Software Reliability Modeling using Fault Tree Analysis and Stochastic Petri Nets

Avijit Khandelwal

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Software Reliability Modeling using Fault Tree Analysis and Stochastic Petri Nets

Thesis submitted in partial fulfilment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Software Engineering)

by

Avijit Khandelwal

(Roll No. 211CS3294)

under the supervision of

Prof. S. K. Rath

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

June 2013

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***Software Reliability Modeling using Fault Tree Analysis and Stochastic Petri Nets*** by ***Avijit Khandelwal*** is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology with the specialization of Software Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: June 3, 2013

(Prof. S. K. Rath)
Professor, CSE Department
NIT Rourkela, Odisha

Acknowledgment

I am grateful to numerous local and global peers who have contributed towards shaping this thesis. At the outset, I would like to express my sincere thanks to Prof. S. K. Rath for his advice during my thesis work. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge.

I extend my thanks to our HOD, Prof. A. K. Turuk for his valuable advices and encouragement.

I am really thankful to my all friends Pratik Agrawal, Pratik Agarwal, Vinay Iyer, Meghansh Sharma, Mukesh Kumar, Anita Bai, Suchitra Mishra, Swagatika Swain (Puzzle group). My sincere thanks to everyone who has provided me with kind words, a welcome ear, new ideas, useful criticism, or their invaluable time, I am truly indebted.

I would also like to thank specially Priyanka Khandelwal (Sister) and Sugandha Saha(Friend) for standing besides me all the time and support me morally and ethically.

I must acknowledge the academic resources that I have got from NIT Rourkela. I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles.

Last, but not the least, I would like to dedicate this thesis to my family, for their love, patience, and understanding.

Avijit Khandelwal
Roll: 211CS3294

Abstract

This thesis estimates the performance of software system using Stochastic Petri Nets (SPN) and Fault Tree Analysis (FTA).This paper explains how a working Petri net model and fault tree model is developed for a soft real time system. Evaluating reliability of software at the early stage of the development is become key concern for most of the organization. Assessing the reliability at early stage helps in removing many bugs and hence improving the efficiency of the architecture. Reliability is computed based on failure rates of systems by using a modified architecture based approach for reliability modelling.Fault Tree Analysis is a method for identifying and documenting the combinations of lower-level software events that allow a top-level event (or root node) to occur. But fault trees are event oriented. The dependency between the components can not be easily incorporated in the model. Similarly redundancies,time delay conditions and other dynamic behaviour can not be easily modelled using fault trees, since they are static in nature. SPN model is used to counter the problems of Fault Trees. A ATM system is taken as an example to model the system in both fault tree and stochastic petri net.

Keywords: *Failure Rate; Fault Tree Analysis; GSPN; Reliability; Stochastic Petri Nets; SRNs;*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1

Introduction

1.1 Introduction

Software system now a days being used in all spheres. Most of these developed in a heterogeneous manner rather than homogeneous. The applications are designed on the basis of components [1], this makes it difficult to define the performance characteristics of the system from input and output. With increasing trend of COTS(component off the shelf) and reuse, the system features can be accurately determined only if each of the component and the interaction between them is consider. Previous approaches of reliability estimation were mostly based on hardware reliability models. They estimated reliability from failure rates obtained from testing phase. These approaches are suitable for software developed as monolithic entity. However they cannot estimated reliability of component based system [2]. As in such system the reliability of each individual component affects the overall system reliability. Architecture base approaches are more suitable in such applications. They consider the system to be made of independent entities of modules representing system architecture. The architecture of software is combined with failure behaviour to estimate system reliability.

The system failure are mostly due to software failure rather than hardware failure, which could be due to design anomalies, improper usage by users or deployment in changing environments. Such faults could be corrected either by debugging them when they occur or through preventive measure. The former involves transient fault recovery which is corrective in nature but it results in incurring huge cost as

the recovery starts only when fault occurs. It brings much overhead on time, cost and efforts, so it needs to be reduced. So assessing reliability of software system at the early stage is becomes essential. In this thesis work a ATM system is taken as an example and reliability is predicted using stochastic petri net and fault tree analysis models.

1.2 Reliability

Software Reliability is the probability of failure-free software operations for a specified period of time in a specified environment.

- It gives an overview of the general level and pattern of risk faced by the project.
- It helps in management to focus on the high-risk component in software.
- Requirements of safety-critical systems to operate without a system failure for a given period of time.
- Reliability of a system at time t is defined as,
  R(t) = 1 - P {Probability of failure state}
- For component software reliability of each component is defined as,

$$R_i = e^{-\lambda_i \sum_{j=1}^{k} \pi_j \frac{t}{c_j} I_{i,j}} \tag{1.1}$$

  where, λ_i is failure rate of component i,
  π_j is steady state probability of state j,
  t is average execution time in state j,
  c_j is number of active component,
  $I_{i,j} = 1.0,\ 1 \leq i \leq n, 1 \leq j \leq k$, if component j is active in state i and 0 otherwise.

- For Series components reliability of system is given by,

$$R_s = \prod_{i=1}^{n} R_i\,(t) \tag{1.2}$$

- For Parallel components reliability of system is given by,

$$R_s = 1 - \prod_{i=1}^{n} (1 - R_i(t)) \tag{1.3}$$

1.3 Petri Nets

A Petri Nets is mathematical model used to describe any system. It is a directed bipartite graph defined as a 6 tuple, (P,T,A,I(.),O(.),m_0), where,

- P is a set of places.
- T is a set of transitions.
- A is a set of arcs.
- I(.), the input functions, maps transitions to places.
- O(.), the output function, that maps places to transitions.
- m_0 is the initial marking of the net, where marking is the number of tokens in each places.

Petri nets generally represented graphically. Places are represents as circle, transitions are represented as bars and arcs are represented as arrow. Tokens are represented as dots inside a place. Which describes the number of resources acquired at a particular state. The transitions are said to be enabled when the input places are having tokens. Firing of transition is the process of transfer tokens from input places to output places. Petri Nets are used to capture the behaviour of many real world situations. Sequencing, synchronization, concurrency and conflict behaviour are easily modelled by Petri Nets. The main feature of Petri Nets model is the representation of concurrent execution of activities.

