML file
test case been selected?
No
Select Random Test Case
Yes
The selected test scenario is parsed.
The event name is parsed.
The input variable is parsed.
Test Steps are Triggered.

Figure 4.23. Model of HTML Selection Process

4.2.2. Petri-Net Learning Curve

The process of learning a new knowledge is an extremely long and costly process. Even with a tool like UML which is widely used in the industry and academia, it takes a very long and costly learning process. A major factor that emerged was the cost of training in UML notation. This situation was also related to the educational background. (Fernández-Sáez et al, 2018). However, The table in Figure 4.24. shows the interview results of the "UML" modeling users. The interviews show that users have some biases and cost concerns regardless of modeling.



Cost factor	% of interviewees
Training	48%
in UML notation	32%
in modelling tool	16%
Tooling	32%
Migration of documentation	10%
Change of people's mind	10%
Change of process	6%
Cost of not updating	6%
Cost of creation and updating	6%
Central governance	3%
Learning curve	3%

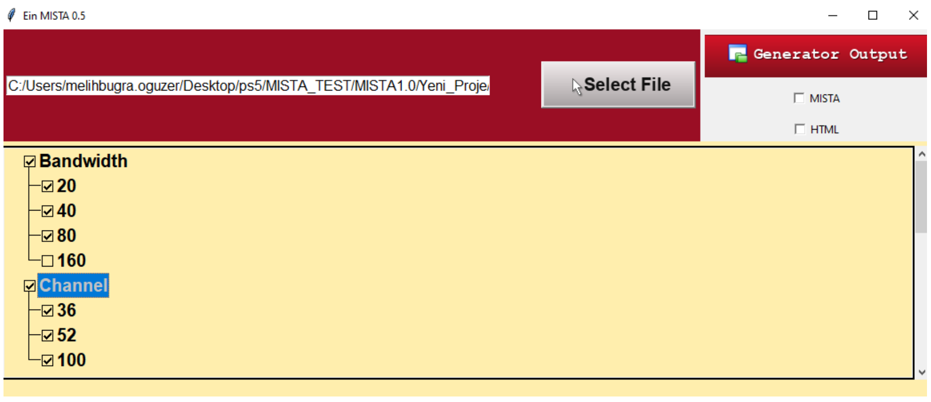
Figure 4.24. Cost factors related to the use of UML
(Source: Fernández-Sáez, et al, 2018)

Considering all these findings, it is an undeniable fact that the learning and cost problems of modeling processes are critical. Within the scope of the research, the reusability of the test scenarios modeled with PN was ensured for the first time. In the next subheading you will find created scenarios in which reuse is possible, as well as some examples of their visualization in the tool.

4.2.2.1. Scenario Examples for Reusable

4.2.2.1.1. Scenario 1

The device tested in the automation system with the designation "GW_C" supports a bandwidth of "160". However, the devices "GW_B" and "GW_A" do not support this configuration setting. At this point, it is sufficient to uncheck"160" from the bandwidth options in Figure 4.25. for reusable test suite.

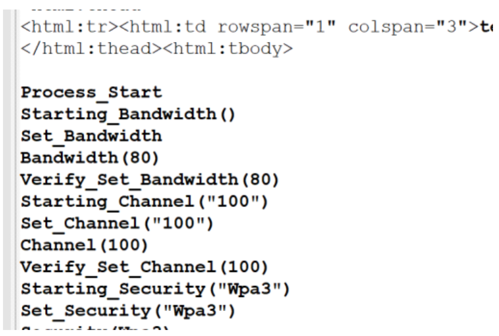

Ein MISTA 0.5
C:/Users/melihbugra.oguzer/Desktop/ps5/MISTA_TEST/MISTA1.0/Yeni_Proje
Select File
Generator Output
MISTA
HTML

☑ Bandwidth
- ☑ 20
- ☑ 40
- ☑ 80
- ☐ 160

☑ Channel
- ☑ 36
- ☑ 52
- ☑ 100

Figure 4.25. Scenario 1 GUI of EIN Tool

Figure 4.26 shows that in none of the test cases generated by MISTA after modeling is the bandwidth value set to "160". Thus, there is no need to create a suite again in the "EIN Tool" application. The previously created suite can be updated effortlessly.



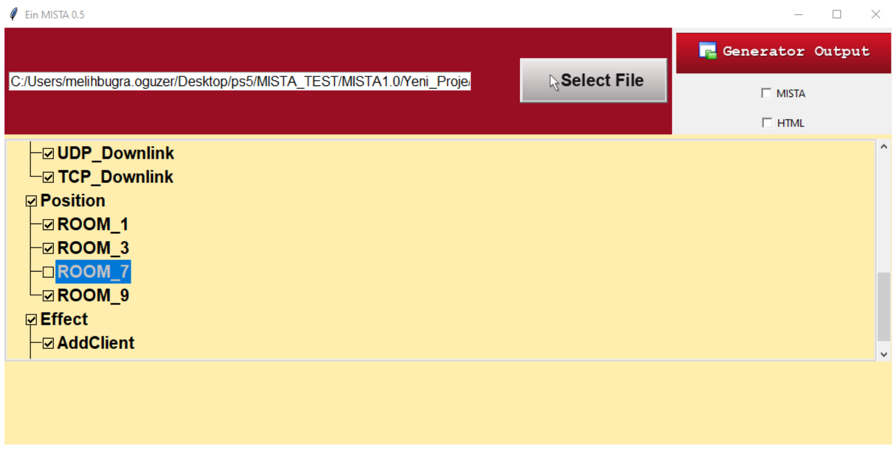
```
<html:tr><html:td rowspan="1" colspan="3">t
</html:thead><html:tbody>

Process_Start
Starting_Bandwidth()
Set_Bandwidth
Bandwidth(80)
Verify_Set_Bandwidth(80)
Starting_Channel("100")
Set_Channel("100")
Channel(100)
Verify_Set_Channel(100)
Starting_Security("Wpa3")
Set_Security("Wpa3")
```

Figure 4.26. Scenario 1 a part of test cases

4.2.2.1.2. Scenario 2

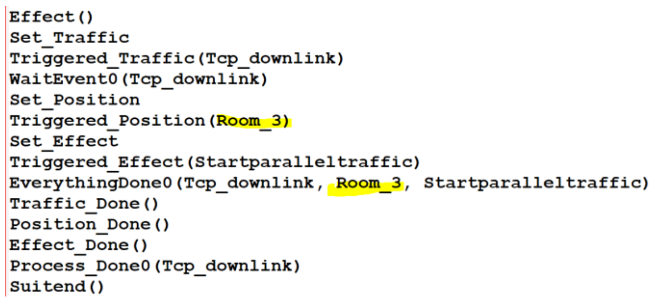
There are differences in the cost/use patterns of the devices. For example, the "GW_C" device has the latest technologies, while "GW_A" is a device for normal ADSL users. These cost-based preferences also apply to the performance of the devices. The "GW_C" and "GW_A" devices stably support the 5 GHz radio frequency. However, their stability distance is different. The position "ROOM _7", which is the client's distance point in Figure 4.27. for the reusable test suite, applies to the "GW_C" device but not to the "GW_A" device.


Ein MISTA 0.5
C:/Users/melihbugra.oguzer/Desktop/ps5/MISTA_TEST/MISTA1.0/Yeni_Proje
Select File
Generator Output
MISTA
HTML

```
☑UDP_Downlink
☑TCP_Downlink
☑Position
  ☑ROOM_1
  ☑ROOM_3
  ☐ROOM_7
  ☑ROOM_9
☑Effect
  ☑AddClient
```

Figure 4.27. GUI Sample of EIN Tool for Scenario 2

Figure 4.28. shows that in none of the test cases generated by MISTA after modeling is the position value set to "ROOM_7". Thus, there is no need to create a suite again in the "EIN Tool" application. The previously created suite can be updated effortlessly.

