CHAPTER 5

DESIGN OF PETRINET MODEL

5.1 INTRODUCTION

The initial phase in the development of any Software Project activity is the establishment of requirements which is to be correct, complete and consistent. The requirements should include composition with values and results for the system or object under consideration.

Component C1 can be considered as a triple of $C_1 = (F, I, P, T_a)$

Where $C_1$ is the component

$F$ is the set of function in the component

$P$ is the set of parameter passed to the component

$T_a$ is the activation time.

$C = < (f_1, f_2, ... f_i), I/O, (p_1, p_2, ... p_j), (T_a) >$

A component is considered as safe if it is composed of towards expected functionality with the other needed components for the task submitted at that time. A component is considered unsafe, if it is composed with components of unexpected functionality. The composition activity is based on many read and selection techniques across the available interfaces during the point of time. It can also be seen that the untimely activation of the necessary component via its interfaces and idle time when a critical activity is being performed is unsafe not only for the software but also for the system...
where it is being activated. Composability can be viewed as the adherence or acceptance of the software module with other modules in or other platforms for effective parameter passing. Composability is dealt with both synchronous (Synchronous components is one in which all components in the system change their state variables simultaneously) and asynchronous components (In asynchronous one component changes its state at each time point). Composition of its components can be expressed as Petri Net which is used in real time application with timing requirements.

The key research question raised here is

1) How to simulate Petri Net model.

2) How test sequences can be generated from a model.

3) How to automate test sequences and generate code from model.

4) How to generate test coverage

The answer to first research question is the development of Petri Net components. A software system under test consists of black box components which can be represented as Petri Net. Model based testing takes UML which is a proclaimed standard by OMG and Petri Net facilitates the construction of behavioural models early in a development lifecycle, thus exposing ambiguities in the specification and design. Petri Net is a behaviour model which consists of a collection of directed arcs connecting places and transitions. Places may hold tokens. The state or marking of a net is its assignment of tokens to places.

The answer to second and third research question of how to automate test generation specifies the use of a tool named CPN which automate the test sequence from Petri Net. For a software system specified as
Petri Net, a test generator may attempt to generate test cases that execute all the transitions and places of the Petri Net model. Test case generation is a most challenging and an extensively researched activity. A test sequence is a high level test where a sequence of tasks or operations is directly generated from a high level behavioural model according to a particular test objective. Test generation can be done either manually or automatically. Recent research focus is towards automation of test generation because of the time constraints for testing. Subsequently coding is generated from automated timed automata with coverage analysis. This chapter was done based on the publication done in Asian Journal of Information Technology, 2016.

Limitations of Petrinet

- Models tend to become large for large systems
- Time Complexity and Space complexity is high
- More token transition in huge systems

5.2 SAFE COMPOSABLE TESTING MODEL WITH PETRI NET (SCTMP)

In order to reduce testing time and to increase the efficiency of safety critical system, SCTMP framework is developed as in Fig 1. Parallel system is designed with Petri Nets which is composable and compatible and thus safety is ensured through reachability, boundedness and liveness property. Automated test sequence is generated and coverage analysis is done for transition and code coverage.
5.3 MODEL WITH PETRI NETS

Petrinets is a diagrammatic tool to model concurrency and synchronization in distributed systems. It is used to model system behaviour based on strong mathematical foundation. Petri Nets are designed for parallel system for Anti collision device with trains T1, T2, T3 and T4 representing objects. Two objects T1 and T2 when receiving GPS signal, calculate distance (D) and on the same track within a range of 3KM represented by the state S1 and goes to wait state representing the trains to stop resulting in safe state. If distance cannot be calculated based on poor weather condition and with the system state in S1 represents an unsafe state with the signals not being sent to make the trains stop as specified in Figure 5.2.
5.4 ALGORITHM FOR SCTMP

Algorithm for verification

Identify the components from requirement specification with common functionalities related to hardware and software
Int i, j, a[], b[], testcase[], n