1.4 Motivation

Reliability has always been the key performance measure in all software systems. Reliability modelling dates back to seventies but still the reliability estimation is in its growing states. With newer systems with more functionality and

complexities the model need to be modified accordingly for accurate reliability estimation. Most of the models based on hardware reliability but as software reliability varies newer approaches have to be considered. With the use of component based software increasing the estimation methodologies have to be made more sensitive to the effect of there software interaction on the software. As reliability is a stochastic measure this thesis estimates reliability of a modified architecture based approach and models it using fault tree analysis and stochastic petri nets.

Software systems are having wide application in critical domains where safety is more important than any other performance measure. However a software aging could be a thread to safety to minimize these preventive measures has to be taken rather than corrective.

1.5 Organisation of thesis

The thesis is organized as follows: Chapter 2 describes the literature review done for this thesis. Chapter 3 discusses the Fault tree analysis model and how the methods is applied to find the reliability of any system. This method is implemented on ATM example In Chapter 4 the stochastic petri nets model is discussed and implemented on the same ATM example to find the reliability. Finally chapter 5 concludes with the summary of work done.

Chapter 2

Chapter 2

Literature Review

Reliability have been classified in the following three ways depending on the manner in which reliability is estimated.

- Black box based reliability analysis : Reliability is estimated from failure during testing or operation. In this the internal details of the software are not considerd.
- Software metric based reliability analysis : Reliability is estimated from software analysis i.e. lines of codes, number of statements and complexity or its developement process and conditions.
- Architecture based reliability analysis : Reliability is estimated from the reliability of software components when combined with system architecture.

IN [2] software reliability prediction at the early stage of the software development life cycle using petri net has been proposed. The work is, prediction of reliability of software at architecture level. It describes how the reliability can be assess during testing phase using the number of defects found and time for detection of defects are used to derive the failure rate of each module. Also the time of removing the errors are used as repair rate of each module. This failure rate and repair rate together used to develop petri net model. Swapna S. Gokhale and Rehab El Kharboutly [3] proposed a methodology based on the stochastic reward nets(SRN) modeling paradigm for architecture based reliability analysis of concurrent software applications. Concurrency is very common in now a days in

modern software application that are developed using object oriented approach. Thus reliability analysis at the arcitecture level that takes into account the concurrency is important. AS the size of the application continuously grows, which results in state space explosion problem, this problem also address in this paper.

Zing et al [4] presents reliability modelling issues at the early stage of software development for fault tolerant software management system. An effective model of hierarchical view for fault tolerant system based on stochastic reward nets has been given. A quantitative approach for software reliability analysis is given.

In [5] a methodology to construct dependability models using generalized stochastic petri nets(GSPN) and stochastic reward nets(SRN) is described. Algorithm for converting fault tree to generalized stochastic petri nets and stochastic reward nets is provided.

In [6] a method based on stochastic petri nets(SPN) that evaluates component software reliability at the early stage of software development is proposed. The major problem of stochastic petri nets is the state space explosion problem with the increasing size of software. In this the problem is resolved by the decomposition of software architecture and a kind decomposition technique 6 is used.

In [7] reliability prediction based on stochastic reward nets has been proposed. The work extends the power of SPN by augmenting SPN with stochatic reward nets. The output is measured as reward based functions,for the evaluation of reliability for complex system. The solution of SRNs involves generation and analysis of the corresponding markov reward model.

In [8] a user oriented reliability model has been developed to measure the reliability of service of a system that is used by user community. In this paper a user oriented reliability figure of merit is defined to measure the reliability of software system with respect to user environment.

In [9] Generalized Stochastic Petri Net is used model lube oil system which takes into consideration the partial failure of their subsystems and common cause failures. The reliability of system is then analysed using Monte carlo simulation approach.

Fault tree analysis first introduced by the H. A. Watson of Bell Telephone Laboratories in connection with a US Air Force contract to study the Minuteman Missile launch control system [10] in 1961. In [11] the common cause analysis for fault tree analysis is described. A method to achieve failure of top events which has some common cause of basic failure events.

In [12] COMCAN a computer code is developed for common cause analysis. It describes the common cause analysis in terms of required inputs and various output options avaliable to the user.

Chapter 3

Chapter 3

Fault Tree Analysis

3.1 Introducton

Fault tree analysis first imagined by the H. A. Watson of Bell Telephone Laboratories in connection with a US Air Force contract to study the Minuteman Missile launch control system [10] in 1961. At the 1965 Safety Symposium, sponsored by the University of Washington and the Boeing Company, several papers were presented that describes the advantages of fault-tree analysis. The presentation of these papers marked the beginning of a Fault tree construction wide-spread interest in using fault-tree analysis as a system safety and reliability tool for complex dynamic systems such as nuclear reactors. Since 1960, great efforts have been made in solving fault trees to obtain reliability information about complex systems. Now a days fault tree analysis is commonly used to prediction the failure probability or frequency of failure of a system based on the given failure data and repair data of system component.

Fault Tree Analysis (FTA) [13] is a top-down approach where an undesirable event is identified as the "top event" in the "tree" and the potential causes that could lead to the undesirable event are identified as "branches" below. The fundamental concept of Fault Tree Analysis is the construction of system in a graphical tree like structures where some basic failure causes leads to the occurrence of the top event.

There are four basic steps for fault tree analysis. These are

- System definition

- Construction of Fault tree
- Qualitative Evaluation
- Quantitative Evaluation

3.1.1 System Definition

The first step in fault tree analysis is to defining a system.

A system is a deterministic entity comprising an interacting collection of discrete elements. First what aspect of system performance are of immediate concerns are identified. Aspects describe what kind of analysis is to be conducted. It can be Whether we are interested in the success of the system to complete a task or it can be the evaluation of finding the failure probability of the system.Fault tree analysis begins with the top event which is the undesired event of the system for example failure state of the system. Three type of information is required [2]

- The operation and failure modes of each component. A detailed description how the input affects the output of the system and how the internal operations are carried out.
- A description of how the system components are interacting with each other a state chart diagram of the components interaction behaviour.
- System boundary conditions, these defines the top event for which the tree is to be constructed. Decides whether the top event is possible or not.