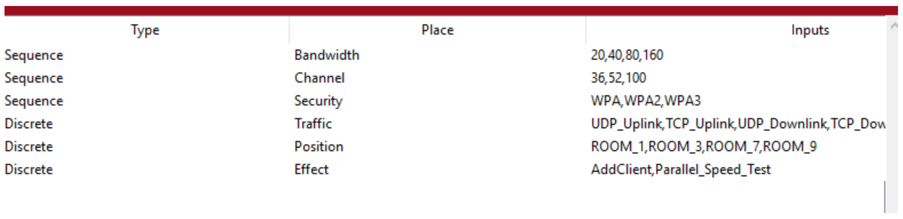


```
Effect()
Set_Traffic
Triggered_Traffic(Tcp_downlink)
WaitEvent0(Tcp_downlink)
Set_Position
Triggered_Position(Room_3)
Set_Effect
Triggered_Effect(Startparalleltraffic)
EverythingDone0(Tcp_downlink, Room_3, Startparalleltraffic)
Traffic_Done()
Position_Done()
Effect_Done()
Process_Done0(Tcp_downlink)
Suitend()
```

Figure 4.28. A part of test cases for scenario 2

4.2.2.1.3 Scenario 3

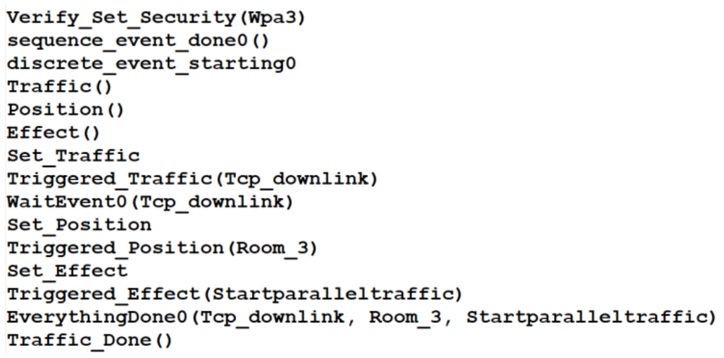
Test scenarios created with newly added integrations may change. It is important to be able to adapt quickly to these changes and to expend as little effort as possible. As part of the scenario, a new sub-function, "Speed Test", must be added to the suite prepared with "EIN Tool". At this point, the user can again create a suite as in Figure 4.29.



Type	Place	Inputs
Sequence	Bandwidth	20,40,80,160
Sequence	Channel	36,52,100
Sequence	Security	WPA,WPA2,WPA3
Discrete	Traffic	UDP_Uplink,TCP_Uplink,UDP_Downlink,TCP_Dow
Discrete	Position	ROOM_1,ROOM_3,ROOM_7,ROOM_9
Discrete	Effect	AddClient,Parallel_Speed_Test

Figure 4.29. New suit sample for scenario 3

Figure 4.30. The newly added integration "Speed Test" can be seen in the test cases created by MISTA. This integration is processed by the test management tool and the required script is executed.



```
Verify_Set_Security(Wpa3)
sequence_event_done0()
discrete_event_starting0
Traffic()
Position()
Effect()
Set_Traffic
Triggered_Traffic(Tcp_downlink)
WaitEvent0(Tcp_downlink)
Set_Position
Triggered_Position(Room_3)
Set_Effect
Triggered_Effect(Startparalleltraffic)
EverythingDone0(Tcp_downlink, Room_3, Startparalleltraffic)
Traffic_Done()
```

Figure 4.30. A part of test cases for scenario 3

4.2.3. Analysis of MISTA application

The "MISTA" application used in the research was also analyzed as part of the research. This analysis relates in particular to the stability and reliability of the test generation system. The analysis began with the scenario where all inputs to the Wireless Steering suite were selected. This process continued as the inputs were gradually reduced. As an alternative to modeling for comparison, a system was developed in which inputs that cannot be included in the suite as a brute force are extracted from the test generator html file containing all inputs. This process is visualized in detail in Figure 4.31.

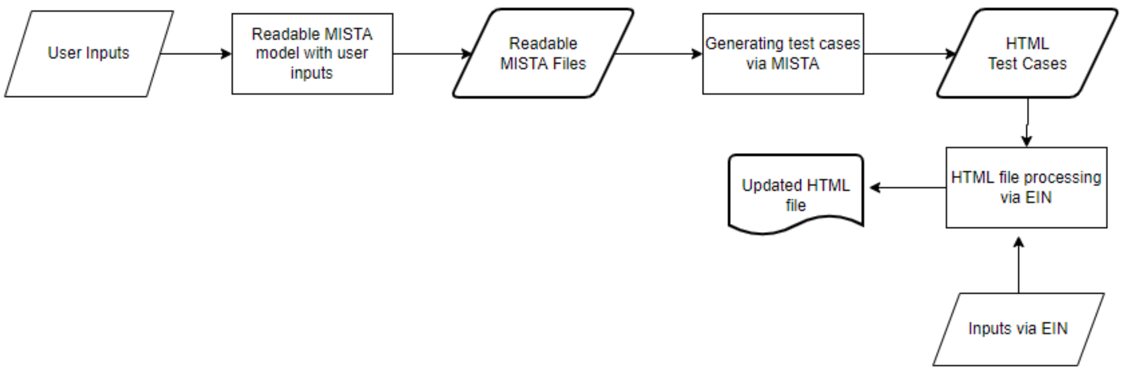

User Inputs
Readable MISTA model with user inputs
Readable MISTA Files
Generating test cases via MISTA
HTML Test Cases
HTML file processing via EIN
Updated HTML file
Inputs via EIN

Figure 4.31. Test Generator Process

As seen in Table 4.2, a total of 6 different events were created for this suite. The probability of events receiving a total of 20 inputs has been created. The number of test cases that can generate all these combinations is "20736".

Table 4.2 Full list of selected events.

	Input Number	Sequence	Discrete	MISTA Test Case	MISTA Test Generator Time	Test Output Time
1-Event	4	1	0			
2-Event	3	1	0			
3-Event	3	1	0			
4-Event	4	0	1			
5-Event	4	0	1			
6-Event	2	0	1	20736	6.511	6.105
Total	**20**					
MISTA Test Case	HTML Test Case					
20736	20736					

The subheadings below show the test cases that can be created on both sides as a result of each reduction cycle.

4.2.3.1. Cycle 1

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 19 inputs and test cases were combined. As you can see in Table 4.3, all test cases were created in less than 6 seconds in total, because the modeling was done via the reusable suite "EIN Tool".

Table 4.3 Test cases with 1 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth	20	0.60565 second	4.622 second
MISTA Total Case	Alternative Total Case		
15552	15552		

4.2.3.2. Cycle 2

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 18 inputs and test cases were combined. As you can see in Table 4.4, all test cases were created in less than 4 seconds in total, because the modeling was done via the reusable suite "EIN Tool"

Table 4.4 Test cases with 2 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth Channel	20 36	0.687359 second	3.095 second
MISTA Total Case	Alternative Total Case		
10368	10368		

4.2.3.3. Cycle 3

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 17 inputs and test cases were combined. As you can see in Table 4.5, all test cases were created in less than 2 seconds in total, because the modeling was done via the reusable suite “EIN Tool”.

Table 4.5 Test cases with 3 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth	20	0.696006 second	1.222 second
Channel	36		
Security	WPA		
MISTA Total Case	Alternative Total Case		
4608	6912		

4.2.3.4. Cycle 4

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 16 inputs and test cases were combined. As you can see in Table 4.6, all test cases were created in less than 2 seconds in total, because the modeling was done via the reusable suite "EIN Tool".

Table 4.6 Test cases with 4 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth	20	0.6348 second	1.023 second
Channel	36		
Security	WPA		
Traffic	UDP Uplink		
MISTA Total Case	Alternative Total Case		
3456	5185		

4.2.3.5. Cycle 5

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 15 inputs and test cases were combined. As you can see in Table 4.7, all test cases were created in less than 2 seconds in total, because the modeling was done via the reusable suite "EIN Tool".

Table 4.7 Test cases with 5 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth	20		
Channel	36		
Security	WPA	0.690676 second	0.953 second
Traffic	UDP Uplink		
Position	ROOM 1		
MISTA Total Case	Alternative Total Case		
2592	3889		

4.2.3.6. Cycle 6

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 14 inputs and test cases were combined. As you can see in Table 4.8, all test cases were created in less than 1 seconds in total, because the modeling was done via the reusable suite "EIN Tool".

Table 4.8 Test cases with 6 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth	20		
Channel	36	0.622375 second	0.306 second
Security	WPA		
Traffic	UDP Uplink		

Position	ROOM 1
Effect	Add Client
MISTA Total Case	Alternative Total Case
1296	1945

4.2.3.7. Cycle 7

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 13 inputs and test cases were combined. As you can see in Table 4.9, all test cases were created in less than 1 seconds in total, because the modeling was done via the reusable suite "EIN Tool".

Table 4.9 Test cases with 7 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth	20		
Channel	36		
Security	WPA		
Traffic	UDP Uplink	0.653108 second	0.274 second
Position	ROOM 1		
Effect	Add Client		
Bandwidth	40		
MISTA Total Case	Alternative Total Case		
864	1296		

4.2.3.8. Cycle 8

During the first cycle, only one input was decreased from the reusable suite. In total, 6 different events and 12 inputs and test cases were combined. As you can see in Table 4.10, all test cases were created in less than 1 seconds in total, because the modeling was done via the reusable suite "EIN Tool".