// i, j represents the iteration through components
// n is the number of components

For (i = 1 to n)

J = 2

Input the no. of components n

Compare (A[i] with b[j]) where a[i] and b[j] are components

If (a[i] and b[j]) is dependent related to interfaces, o/p, control, function call

Compose (a[i] and b[j])

Check for compatible (a[i] and b[j]) with external environment

Form a network with (a[i] and b[j])

Repeat the above process until n

Let s1, s2, ..., sn represents the states of components

S[n+1] = safe, s[n+2] = unsafe

Int Time t = 0;

If (t) connect with condition (s1, s2)

Generate Petri Nets with Tokens, Places and Transition

Testcase[50];

for each testcase[i]; treetraversal(a[i])

If (a[i] = safe) then verify = 1; count++

Else

Verify = 0

If (count >= 40) then

Validate(s1, s2, ..., sn)
Algorithm for Validation (reducing time for testing)

Automate(s1,s2,...,sn)

Automate test sequence(s1,s2...sn)

Generate code coverage

The above algorithm specifies the identification of components from requirements and forming a network with compatible components. The network is designed for Petri Net with properties being checked for safe and unsafe state. Automation is done for speeding up the process with test sequence generation and subsequently testing for code coverage.

5.5 GENERATION OF TEST SEQUENCE

Test sequence generation represents the dynamic behaviour of objects with message passing as illustrated in Figure 5.3. There are three objects represented as Train/Station, GPS Satellite and Controller. On device ready of GPS signal is received which is passed to the microcontroller which computes the distance and apply brakes if another train is on the same track with distance less than 3KM and Status as S1 when another train arrives on same track.
5.6 TRANSITION COVERAGE

Transition coverage specifies the coverage of transitions between states. More transition coverage is achieved if more transitions are encountered when faults are seeded into the model to identify the transition paths. Once the model is developed for Petri Nets, transition coverage can be done illustrating the fault seeded with the nodes and transition traversed.

\[
\text{Transition Coverage} = \frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100
\]
5.7 CODE GENERATION: Partial Code Generation

D:Integer
N:Integer
B1:integer

//N represents the no. Of trains with position c1,c2,c3 c4,c5
Inform ,Apply ,Ready ,Wait ,Send=0
//Inform a Boolean flag signals the microcontroller
//Apply is a Boolean flag
//Ready is a Boolean flag if train starts
//Send is a Boolean flag for sending data
Function S(N)
{
If ((N==0) && (c1==0))
Return 0;
Else If (N==1) && (c1||c2||c3||c4||c5)
Ready=1
M=latitude1;
N=longitude1;
Send=1
Else
//b1 represents the signal from GPS
If (N>1) && (c1||c2||c3||c4||c5) && (b1=1)
D= Compute (latitude1,longitude1,latitude2,longitude2)
Ready=1
If (D<3km&& b1!=0) then
set wait=1
}
If(b1==0)//GPS Signal doesn’t work
Inform=1, Apply=1//Station Controller Manually takes charge
Function Compute(latitude1,longitude1,latitude2,longitude2)
    var R = 6371; // Radius of the earth in km
    var dLat = deg2rad(latitude2-latitude1); // deg2rad below
    var dLon = deg2rad(longitude2-longitude1);
    var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
            Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
            Math.sin(dLon/2) * Math.sin(dLon/2) ;
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    var d = R * c; // Distance in km
    return d;
}
function deg2rad(deg) {
    return f1 = deg * (Math.PI/180)
}

5.8 COVERAGE ANALYSIS

Coverage Analysis with Statement Coverage, Decision coverage, MC/DC accounts for 80 to 100% of code reachable
5.9 SUMMARY

Design of Petri Nets develops a simulated model which is completely automated with tools designed for Petri Net which consists of tokens, places and transition. The simulated model is analyzed for transition coverage. A sequence diagram is generated with model where code can be generated and subsequently tested for code coverage analysis. Next chapter deals with the results of automated timed automata and automated Petri Net model.