3.1.2 Fault Tree Construction

Construction of the fault tree starts after the system is defined. The top event undesired event is identified and then the process proceeds either from top to bottom or bottom to up. In top to bottom approach first those immediate events are identified which causes the top event to occur and the process proceeds towards downwards until the basic events are reach which also called as leaf nodes of the tree. In bottom to up the basic events which can cause the top events to occur, are

identified first and then the process proceeds towards the top wards. Constructions continued until the top event reached. There are two types of symbols are used for fault tree construction gates and events [14].

Events

The events are used to describe the basic causes that lead to the undesired events of top event. There are number of events are used in the fault tree, these are

1. The Basic Events:

   Basic events are represented by the circle. These are the basic causes for which the probabilistic failure data or repair data is available or it has to be provided to the tree for the quantitative evaluation of the tree.

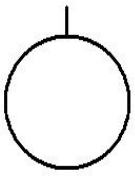

Figure 3.1: Basic Event.

2. Intermediate events:

   An intermediate events are those which occurred due to the occurrence of the one or more basic events acting through the gates.

3. The Undeveloped Events :

   The diamond shaped symbol is used to describe the undeveloped events. These are those type of specific fault events which is not further developed and for which the information is unavailable or insufficient information is available.

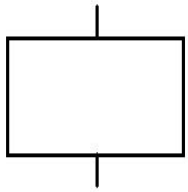

Figure 3.2: Intermediate Event.

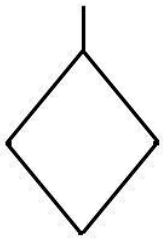

Figure 3.3: Undeveloped Event.

4. The External event:

   House type symbol is used to represents the external events. These are those types of events is to describe that the event is likely to occur. These are not the faults of themselves.

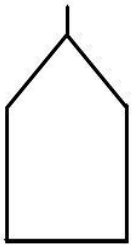

Figure 3.4: External Event.

Gates

Gates are used for linking of the events. The events are linked through the gates to reach the final top event in the tree. Some of the common gates are,

1. The OR Gate

   The output of or gates occur only if any one of the input event occur. The ouput event does not occur only if both the events does not occur. The following symbol is for or gates, For OR gate if the input to the gate is two

Figure 3.5: OR Gate.

   basic events and they are let A and B respectively. Then the probability of the gate is given by,
   P(OR Gate) = P(A or B)
   P(OR Gate) = P(A) + P(B) - P(A and B)
   P(OR Gate) = P(A) + P(B) - P(A) * P(B)

2. The And Gate

   The output event of the AND gates occurs only if both the input event occur. The Output event fails if any one of the input event to the AND gate does not occur. The following symbol is for AND Gate,

   For AND gate if the input to the gate is two basic events and they are let A and B respectively. Then the probability of the gate is given by,
   P(AND Gate) = P(A and B)
   P(AND Gate) = P(A) * P(B)

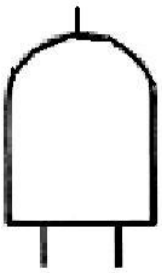

Figure 3.6: AND Gate.

3. NOT Gate

   The output event occur if the input event does not occur. The output is the complement th inputs.

Figure 3.7: NOT Gate.

3.1.3 Qualitative Evaluation

To carry out qualitative analysis we have to determine minimal cut sets [15] and minimal path sets and common cause analysis [12]. Two approaches are mainly followed to determine the minimal cut sets, one is Monte Carlo simulation and another one is deterministic methods.

A cut set is defined as a set of basic events whose occurrence results in an undesired event. Further more, if a cut set cannot be reduced but ensures the occurrence of the undesired event, the set is a minimal cut set. Obtaining minimal cut sets is a tedious process. The Monte Carlo simulation [16] procedures for finding the

minimal cut sets first assigns the random time to failure to each components based on exponential distribution. These time are generated randomly between 0 and 1 at the first time and and then finding the corresponding failure probability. These failure probability assigned to each component one at a time with the increasing time until the top event is reached. This way the cut sets are generated. After that the minimal cut sets are generated from the cut sets.

The main idea behind the deterministic approach is direct expansion of top event of the fault tree in terms of basic events using Boolean algebra.

Common cause analysis is any occurrence of event that causes multiple component to fail. Common cause events are those that multiple minimal cut sets are having common basic events. The minimal cut sets are given as input to the common cause analysis. Now the process starts by searching those minimal sets that are having common basic events.

3.1.4 Quantitative Evaluation

For quantitative analysis of the fault tree failure rate os probability of failure of each component has to be provided. The failure probability of basic events then assign to each basic events and the probability of the top event is calculated using the property of logic gates. Lets take an example [17] as shown in below figure,

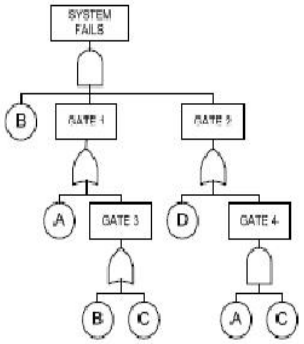

Figure 3.8: FTA Example.

From the above example two minimal cut sets can be generated as A,B,C and B,D. If the failure probability basic evets A,B,C,D are given then the probability of the top event can be computed as,

P(TOP)= B.GATE1.GATE2

=B.(A+GATE3).(D+GATE4)

=B.(A.D+A.GATE4+ GATE3.D+GATE3.GATE4)

=B.[A.D+A.A.C+(B+C).D+(B+C).A.C]

=B.[A.D+A.A.C+B.D+C.D+B.A.C+C.A.C]

Where '.' represents AND and '+' represents OR gate. The equation is simplified to ,

P(top) = A.B.C + B.D

From above equation the two minimal cut sets are A,B,C and B,D.

3.2 Implementation and Results

As we know in fault tree analysis the undesirable event is identified as top event. In our ATM example the top event is the failure of the system as shown in below figure. First the card read is basic event since if the card read is failed then the system will fail. The next event is transaction, withdrawl and balance inquiry. These events comes under the transaction fail event which is intermediate event for the top event.Here balance inquiry is basic event for transaction event. The withdrawl and transfer events are intermediate event because failure of these events caused by several basic events. The withdrawl function will fail if the balance check is fail or the network communication fail. The transfer fail event occurs if the two basic events either network or account check fails. The simulation is done using GRIF2012 tool.