Table 4.10 Test cases with 7 input removed

Combination	Removed Input	Reusable Time	MISTA Test Generator Time
Bandwidth	20	0.6655386 second	0.27 second
Channel	36		
Security	WPA		
Traffic	UDP Uplink		
Position	ROOM 1		
Effect	Add Client		
Bandwidth	40		
Channel	52		
MISTA Total Case	Alternative Total Case		
432	648		

CHAPTER 5

EXPERIMENTAL RESULTS

Within the research, we study this topic in 3 main sections, as the analyzes are diverse and based on different cause-effect relationships.

1. Analysis of Wireless Mesh Network Performance Analysis
2. Analysis of MISTA application
3. The interview of the tester user

5.1. Wireless Mesh Network Performance Analysis

Tests were performed regularly for two months using the tool created as part of the research. The devices used for the tests are listed in Table 3.0 and Table 3.1. The test results are examined under 3 different headings.

- GW_A (support wifi-4)
- GW_B (support wifi-5)
- GW_C (support wifi-6)

The "Steering" test suite modeled with "EIN Tool" was reused according to the different requirements of each device. The test environment is designed not to interfere with the radio frequencies of the devices. Since the test environment is an expensive and actively used environment, the test cases executed as part of the research are not sufficient. However, even with this limited number of test executions, satisfactory results were obtained with respect to the expectations of the research. During the research, two important elements were highlighted in the test results.

1) Defect rates that can be detected by test scenarios created via PN Model outside of the currently running test scenarios.

2) Identifying previously unused test cases that can be created by the "Steering" test scenario modeled with PN and improving the past test cases.

As part of the study, 12 tests were executed with each gateway model. These executions were performed in different time periods. However, the tests were executed in the same environment and at the same distance between devices. The collected test logs were not included in the paper because they contain critical information within the company. The labeling "SC" in the column with the test results in the prepared table shows that the result is the same as that of the previous automation test cycle or that it could not detect any bug. In addition, detected bugs or new scenario techniques are indicated with code names in the "Description" column.

5.1.1. GW(A) Execution Results

In the tests running on the GW_A coded gateway, the tests were run only at 40 MHz bandwidth. In addition, parallel traffic and uplink UDP traffic type could not be executed due to device limitations. All test results are shown in Table 5.1.

Table 5.1 GW(A) execution results

	Bandwidth	Channel	Security Type	Position	Effect	Test Result	Description
Test Case 1	40	36	WPA_2	ROOM_3	X	SC	
Test Case 2	40	36	WPA_2	ROOM_3	X	FAIL	DES-1
Test Case 3	40	36	WPA_2	ROOM_4	X	SC	
Test Case 4	40	36	WPA_2	ROOM_4	X	SC	
Test Case 5	40	36	WPA_2	ROOM_7	X	SC	
Test Case 6	40	36	WPA_2	ROOM_7	X	SC	
Test Case 7	40	100	WPA_2	ROOM_7	X	SC	
Test Case 8	40	100	WPA_2	ROOM_7	X	SC	
Test Case 9	40	100	WPA_2	ROOM_9	X	SC	
Test Case 10	40	100	WPA_2	ROOM_9	X	SC	
Test Case 11	40	100	WPA_2	ROOM_10	X	SC	
Test Case 12	40	100	WPA_2	ROOM_10	X	SC	

During the tests, a minor bug labeled DES -1 was found. The client was moved to a different position point that was not previously used in the Steering Automation tests. During the test, the client remained at the limit of an RSSI level where the steering mechanism was activated based on its position. As a result, the client's throughput fluctuated more than average while steering.

However, this problem which was discovered does not have a catastrophic side to it. Once the bug was analyzed, it was quickly discovered that it only occurred in a mobile phone operating system. As a result of the analysis made by another team, it was determined that there was a general problem experienced in mobile devices with the same operating system. During the other 11 tests which were made, no unusual events occurred.

5.1.2. GW(B) Execution Results

In the tests running on the GW_B coded gateway, the tests were run only at 80 MHz bandwidth ranges due. All test results are shown in Table 5.2.

Table 5.2 GW(B) execution results

	Bandwidth	Channel	Security Type	Position	Effect	Test Result	Description
Test Case 1	80	36	WPA_2	ROOM_3	X	SC	
Test Case 2	80	36	WPA_2	ROOM_3	X	SC	
Test Case 3	80	36	WPA_2	ROOM_4	X	SC	
Test Case 4	80	36	WPA_2	ROOM_4	X	SC	
Test Case 5	80	36	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 6	80	36	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 7	80	100	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 8	80	100	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 9	80	100	WPA_2	ROOM_9	X	SC	
Test Case 10	80	100	WPA_2	ROOM_9	X	SC	
Test Case 11	80	100	WPA_2	ROOM_10	X	SC	
Test Case 12	80	100	WPA_2	ROOM_10	X	SC	

There were no unusual events in all of the tests. The detected bugs were also detected in the tests without PN modeling.

5.1.3. GW(C) Execution Results

In the tests running on the GW_C coded gateway, the tests were run only at 160 MHz bandwidth ranges due. All test results are shown in Table 5.3.

Table 5.3 GW(C) execution results

	Bandwidth	Channel	Security Type	Position	Effect	Test Result	Description
Test Case 1	160	36	WPA_2	ROOM_3	X	SC	
Test Case 2	160	36	WPA_2	ROOM_3	X	SC	
Test Case 3	160	36	WPA_2	ROOM_4	X	SC	
Test Case 4	160	36	WPA_2	ROOM_4	X	SC	
Test Case 5	160	36	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 6	160	36	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 7	160	100	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 8	160	100	WPA_2	ROOM_7	Parallel Traffic	SC	
Test Case 9	160	100	WPA_2	ROOM_9	X	SC	New_1
Test Case 10	160	100	WPA_2	ROOM_9	X	SC	
Test Case 11	160	100	WPA_2	ROOM_10	X	SC	
Test Case 12	160	100	WPA_2	ROOM_10	X	SC	

During all the testing, no unusual events were detected. However, due to the executing test case number 9, another test scenario attracted attention. In this scenario, the point "ROOM _9", which is not used much under normal conditions, was used for the steering test. However, during the test, an environment-based problem occurred and an access point that needed to be closed was not turned off. The combination of these two factors resulted in the discovery of a new steering test case that was not present in the previous test procedures. Based on this newly discovered test case. Similar test cases were also created. These test cases were also reported to the telecommunications company that owns the products used in the research.

5.2. Analysis of MISTA Tool

As a result of the comparisons, stability problems were detected in test generation processes with the MISTA tool in high-combination situations. As in the graphic shown in Figure 4.7, after a certain cycle, the difference is seen in the reusable test generation system that we produce to compare with the test cases MISTA Tool produces after each modeling process.

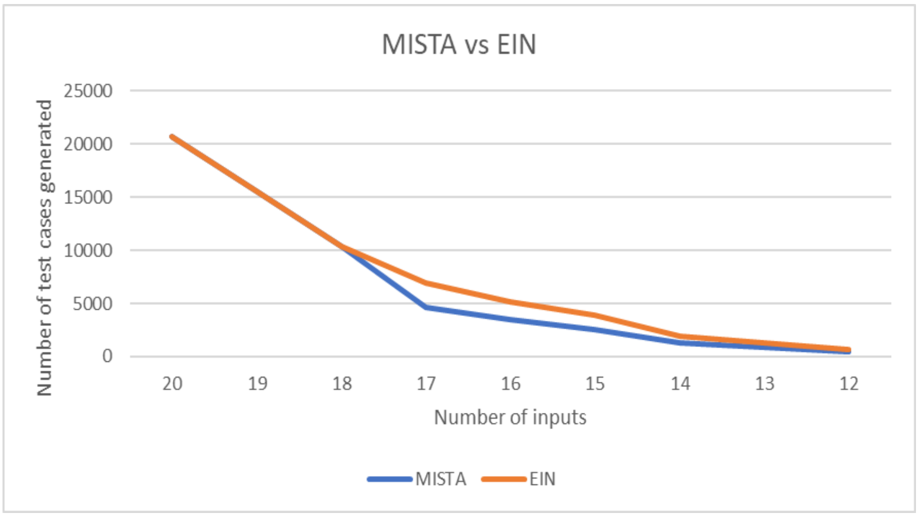


Figure 5.1. MISTA and EIN Tool test cases comparison

When this difference was analyzed, it was determined that some of the tests created from a model with 6 events and 20 inputs by MISTA were duplicates. It has been determined that the test cases numbered "20408" and "20216" are exactly the same in the test case file created with the same model by MISTA Tool. This duplicate is shown in Appendix E. Because of this duplication, the system that reusable updates test cases via html with EIN Tool has more cases than the MISTA tool in certain combinations.

When these duplicates were analyzed, they were found to be directly proportional to the number of events. It was observed that these duplicates increased as the complexity of the model increased.

In addition, the relationship between the number of events and test generate time of the "MISTA" application was also analyzed. As can be seen in Figure 5.2. as a result of this analysis, the test generation time of a model with 20 events was over 6 seconds. As the number of events decreases, the test generates times seen to decrease.

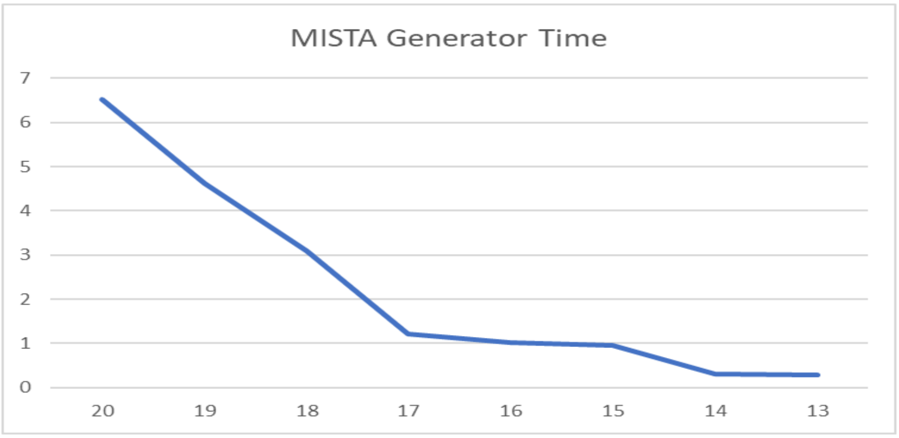


Figure 5.2. MISTA Tool Time relationship between number of events and test cases

5.3. Survey of Tester Users

Survey was conducted with actively working test engineers and technicians about the reusable use of test automation. As a result of this survey, the advantages and disadvantages of reusable suites were identified. The testers are anonymous and each tester receives an "ID". In addition, the companies for which the testers work are different. Detailed information about the testers can be found in Table 5.4.

Table 5.4 Participant details

Tester ID	Tester Title
US_1	Test Engineer
US_2	Experienced Test Engineer
US_3	Experienced Test Engineer
US_4	Test Technician

US_5	Automation Lead
US_6	Programmer
US_7	Test Engineer
US_8	Senior Team Leader
US_9	Computer Engineer
US_10	Test Engineer

During the survey, participants were asked about the reusable test suite and the tool developed as part of the research. During the survey, a scale between 1 and 10 was set for the answers to the questions up to the 6th question. For the 7th question, a percentage value was requested from the participants.

1. How much does the Reusable Test Suite facilitate your work?
2. To what extent do you believe that the test approach processes in your company can detect bugs?
3. Do you think that the scenarios that you will create during the testing processes may be useful?
4. Do you think you have enough time to implement the test scenarios you have planned?
5. If you had the opportunity to create or use tests using modeling during your working life, how satisfied were you?
6. How concerned are you about the learning and cost curve of the modeling process?
7. To what extent do you think there is a bug leak at the end of the scenarios created with the testing approach in your company?

This interview with users in the research was conducted online in accordance with the remote working rules due to corona virus

The consent form was taken from the participants before participating in the survey. Interview results can be seen in Table 5.5.

Table 5.5 Interview results

	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6	Question 7
US_1	7	9	8	4	Not	1	%5
US_2	10	8	5	3	10	2	%20
US_3	8	8	8	7	Not	3	%20
US_4	8	7	8	8	Not	2	%10
US_5	8	7	8	4	Not	3	%25
US_6	6	5	9	6	Not	2	%30
US_7	6	5	9	6	Not	2	%30
US_8	8	5	8	5	7	8	%15
US_9	7	9	6	3	Not	4	%10
US_10	6	6	6	4	Not	4	%15

As a result of the interviews, it is seen that the use of modeling in test automation processes is still insufficient. Clearly most of the participants did not use modeling in any of their working experiences. In another noted situation, it was found that participants gave an average of “7.4” points on the first question. According to this result, participants showed a positive attitude towards reusable test suites.

Another situation indicated by the participants is their desire to participate in the testing processes in a dynamic way. This situation also means that participants want to incorporate their own experiences into the tests too. Contrary to the research findings, it appears that most participants did not hesitate in the modeling learning processes, which relates to the 6th question. However, it is another contradiction that the same participants were complaining about the 4th question due to lack of time.

Finally, considering the answers to the 2nd and 7th questions, it is seen that the participants do not fully trust the current test cycles. It is seen that the participants think that there are undetected bugs at a rate of "18%" on average from the products coming out of the cycle.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1. Conclusion

Today, the QA department of any company that wants to be globally competitive must implement test automation processes. However, in some fields, the budgets and costs for automation processes are very high. At this point, it is necessary to include efficiency in test automation cycles.

One of the areas that meets these criteria is the companies that produce user-facing products for wireless networks. Being able to adapt to rapidly developing internet infrastructure and new developing technologies. This is very important for ISP companies that are customers of companies that operate in this space. The expectation creates fast and efficient technology adaptations along with customer requirements. To meet this expectation, fast QA cycles are essential. Departments should involve methods which speed up these processes as budgets for setting up and using test automation in this space are very high as well.

At this point, our research proposes to model the specific testing processes of the products with Petri nets and generate test scenarios from this model. Within the scope of the research, the automation test process of "Steering", one of the functions of Wireless Mesh Networks, was modeled with Petri nets. A total of "20736" tests were generated from the model created using the MISTA application. From these test cases, 12 test cases of the same standards for 3 different types of devices were selected and tested in the test environment. Due to the high cost of using the test environment, a total of "36" test cases were executed for 3 groups of devices. As a result of these 36 tests, "1" minor bug was detected. Also, "1" test scenario was discovered that was unpredictable.

It was determined by the end of the process that the model, created process could produce a lot of test cases but these cases were not quite efficient, especially in high-cost test environments. However, it was noticed that if the relations between the events could be increased during the process, more pinpoint detections would be made. At this point, it can be seen that the process is not very successful.

Another issue covered within the scope of the research is about bringing petri-nets to a more efficient and reusable point. Herein, it has been seen that it can increase efficiency in uncomplicated test processes consisting of the tool, sequence and discreate events we have created. Once prepared, it has been determined that the test scenario can work in a way to increase efficiency in different device types and requirements under test. However, it has been noticed that as the complexity of this process increases, the quality of the generated tests may decrease too. At this point, it has been determined that the developed tool is suitable for improvement.

6.2. Future Work

Based on the information obtained at the end of the research and the feedback from the users, some new ideas were developed and issues that need to be addressed in the future of the project were identified. In this context;

1) It is planned to make the control requirements of the reusable test suites prepared for the devices more intelligent. In this regard, it is planned that this test suite will update itself according to the usage rate of the devices in the field, thanks to the cloud integration.

2) To further increase the efficiency of the products in the automation processes, a separate branching according to the main chip components used in the devices is planned. In this way, a cost-oriented and dynamic test process will be created that can also function as a smoke test. It is assumed that this structure can also reduce the "pesticide" paradox effect.

3) Generation of test scenarios with the model via "EIN Tool". Adaptation of the open-source codes of the application "MISTA" to the application "EIN Tool" created during the research.

REFERENCES

Anbunathan, R, and Anirban Basu. 2017. "Automation Framework For Test Script Generation For Android Mobile". *2017 2Nd IEEE International Conference On Recent Trends In Electronics, Information &Amp; Communication Technology (RTEICT)*. doi:10.1109/rteict.2017.8256930.

Ayub, Shahid, Sharadha Kariyawasam, Mahsa Honary, and Bahram Honary. 2013. "A Practical Approach Of VLC Architecture For Smart City". *2013 Loughborough Antennas &Amp; Propagation Conference (LAPC)*. doi:10.1109/lapc.2013.6711862.

Beizer, Boris. 2003. *Software Testing Techniques*. New Delhi: Dreamtech.

Bianchi, G. 2000. "Performance Analysis Of The IEEE 802.11 Distributed Coordination Function". *IEEE Journal On Selected Areas In Communications* 18 (3): 535-547. doi:10.1109/49.840210.

David, René, and Hassane Alla. 2014. *Discrete, Continuous, And Hybrid Petri Nets*. Berlin: Springer Berlin.

Devi, T.R. (2012). Importance of Testing in Software Development Life Cycle.

DJEMAME, K. 2001. "DISTRIBUTED SIMULATION OF HIGH-LEVEL ALGEBRAIC PETRI NETS WITH LIMITED CAPACITY PLACES". *Parallel Algorithms And Applications* 16 (3): 207-241. doi:10.1080/01495730108935272.