Experimental Data

The failure probability of each basic event that are taken for simulation is as follows,

- Card Read = 0.25
- Balance Inquiry = 0.005
- Balance check = .035

- Network Error = 0.015
- Account Check = 0.001

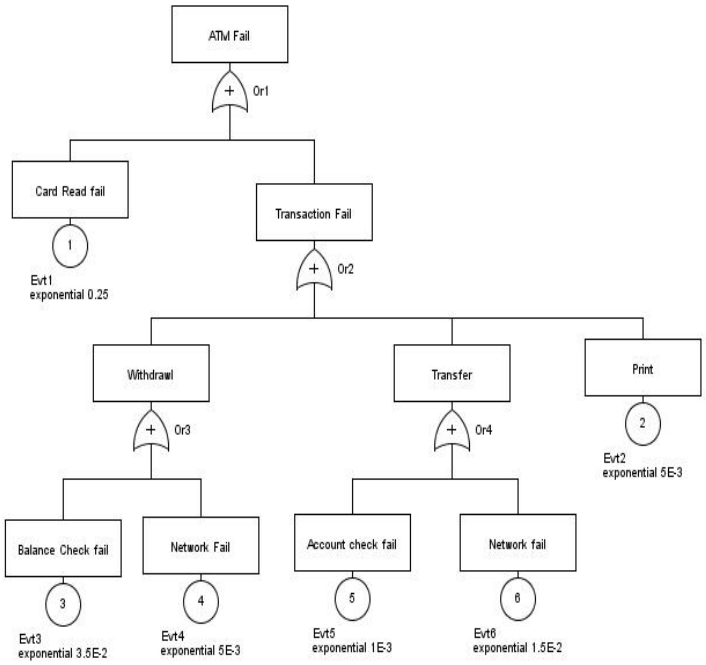

Figure 3.9: Fault Tree of ATM System

Table 3.1: Result

SN	**Time**	**Prob of Failure**	**Reliability**
1	10	0.2640	0.7360
2	20	0.4583	0.5417
3	30	0.6013	0.3987
4	40	0.7065	0.2935
5	50	0.7840	0.2160

Chapter 4

Chapter 4

Stochastic Petri Nets

The modelling power of Petri nets [18] increased by associating the firing time with either the places or transitions. When the waiting is associated with the places then the tokens arrived at the place has to wait to that time before enabling the transitions. If the waiting time is associated with the transitions then the transition becomes enable as soon as token arrives in its input places but fires only after the waiting time becomes elapsed.

Stochastic Petri Nets (SPN) [19] is helpful for performance modelling of complex software systems. They are a variant of Petri nets. They are represented as bipartite graphs, but have the time consideration which makes them suitable for performance and reliability modelling. Like Petri nets they also consist of sets of places and transitions, but unlike petri nets transitions here could be of two types immediate or timed. Any transition is enabled when there is at least one token in each of its input places and no token in any of its inhibitor places. When such an enable transition fires it deposits one token per place from a random subset of input places to per subset of its output places. The transition in SPN could be of three types:

- Immediate with no time delay
- Timed with exponentially distributed firing time.
- Timed with generally distributed firing time.

SPN models increased modelling power by associating the exponentially distributed firing time with the transitions. SPN models can have more than one transitions enabled at time. The firing of transitions is specified by the execution policy. Two alternatives are race policy and the pre-selection policy. In race policy the transitions whose firing time elapsed first is consider to be one to fired first. In pre-selection policy, the next transitions to be fired chosen from the enabled transition based on the probabilistic distribution function. A SPN model uses the race policy.

There are two variants of SPN proposed by two authors, these are

- Generalised Stochastic Petri Nets.
- Stochastic Reward Nets.

4.1 Generalised Stochastic Petri Nets

Generalized Stochastic Petri Nets (GSPN), proposed by Ajmone Marsan et al [20] is an extensions of Stochastic petri nets which allows transitions to have zero firing time or exponentially distributed firing time. When the firing associated with any transitions is zero then the transition is called immediate transitions otherwise it is called as timed transitions. When both immediate and timed transitions enabled at a time then the immediate transitions fires first. The marking of Generalized Stochastic Petri Nets are classified in to two types. If any immediate transitions enabled in the marking then the marking is called vanishing and if only timed transition is enabled or no transitions is enabled then the marking is called tangible. When two immediate transitions become enabled at a time then the conflict among the transitions is resolved using the random switch i.e. the probability mass function has to be specified in order to the selection of the first transition. The parameters of an exponential or general distribution are said to be marking dependent if they can be different in each marking.

When the SPN consist of either immediate transition or timed transition with exponential distributed firing time, it is said to be Markovian in nature where the future events depend only upon the present state and not on the past state.

It could then be used to generate underlying Markov chain [21] and solve it for computing various performance measures. However if it consist even one timed transition with generally distributed firing time, the Markov property does not hold well. To solve this problem certain instances of time are needed where the past state may not be considered. These instances are known to be regeneration point where the future depends only on the present state. Timed transitions are used to signify events which are time consuming where the enabling of transition denotes the beginning of the activity while the firing of the transition denotes its completion. Immediate transitions are used to represent instantaneous events which occur no delay. Immediate transitions thus have a sort of precedence over timed transitions. When system is modeled as SPN, its various performance characteristics can be obtained as a time average limit of markings of net. SPN is suitable for applications which involve randomness and probability as well as timed consideration. So these are efficient for modeling real time application.

Following is an example of two component parallel system [5]to describe the reliability model of a system using generalized stochastic petri net.

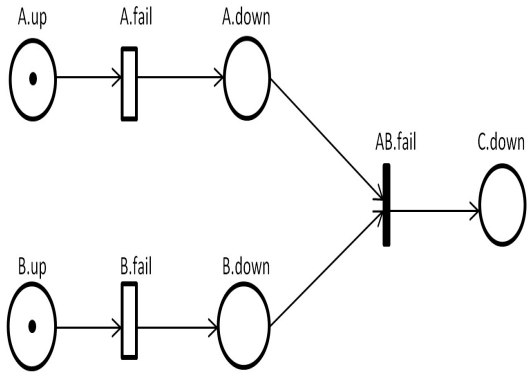

Figure 4.1: 2 Component parallel system.

Above system is 2 component parallel systems. The components are labeled as A and B. A.fail and B.fail transitions represents the failure of component A and B respectively. AB.fail transition represents the failure of the entire system. Transitions AB.fail is an immediate transitions which is fire if and only if there is token in A.down as well as B.down which removes token from place A.down and B.down and deposited tp C.down which represents the failure of entire system.