Fernández-Sáez, Ana M., Michel R. V. Chaudron, and Marcela Genero. 2018. "An Industrial Case Study On The Use Of UML In Software Maintenance And Its Perceived Benefits And Hurdles". *Empirical Software Engineering* 23 (6): 3281-3345. doi:10.1007/s10664-018-9599-4.

Fragal, Vanderson Hafemann, Adenilso Simao, Andre Takeshi Endo, and Mohammad Reza Mousavi. 2017. "Reducing The Concretization Effort In FSM-Based Testing Of Software Product Lines". *2017 IEEE International Conference On Software Testing, Verification And Validation Workshops (ICSTW)*. doi:10.1109/icstw.2017.61.

Henty, B.E. (2001). Throughput Measurements and Empirical Prediction Models for IEEE 802.11b Wireless LAN (WLAN) Installations.

Hessel, Anders, and Paul Pettersson. 2007. "Model-Based Testing Of A WAP Gateway: An Industrial Case-Study". *Formal Methods: Applications And Technology*, 116-131. doi:10.1007/978-3-540-70952-7_8.

Hujsa, Thomas, and Raymond Devillers. 2018. "On Deadlockability, Liveness And Reversibility In Subclasses Of Weighted Petri Nets". *Fundamenta Informaticae* 161 (4): 383-421. doi:10.3233/fi-2018-1708.

Lopez, A.G., & Viela, I. 2012. Automated Telecommunication Software Testing : An automated model generator for Model-Based Testing.

Miczulski, Piotr, and Marian Adamski. 2006. "ANALYSIS OF SAFENESS, LIVENESS AND PERSISTENCE PROPERTIES OF PETRI NETS BY MEANS OF MONOTONE LOGIC FUNCTIONS". *IFAC Proceedings Volumes* 39 (17): 137-142. doi:10.3182/20060926-3-pl-4904.00023.

Murata, T. 1989. "Petri Nets: Properties, Analysis And Applications". *Proceedings Of The IEEE* 77 (4): 541-580. doi:10.1109/5.24143.

Unal Ibrahim, 2008. “Timed discrete event systems and applications”

Utting, Mark, Alexander Pretschner, and Bruno Legeard. 2011. "A Taxonomy Of Model-Based Testing Approaches". *Software Testing, Verification And Reliability* 22 (5): 297-312. doi:10.1002/stvr.456.

Xu, Dianxiang. 2011. "A Tool For Automated Test Code Generation From High-Level Petri Nets". *Applications And Theory Of Petri Nets*, 308-317. doi:10.1007/978-3-642-21834-7_17.

Meddour, D.-E., A. Kortebi, and R. Boutaba. 2010. "Mesh-Based Broadband Home Network Solution: Setup And Experiments". *2010 IEEE International Conference On Communications*. doi:10.1109/icc.2010.5502620.

Nouman Zafar, Muhammad, Wasif Afzal, Eduard Paul Enoiu, Athanasios Stratis, and Ola Sellin. 2021. "A Model-Based Test Script Generation Framework For Embedded Software". *2021 IEEE International Conference On Software Testing, Verification And Validation Workshops (ICSTW)*. doi:10.1109/icstw52544.2021.00041.

Offutt, Jeff, and Sunitha Thummala. 2018. "Testing Concurrent User Behavior Of Synchronous Web Applications With Petri Nets". *Software &Amp; Systems Modeling* 18 (2): 913-936. doi:10.1007/s10270-018-0655-8.

Pizziol, Sergio, Catherine Tessier, and Frédéric Dehais. 2014. "Petri Net-Based Modelling Of Human–Automation Conflicts In Aviation". *Ergonomics* 57 (3): 319-331. doi:10.1080/00140139.2013.877597.

Sommerville, Ian. 2016. *Software Engineering*. Boston [etc]: Pearson.

APPENDICES

APPENDIX A

TV PANEL AUTOMATION MODEL OF PN

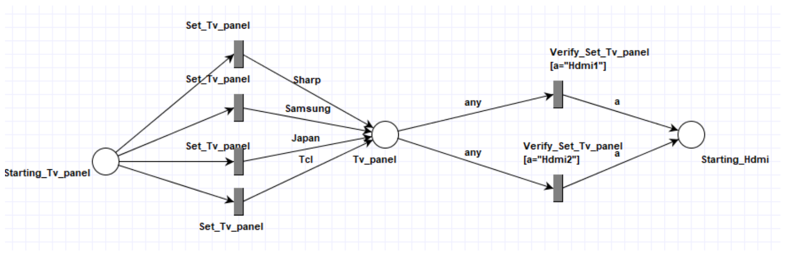


Figure A.1. *Set Panel* of Event TV Automation PN

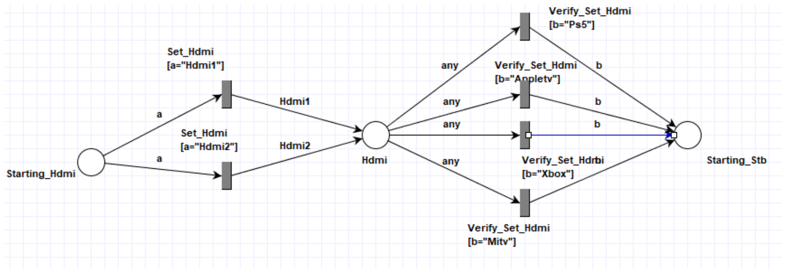


Figure A.2. Set HDMI Input of Event TV Automation PN

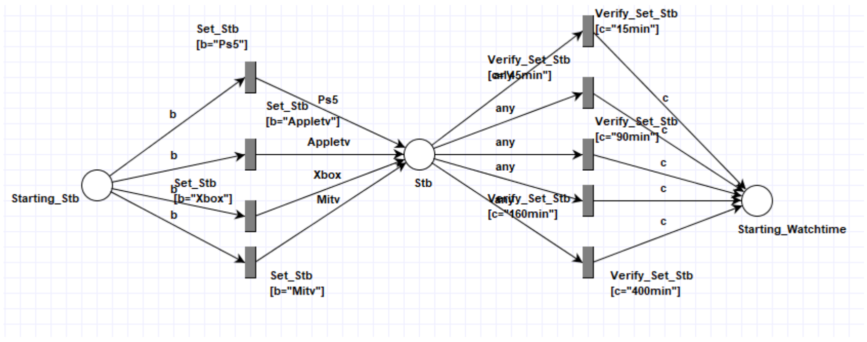


Figure A.3. Set STB of TV Automation PN Model

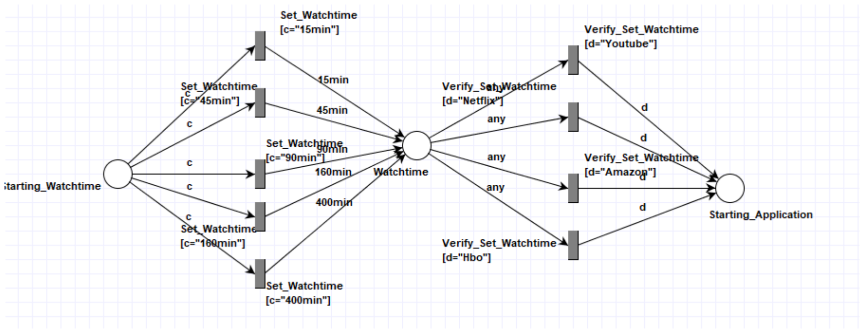


Figure A.4. Set Watch Time Event of TV Automation PN Model

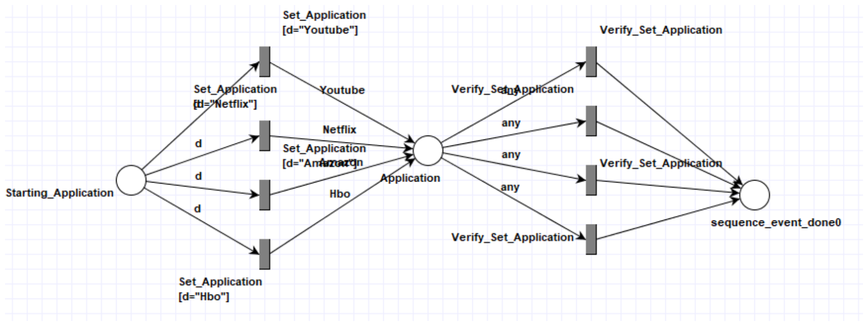


Figure A.5. Select On-Demand Services Event of TV Automation PN Model

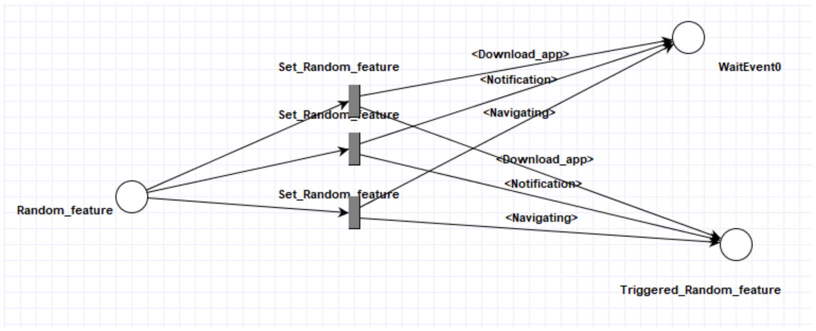


Figure A.6. Set Random Feature of TV Automation PN Model

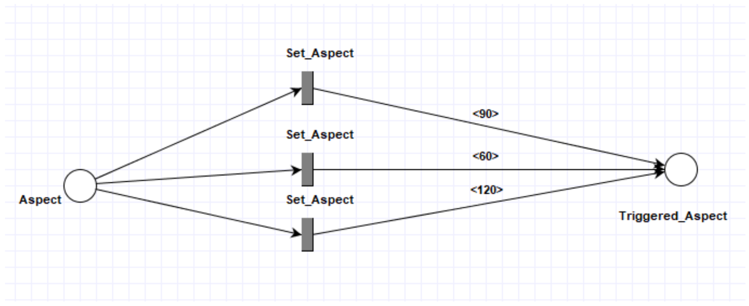


Figure A.7. Set Aspect Feature of TV automation PN Model

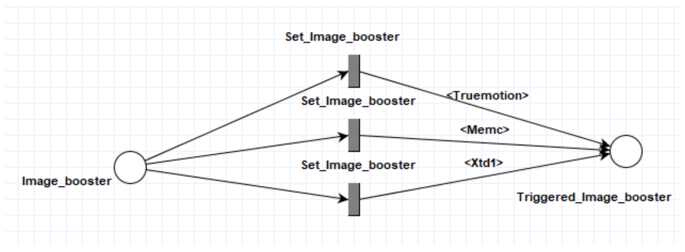


Figure A.8. Set Image Booster of TV automation PN Model

APPENDIX B

LAPTOP MANUFACTURER MODEL OF PN

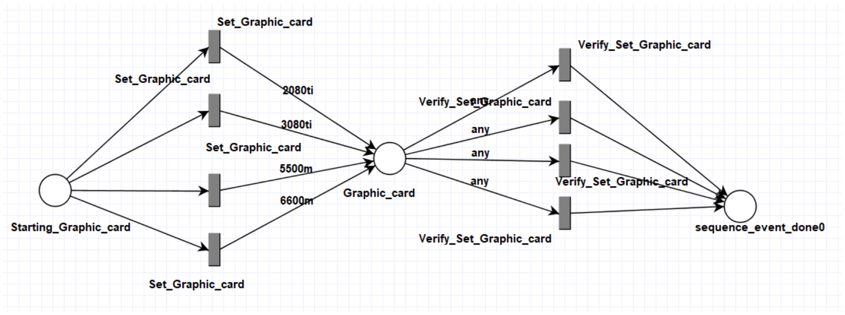


Figure B.1. Set Graphic Card Model of Laptop Manufacturer PN Model

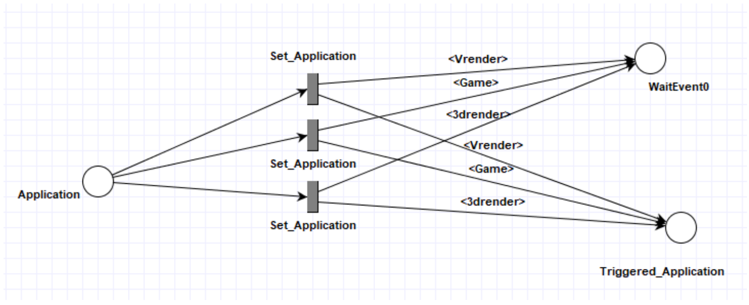


Figure B.2. Set Application Model of Laptop Manufacturer PN Model

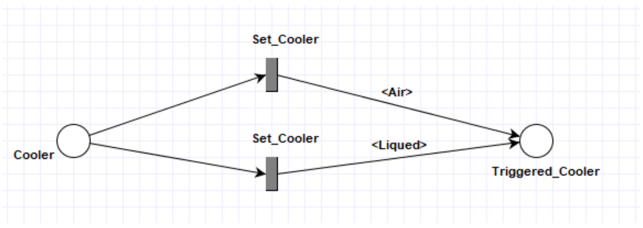


Figure B.3. Set Cooler Type Model of Laptop Manufacturer Model

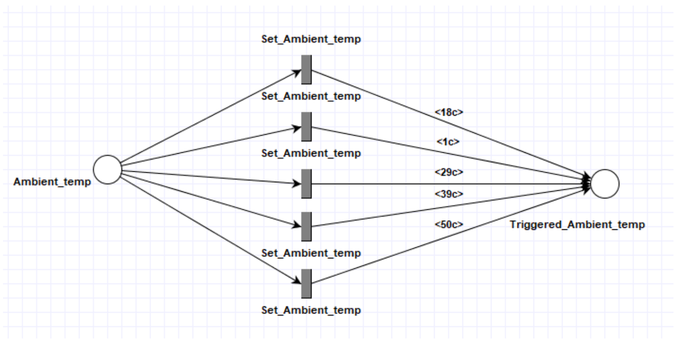


Figure B.4. Set Environment Temperature Model of Laptop Manufacturer Model