The system unreliable at time t is given by,
Pr { there is a token in place C.down at time t }.

4.2 Stochastic Reward Nets

Ciardo et al [22] [7] introduced structural extensions to GSPN named as Stochastic Reward Nets(SRN). There are various functionality he introduced in SRNs;

- Variable cardinality arc: The cardinality of an arc could be varied so as to denote the minimum number of tokens to be present in input places for a transition to be enabled.
- Transition Priorities: The priority of transition could be defined to establish priority relationship between them the one having higher priority will be preferred to that with lower priority.
- Guard Function: A transition could have an associated guard function which is evaluated only if there is possibility of the transition being enabled in that marking then transition is enabled only if the guard function is evaluated to be true. It is good for representing constraint which are difficult to be represented graphically in terms of input, inhibitor and output arcs. The stochastic process is calculated by a marking process M(t), $t > 0$ which is obtained by constructing the reachability graph of the net. First of all the reachability set has to be determined i.e. the set of possible states in the system. Then from the initial marking M_0, reachability graph can be constructed by connecting arcs when a transition enabled in one marking reaches another marking. Even if one immediate transition is enabled the marking is said to vanishing otherwise tangible.

Following is an example of two component parallel system to describe the reliability model of a system using stochastic reward nets.

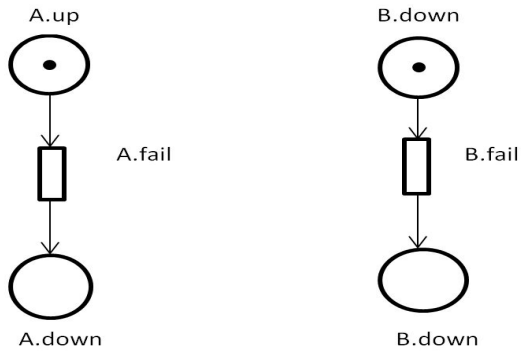

Figure 4.2: 2 Component parallel system.

Stochastic Petri Nets(SPN) are well suited to representing concurrency, synchronization, precedence and priority. The new marking probabilities determined the mechanism by which a transition removes token from a random subset of its normal input place and deposits token in a random subset of its output places when it fires. Consideration of a queuing system with batch arrivals shows that new marking probabilities must be allowed to depend explicitly on the current marking; that is, the SPN formalism must include marking dependent transition.

Additional constructs are used to selectively disable a transition in a marking which would otherwise enable it. A priority is associated with each transition. If S is the set of transitions enabled in a marking and if the transition with the highest priority among them is k , then any transition in S with priority lower than that of transition k will be disabled. To avoid theoretical difficulties, time and immediate transitions cannot have the same priority. Another way to disable transition is the inhibitor arch. An inhibitor arch from place P to transition t with multiplicity m will disable t in any marking where p contains at least nm tokens. If this two constructs are not sufficient to describe a particular mechanism, the marking dependent enabling function (also called a guard) with each transition can be used. If this function evaluates to zero in a marking, then a transition is disabled. If timing is ignored ordinary Petri net is name as a Stochastic Petri net where all transitions have the same priority, where no inhibitor arcs are present,

and where the enabling functions are identically equal to 1.

A marking which does not enable any transition is absorbing; hence it is tangible by definition. A (maximal) set of vanishing markings that are mutually reachable by immediate transition firings is called a loop (of vanishing marking). A loop is said to be absorbing if no marking in it reaches a marking outside the loop, otherwise the loop is transient. An absorbing loop is considered an error.

The reachability graph contains an arc for each different transition enabled in each marking. In particular, self-transitions (with equal input bags and output bags) are allowed by the definition of model, so arcs with coinciding source and destination may be present in the reachability graph. These arcs are ignored during the solution steps.

4.3 The Marking Process

The marking process of SPN records the marking as it emerges over continuous time. The formal definition of the marking process is in terms of an underlying general state-space Markov chain that describes the net successive marking changes. This definition leads to an algorithm for generating sample paths for the process. Many performance measures such as long run utilization, availability ,reliability, average revenue,and throughput can be specified as time limits of the marking process or underline chain or as functions of such limits. The lifetime of marking must be almost surely infinite for time average limits to be well defined. For some SPN, however, infinitely many marking changes can occur in a finite time interval, so that the lifetime is finite. Such pathological behaviour occurs if the process is absorbed into the set as of immediate marking or if the marking changes occur even more rapidly so that the sequence of occurrence times has an accumulation point. In the presence of non-exponential clock setting distributions, this latter type of explosion can occur with probability 1 even when the expected time between successive marking changes increases linearly. When the marking process of an SPN is a continuous time markov chain (CTMC), the sequence of successive time marking forms a discrete time markov chain and, given this se-

quence, the successive time between state transitions of the marking process are independent and exponentially distributed. This special structure makes it possible, in principle, to compute time average limits either analytically or numerically. One might expect that the marking process of an SPN is a CTMC if its clock setting is exponential distribution. This result is not quite true: the marking process can fail to have the markov property when the clock setting distribution function explicitly depends on current and new marking.

4.4 System modelling for reliability

Architecture based approach for reliability require following steps:

- Module identification: The various modules representing independent functional entities are identified.

- Architecture of software: This determines the mode of interaction between different modules as to whether they are sequential or concurrent.

- Failure behaviour: The failure behaviour is combined with software architecture which could be either during execution of any module or the interaction between module.

- Combining failure behaviour with architecture: The failure behaviour is combined with software architecture across the possible modes of execution by computing it only when the module corresponding to it is visited in the underlying reachability graph.

The SPNP tool is used to model the software modules. The main functional entities are represented as places while the interfaces between the modules are represented as transition. The failure rate of each transition is considered for reliability computation only if the place enabling it is marked in that marking.

This models varies from the traditional architecture based approach as it considers the failure distribution for sequential and concurrent modules to be different. The sequential modules have been represented hypo exponential distribution while the concurrent modules are represented with hyper exponential distribution. Also the utilization of each transition is considered. The equivalent reliability is

estimated by multiplying all the component reliability.