APPENDIX C

AUTOMOTIVE GEARBOX MODEL OF PN

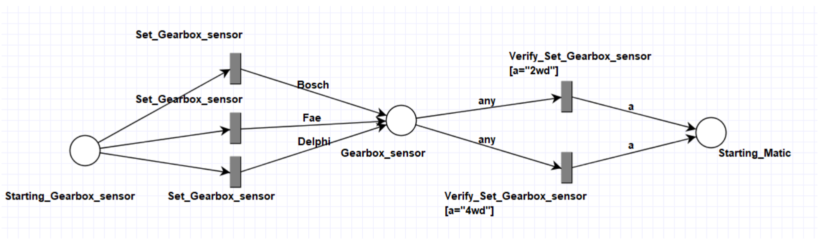


Figure C.1. Set Gearbox Sensor Model of Automotive Gearbox Model

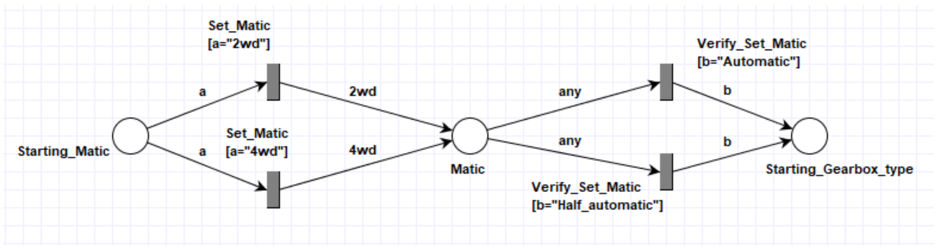


Figure C.2. Set Matic Type Model of Automotive Gearbox Model

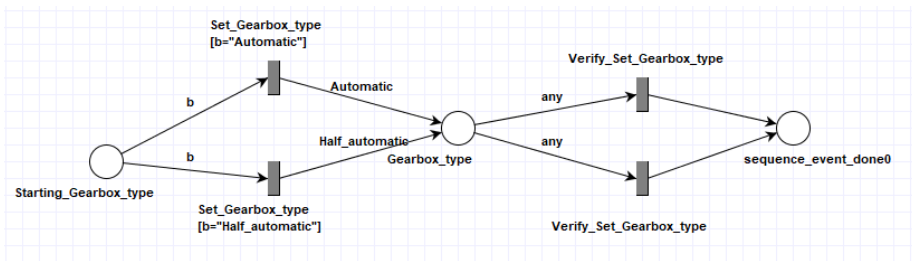


Figure C.3. Set Gear Type Model of Automotive Gearbox Model

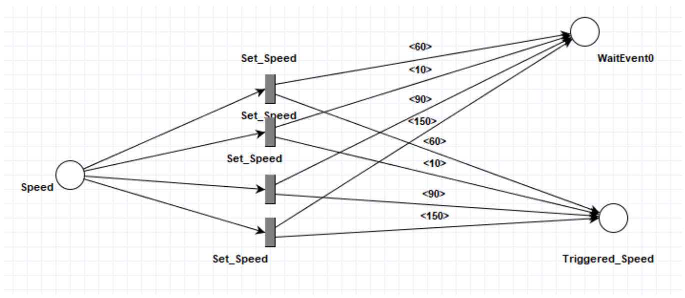


Figure C.4. Set Environment Speed Model of Automotive Gearbox Model

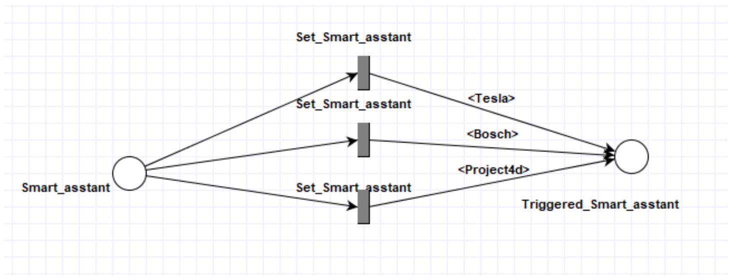
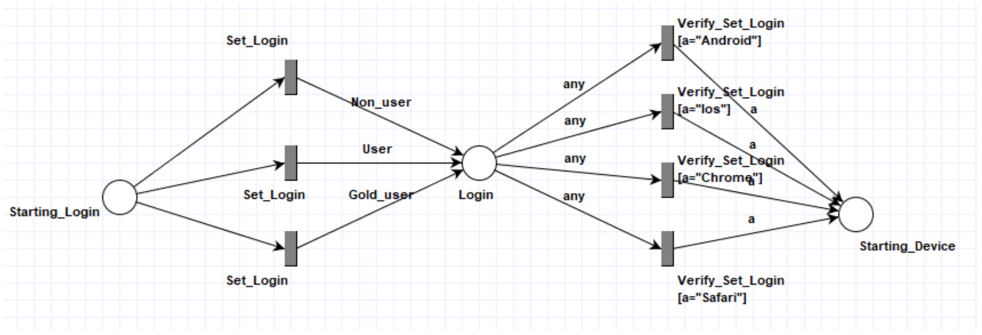


Figure C.5. Set Steer Assistant of Automotive Gearbox Model

APPENDIX D

WEB SITE MODEL OF PN



Set_Login
Verify_Set_Login [a="Android"]
Non_user
any
Verify_Set_Login [a="Ios"]
a
any
User
a
any
Verify_Set_Login [a="Chrome"]
Starting_Login
Set_Login
Gold_user
Login
any
a
a
Starting_Device
Set_Login
Verify_Set_Login [a="Safari"]

Figure D.1. Set User Type Model of WEB Site

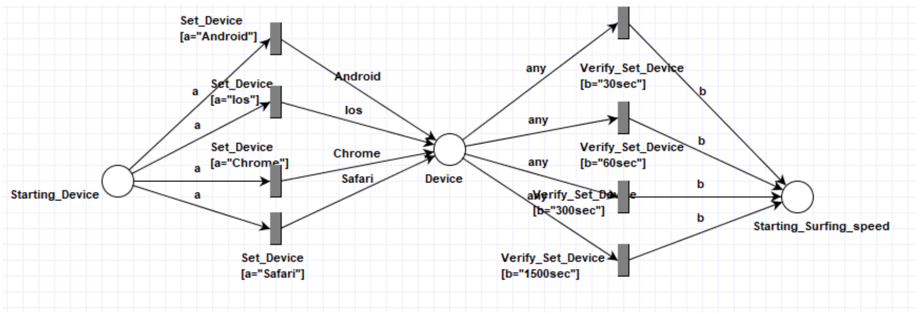


Figure D.2. Set Access Way Model of WEB Site

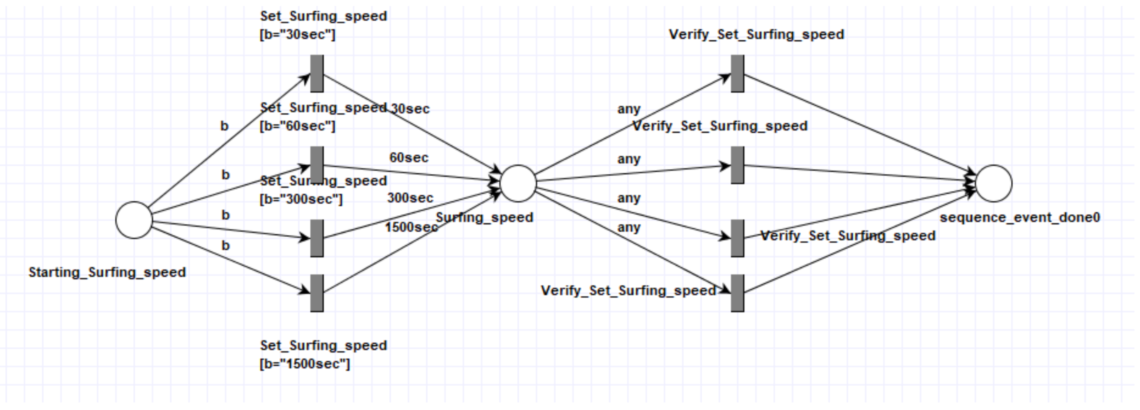


Figure D.3. Set Page Refresh Speed Model of WEB Site

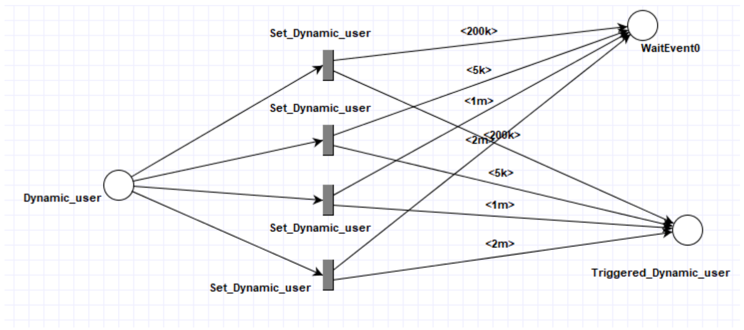


Figure D.4. Set Total of Active User Model of WEB Site

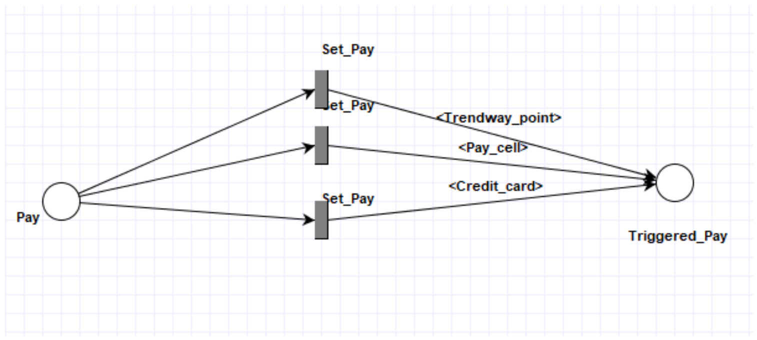
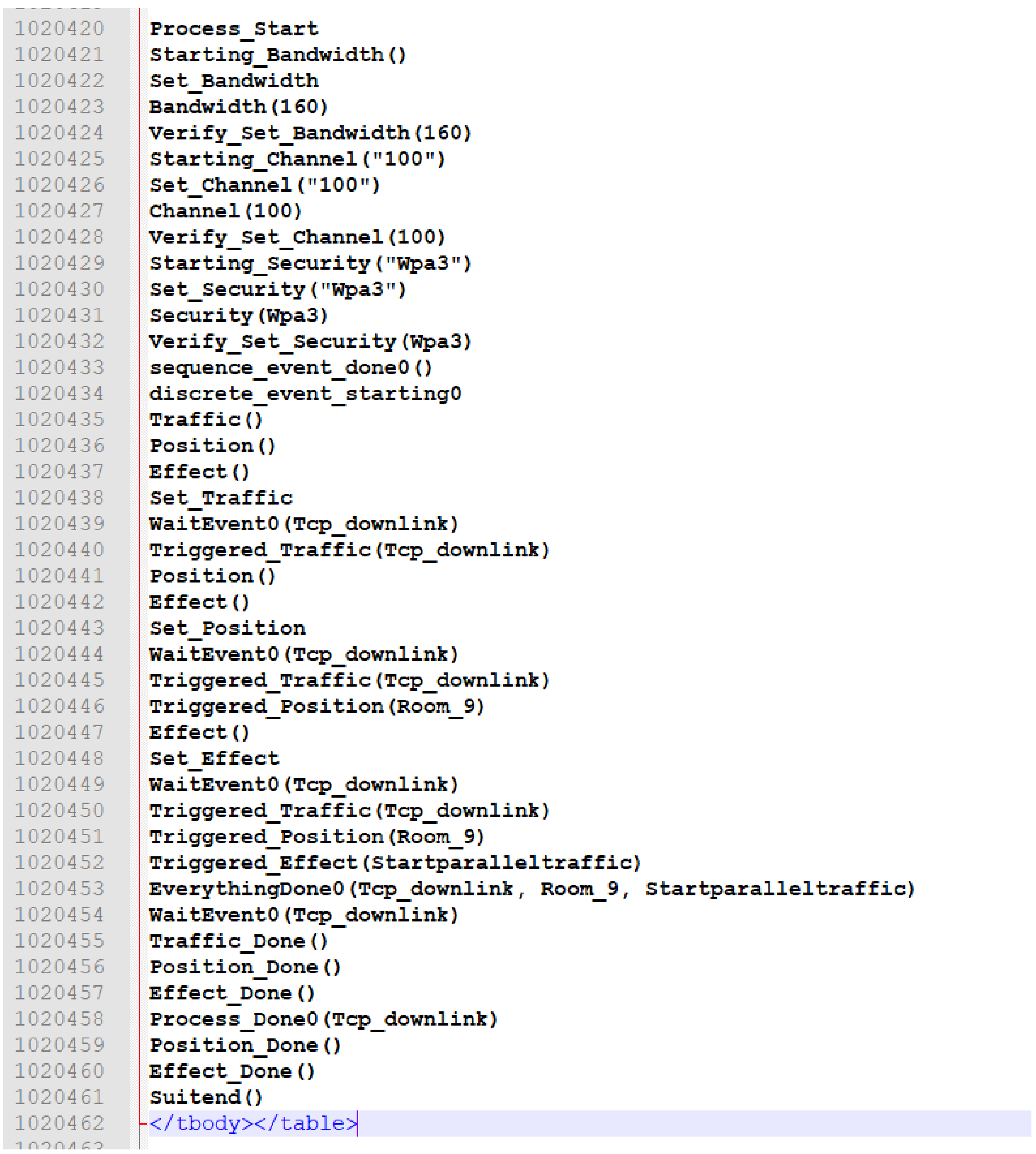


Figure D.5. Set Payment Method Model of WEB Site

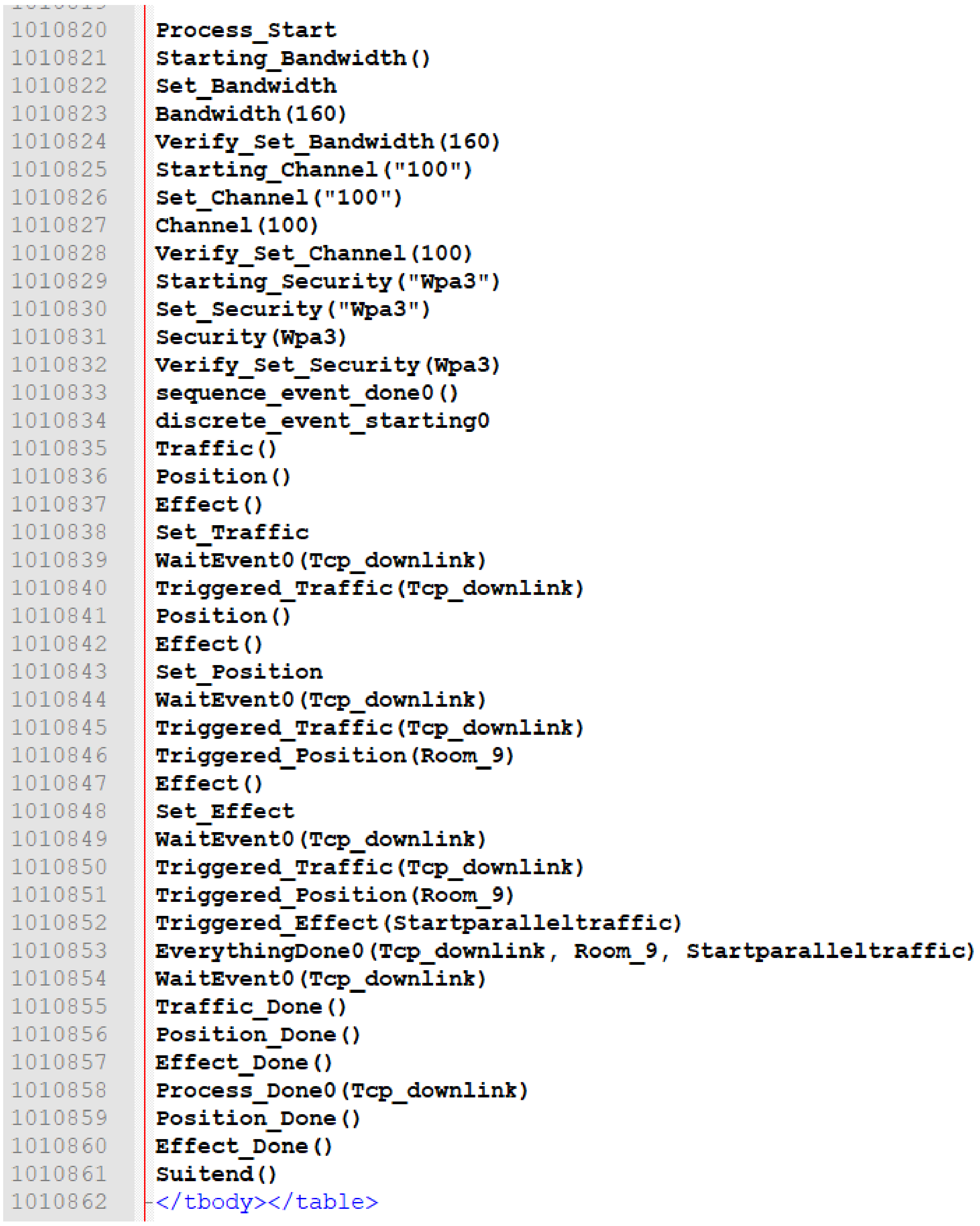
APPENDIX E

DUBLICATE TEST CASES



```
Process_Start
Starting_Bandwidth()
Set_Bandwidth
Bandwidth(160)
Verify_Set_Bandwidth(160)
Starting_Channel("100")
Set_Channel("100")
Channel(100)
Verify_Set_Channel(100)
Starting_Security("Wpa3")
Set_Security("Wpa3")
Security(Wpa3)
Verify_Set_Security(Wpa3)
sequence_event_done0()
discrete_event_starting0
Traffic()
Position()
Effect()
Set_Traffic
WaitEvent0(Tcp_downlink)
Triggered_Traffic(Tcp_downlink)
Position()
Effect()
Set_Position
WaitEvent0(Tcp_downlink)
Triggered_Traffic(Tcp_downlink)
Triggered_Position(Room_9)
Effect()
Set_Effect
WaitEvent0(Tcp_downlink)
Triggered_Traffic(Tcp_downlink)
Triggered_Position(Room_9)
Triggered_Effect(Startparalleltraffic)
EverythingDone0(Tcp_downlink, Room_9, Startparalleltraffic)
WaitEvent0(Tcp_downlink)
Traffic_Done()
Position_Done()
Effect_Done()
Process_Done0(Tcp_downlink)
Position_Done()
Effect_Done()
Suitend()
</tbody></table>
```

Figure E.1 Test Case 20408



```
Process_Start
Starting_Bandwidth()
Set_Bandwidth
Bandwidth(160)
Verify_Set_Bandwidth(160)
Starting_Channel("100")
Set_Channel("100")
Channel(100)
Verify_Set_Channel(100)
Starting_Security("Wpa3")
Set_Security("Wpa3")
Security(Wpa3)
Verify_Set_Security(Wpa3)
sequence_event_done0()
discrete_event_starting0
Traffic()
Position()
Effect()
Set_Traffic
WaitEvent0(Tcp_downlink)
Triggered_Traffic(Tcp_downlink)
Position()
Effect()
Set_Position
WaitEvent0(Tcp_downlink)
Triggered_Traffic(Tcp_downlink)
Triggered_Position(Room_9)
Effect()
Set_Effect
WaitEvent0(Tcp_downlink)
Triggered_Traffic(Tcp_downlink)
Triggered_Position(Room_9)
Triggered_Effect(Startparalleltraffic)
EverythingDone0(Tcp_downlink, Room_9, Startparalleltraffic)
WaitEvent0(Tcp_downlink)
Traffic_Done()
Position_Done()
Effect_Done()
Process_Done0(Tcp_downlink)
Position_Done()
Effect_Done()
Suitend()
</tbody></table>
```

Figure E.2 Test Case 20216