Consider a simple multiprocessor of two different processor P0 and P2. The failure occurrence time is assumed to be random variable with corresponding exponential distributions of γ_1 and γ_2. The reliability of the two processor would then be,

$R_1(t) = e^{-\gamma_1 t}$

$R_2(t) = e^{-\gamma_2 t}$

We consider the system is functioning as long as one of the two processor is functioning. If we consider the failure of two processor are to be independent, then the reliability of this system,

$R_{sys}(t) = 1 - (1 - R_1(t))(1 - R_2(t))$

The following table 4.1 and table 4.6 gives the description of different places and transitions of the system model given in fig 4.3.

Table 4.1: Description of places

SN	**Place**	**Description**
1	P0	Processor 1 up
2	P1	Processor 1 down
3	P2	Processor 2 up
4	P3	Processor 2 down

Table 4.2: Transition Description

SN	**Transition**	**Description**	**Firing Rate**
1	T0	Processor 1 fail	0.001
2	T1	Processor 2 fail	0.001
3	T2	Common Failure	0.0005
4	T3	Processor 1 Repair	0.25
5	T4	Processor 2 Repair	0.25

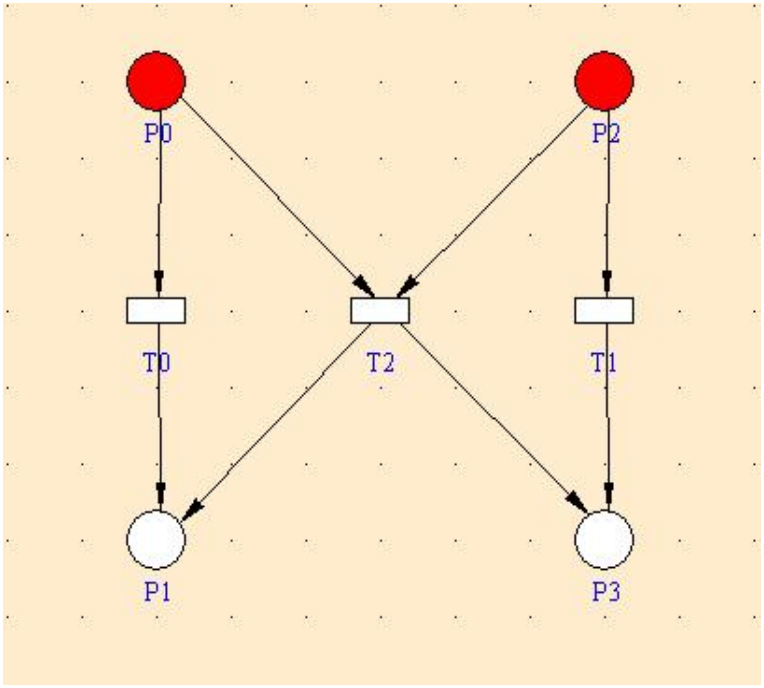

Figure 4.3: SRN model of multiprocessor system

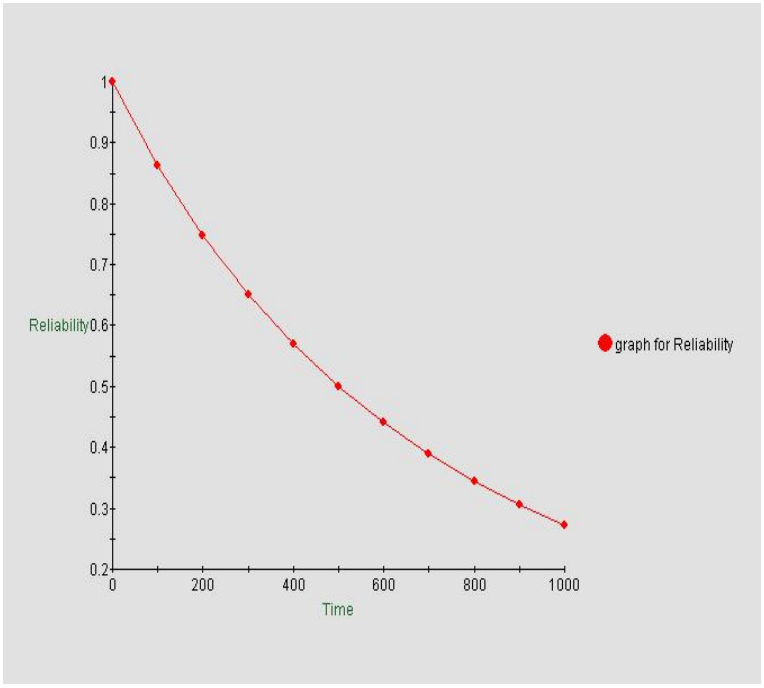

Figure 4.4: Graph for SRN model reliability

The above graph shows the reliability of the given multiprocessor system's srn model.

There also be a situation where both the processor fail simultaneously, the distribution of which is γ_3. The following figure is a complex srn model of multi-processor system with repair mechanism.

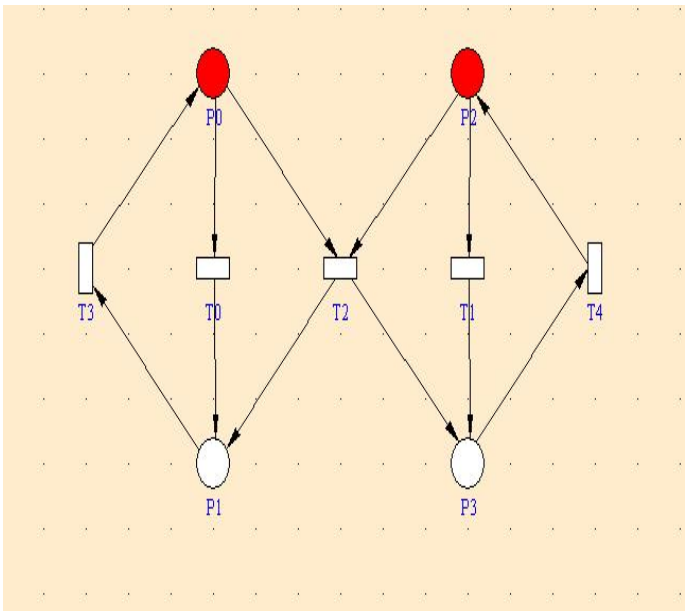

Figure 4.5: Complicated SRN model

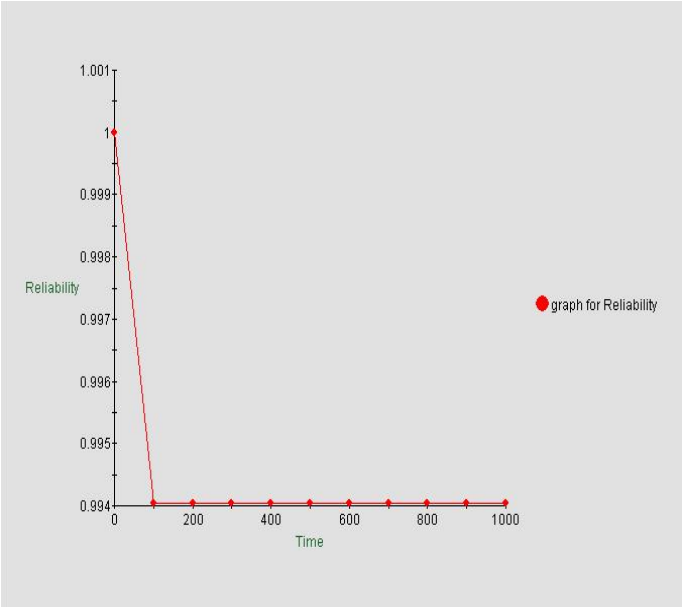

Figure 4.6: Graph for Complicated SRN model reliability

4.5 Implementation and Result

In this work, the ATM system [23]is considered for reliability modelling [24]. The various activity related to ATM functioning are modelled using stochastic petri net package [22]. The ATM system is well known to everyone since it is widely used in our day to day life. The places are represents the different states of the system.Table-4.3 gives the description of each places and Table 2 gives the description of each transition.

Table 4.3: Description of places

SN	Place	Description
1	P0	Idle
2	P1	Card read
3	P2	Pin validate
4	P3	Withdrawl
5	P4	Transfer
6	P5	Balance Inquiry
7	P6	Amount Check
8	P7	Amount collection
9	P8	Success
10	P10	Fail

IN figure 1 ATM system is modelled where initialy idle state is active. Whenever a user insert a card the idle state changes to card insert state. After the card is inserted the card is checked and state changes to pin validation state. Here the pin is validated with the central bank system and if the validation fails then token is transferred to fail state otherwise three states Withdrawl, Transfer and Balance inquiry simultaneously becomes active . Here there is possibility for each of this function to become fail. If the failure is occurred then the token is directly transferred to fail state. If withdrawl is success then the amount collection state is become active where all the balance limit is checked and after success amount collection is become active where there again is possibility that either the failure is occurred or transaction become success. After the successful transaction the token is transferred to the idle state and system become initialized to its initial state.

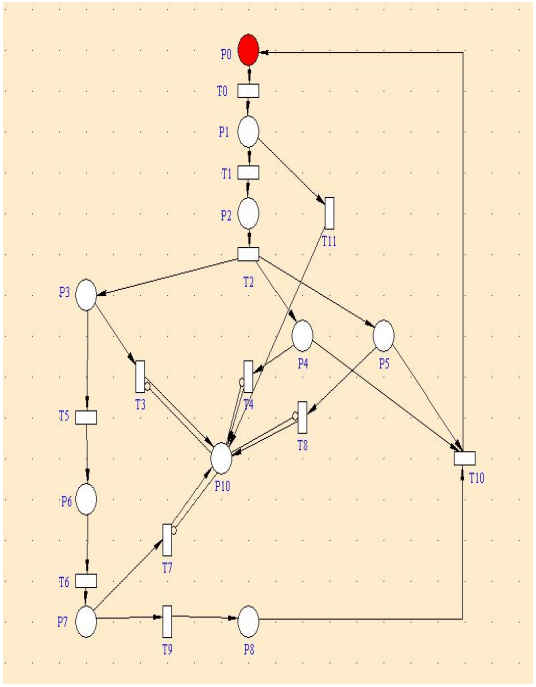

Figure 4.7: Petri Net model of ATM System

4.5.1 Result

The ATM system model is simulated using SPNP [22] tool. The system is simulated for 5 times and the corresponding probability of failure and reliability is represented in the following table. Reliability (R) is the probability that a product or service will function properly for a specified period of time under design conditions without failure. The reliability of any system is computed as the probability of failure free operation of that system in the specified period of time. Reliability is a measure that is directly related to the failure rate. Reliability is time dependent, longer the time lower the reliability, and it is computed as,

R(s) = 1 - Prob. of failure state

Table 4.4: Reliability of ATM system

SN	Time	Prob. of Failure	Reliability
1	10	4.433907124456e-001	0.556609287554
2	20	5.858311821691e-001	0.414168817831
3	30	6.700484529497e-001	0.329951547050
4	40	7.418539594250e-001	0.258146040575
5	50	8.015216261272e-001	0.198478373873

The following table shows the different firing rates of each transitions. The firing rates are the assumed data which says how frequently the firing takes place of the transitions. The transition rates are exponentially distributed with the following function,

$$F(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, x \geq 0 \\ 0, x < 0 \end{cases} \tag{4.1}$$

Table 4.5: Transition rates

SN	**Transition**	**Firing Rate**
1	T0	0.025
2	T1	0.025
3	T2	0.025
4	T3	0.005
5	T4	0.015
6	T5	0.025
7	T6	0.025
8	T7	0.035
9	T8	0.0005
10	T9	0.025
11	T10	0.025
12	T11	0.015

- The following table gives the firing rate of each transition after applying exponential distribution function with random value as 1.

Table 4.6: Transition rates

SN	**Transition**	**Firing Rate**
1	T0	0.02438
2	T1	0.02438
3	T2	0.02438
4	T3	0.00497
5	T4	0.01477
6	T5	0.00497
7	T6	0.04301
8	T7	0.03379
9	T8	0.000499
10	T9	0.2438
11	T10	0.02438
12	T11	0.01477

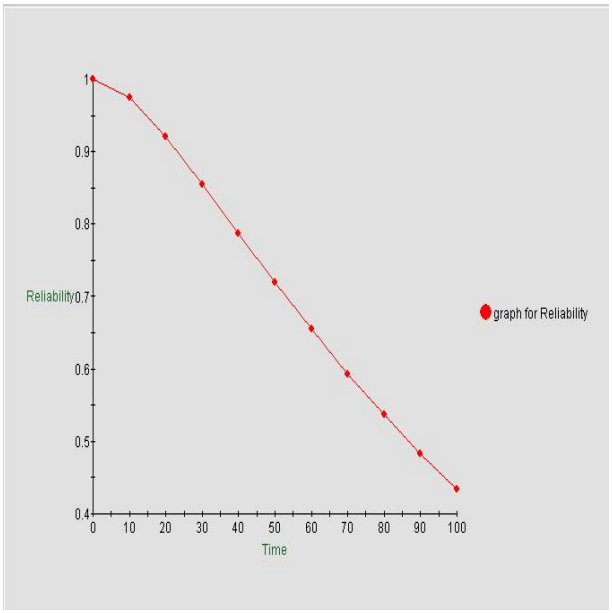

Figure 4.8: Reliability Graph for ATM System

The reliability graph above shows how the reliability of the system decreases with the time. Initially the reliability is 1 but as the simulation runs for the given time, the reliability decreases over the given period of time.

Chapter 5

Chapter 5

Conclusion

The work has been done using two different methods to model the software system. Reliability is predicted using fault tree analysis method is described and the problem associated with the method is also explained. Simultaneously the stochastic petri net method is elavorated to show how the SPN overcomes the problem of fault tree. Also it is discussed how the stochastic petri net can be used to model the dynamic behaviour of the system and reliability is predicted using the failure rate of different components. The description of how the expressive power of SRNs could be used to represent constraints on the execution of the application also presented. Reliability of concurrent software application also addressed.

Bibliography

[1] W. W. Everett, "Software component reliability analysis," in *Application-Specific Systems and Software Engineering and Technology, 1999. ASSET'99. Proceedings. 1999 IEEE Symposium on*, pp. 204–211, IEEE, 1999.

[2] K. K. Mohan, A. Verma, A. Srividya, G. V. Rao, and R. K. Gedela, "Early quantitative software reliability prediction using petri-nets," in *Industrial and Information Systems, 2008. ICIIS 2008. IEEE Region 10 and the Third international Conference on*, pp. 1–6, IEEE, 2008.

[3] R. El Kharboutly and S. S. Gokhale, "Architecture-based reliability analysis of concurrent software applications using stochastic reward nets," in *The 23rd International Conference on Software Engineering and Knowledge Engineering SEKE*, 2011.

[4] J. Zhao, H.-w. Liu, G. Cui, and X.-z. Yang, "Early stage software reliability estimation with stochastic reward nets," *JOURNAL-DONG HUA UNIVERSITY-ENGLISH EDITION-*, vol. 22, no. 3, p. 33, 2005.

[5] M. Malhotra and K. S. Trivedi, "Dependability modeling using petri-nets," *Reliability, IEEE Transactions on*, vol. 44, no. 3, pp. 428–440, 1995.

[6] L. Zhu and Y. Li, "Reliability analysis of component software based on stochastic petri nets," in *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, pp. 296–301, IEEE, 2007.

[7] J. Muppala, G. Ciardo, and K. S. Trivedi, "Stochastic reward nets for reliability prediction," *Communications in reliability, maintainability and serviceability*, vol. 1, no. 2, pp. 9–20, 1994.

[8] R. C. Cheung, "A user-oriented software reliability model," *Software Engineering, IEEE Transactions on*, no. 2, pp. 118–125, 1980.

[9] G. Thangamani, "Availability analysis of a lube oil system using generalized stochastic petri net," in *Quality and Reliability (ICQR), 2011 IEEE International Conference on*, pp. 186–190, IEEE, 2011.

[10] H. Watson, "Bell telephone laboratories. launch control safety study," *Bell Telephone Laboratories, Murray Hill, NJ USA*, 1961.

[11] R. Allan, I. Rondiris, and D. Fryer, "An efficient computational technique for evaluating the cut/tie sets and common-cause failures of complex systems," *Reliability, IEEE Transactions on*, vol. 30, no. 2, pp. 101–109, 1981.

[12] G. Burdick, "Comcan-a computer code for common-cause analysis," *Reliability, IEEE Transactions on*, vol. 26, no. 2, pp. 100–102, 1977.

[13] W.-S. Lee, D. Grosh, F. A. Tillman, and C. H. Lie, "Fault tree analysis, methods, and applications a review," *Reliability, IEEE Transactions on*, vol. 34, no. 3, pp. 194–203, 1985.

[14] D. Haasl, N. Roberts, W. Vesely, and F. Goldberg, "Fault tree handbook," tech. rep., Nuclear Regulatory Commission, Washington, DC (USA). Office of Nuclear Regulatory Research, 1981.

[15] R. Bennetts, "On the analysis of fault trees," *Reliability, IEEE Transactions on*, vol. 24, no. 3, pp. 175–185, 1975.

[16] D. Vose, *Quantitative risk analysis: a guide to Monte Carlo simulation modelling*. Wiley Chichester, 1996.

[17] J. Andrews, "Introduction to fault tree analysis," in *2012 Annual RELIABILITY and MAINTAINABILITY Symposium*, 2012.

[18] J. Billington, S. Christensen, K. Van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber, "The petri net markup lan-

guage: concepts, technology, and tools," in *Applications and Theory of Petri Nets 2003*, pp. 483–505, Springer, 2003.

[19] A. Puliafito, M. Telek, and K. S. Trivedi, "The evolution of stochastic petri nets," 1997.

[20] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, "Modelling with generalized stochastic petri nets," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 2, p. 2, 1998.

[21] R. N. Hiscott, "Markov chain analysis," *Mathematical Geology*, vol. 14, no. 5, pp. 543–544, 1982.

[22] G. Ciardo, J. Muppala, and K. Trivedi, "Spnp: stochastic petri net package," in *Petri Nets and Performance Models, 1989. PNPM89., Proceedings of the Third International Workshop on*, pp. 142–151, IEEE, 1989.

[23] I. Jacobson, G. Booch, and J. Rumbaugh, "The unified software development process–the complete guide to the unified process from the original designers," *Rational Software Corporation, US*, 1999.

[24] G. Ciardo, J. K. Muppala, and K. S. Trivedi, "Analyzing concurrent and fault-tolerant software using stochastic reward nets," *Journal of Parallel and Distributed Computing*, vol. 15, no. 3, pp. 255–269, 1992.