

## ABSTRACT

AKHAVAN-TABATABAEI, RAHA. A Markov Chain Framework for Approximation of Cycle Time in Semiconductor Manufacturing Toolsets. (Under the direction of Dr. Yahya Fathi.)

Accurate and efficient cycle time approximation is a critical issue in semiconductor manufacturing systems (SMS), since it facilitates the subsequent production planning and scheduling activities and helps reduce the overall cycle time. Presently computer simulation is a common approach to cycle time approximation and performance analysis in SMS. Simulation models however, have several inherent short-comings, and it can be difficult and time-consuming to use such models to explore various what-if questions. Compared with simulation models, analytical approaches based on queuing theory can be much faster in achieving reasonable results, and they typically provide more insights for performance improvement. But in the context of the SMS, performance of the queuing models has not been satisfactory due to inaccurate results.

We believe that a major cause of this poor performance is the application of operational rules in SMS. These rules are typically invented by line managers so as to interfere with various components of the system such as the arrival process, the service process, and the repair process, in an attempt to increase the speed of the production flow. Deployment of such rules, however, creates dependencies among these components of the system which are typically not captured by classical queuing models. This, in turn, could render the results obtained via these models somewhat inaccurate and unsatisfactory.

In this dissertation we propose a Markov chain framework to model the behavior of a toolset (workstation) in SMS under various operational rules, and employ this model to approximate, with a relatively high degree of accuracy, the long-run average cycle-time of jobs at the toolset.

To this end, we first develop a basic state-dependent Markov chain model for an SMS toolset. In this model we assume that all random components of the system, i.e., the inter-arrival time, the service time, the time between failures, and the repair time, are exponentially distributed. The state of this model is defined by a two dimensional vector consisting of the current number of active servers and the WIP level in the toolset, and the transition rates between the states are determined based on the operational rules adopted for the workstation. Given the transition rates we solve the balance equations of the Markov chain to calculate the steady state probabilities for this system.

We then use these steady state probabilities to calculate the expected value of WIP for the system, and employ Little's law to determine the long-run average cycle time for the workstation.

Subsequently we extend this Markov chain model to develop a framework for modeling toolsets with other underlying conditions and assumptions. The scenarios that we consider include toolsets with non-exponential distributions for the arrival and the service processes, toolsets with heterogeneous servers, and servers that are prone to multiple types of failure.

In order to evaluate the accuracy of this approach, we conduct a comprehensive computational experiment in which we compare the numeric values obtained via the resulting Markov models with those obtained via the corresponding simulation models. The results of this experiment show that our proposed approach obtains numeric values that are significantly more accurate than those obtained via classical queuing models.

The queuing models developed within the proposed framework can be used to compare the impact of different line-management policies, i.e., operational rules, and to answer various what-if questions regarding the cycle time of SMS toolsets. Hence these models can be employed for mid to long-range capacity planning of the toolsets as well as for assessing the factory cycle time under various conditions.

A Markov Chain Framework for Approximation of Cycle Time in Semiconductor  
Manufacturing Toolsets

by  
Raha Akhavan-Tabatabaei

A dissertation submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the Degree of  
Doctor of Philosophy

Industrial Engineering

Raleigh, North Carolina

2011

APPROVED BY:

---

Dr. Tom Culbreth

---

Dr. Yahya Fathi  
Chair of Advisory Committee

---

Dr. George Shanthikumar

---

Dr. Reha Uzsoy

---

Dr. James R. Wilson

## BIOGRAPHY

Raha Akhavan-Tabatabaei was born in San Francisco, California to Iranian parents. Soon after her birth the family moved back to Tehran, Iran where Raha was raised and went to college. In December 2000 she graduated from Sharif University of Technology with a Bachelor of Science degree in industrial engineering. She then joined an automotive manufacturing company where she worked as the project manager for quality management systems. In August 2003 she moved to Raleigh, NC in pursuit of higher education. In 2005 she received a dual Master of Science degree in industrial engineering and operations research from NC State and immediately joined the Ph.D. program at the Fitts Department of Industrial and Systems Engineering. During her years at NC State, Raha served as teaching assistant and lecturer for various undergraduate courses. In January 2007 she joined Intel Corporation as a senior industrial engineer in Chandler, Arizona, while remotely working on her dissertation. In the fall of 2009 she joined the faculty of the Department of Industrial Engineering at Universidad de los Andes in Bogotá, Colombia where she currently resides. Her research interests include stochastic processes, queueing theory, probabilistic dynamic programming and simulation.

## ACKNOWLEDGEMENTS

I want to express my deepest gratitude to my adviser Dr. Yahya Fathi for his utmost support, patience and understanding during the course of preparing this dissertation. The persuasion and encouragement that I have received from him have constantly energized my work and due to his attention and care I never felt being far away from campus, although I actually was.

I would also like to thank Dr. George Shanthikumar who raised my interest in stochastic modeling, introduced me to the topic of this dissertation and kindly accepted to serve on my committee. I truly appreciate the creative ideas and constructive comments that I have received from Dr. Reha Uzsoy throughout the course of my research. I sincerely thank Dr. Jim Wilson for the constant encouragement and advice and Dr. Tom Culbreth for providing me the opportunity of my first teaching experience at the university level and also serving on my committee. I also appreciate the time and efforts of Dr. Janice Wells to serve as the Graduate School representative on my committee.

Additionally, I want to thank my colleagues at Universidad de los Andes for the invaluable opportunity that they have provided me and the continuous support and interest that they have shown towards my research. Specifically, I would like to thank Dr. Roberto Zarama and Dr. Andres Medaglia for their warm support.

Last but not least, I would like to express my deepest gratitude to all the great teachers I have had in my life. First and foremost I would like to thank my beloved parents who taught me the art of living and gave me wings to fly. My father taught me strength and boldness to leave the base and explore opportunities while my mother taught me how to be grounded. My younger brother has always been an encouragement to me in every step of this journey. Throughout my years of school and college I have been blessed with numerous wonderful teachers who have shaped me and I cannot name them all here. Finally, the completion of this dissertation would have not been possible without the help and encouragement of my great friends from around the world, specially the Iranian gang of NC State!

## TABLE OF CONTENTS

List of Tables .....	v
List of Figures .....	vi
1 Introduction .....	1
1.1 Motivation and Problem Statement .....	1
1.2 Research Objectives and Methodology .....	2
1.3 Structure of Dissertation .....	3
2 Background .....	4
2.1 Overview of Semiconductor Manufacturing .....	4
2.2 Literature Review .....	10
2.3 Chapter Summary and Conclusions .....	20
3 A State Dependent Markov Chain Model .....	22
3.1 Definitions and Notation .....	22
3.2 The Basic State-Dependent Markov Chain Model .....	23
3.3 Cycle Time Approximation Using The Proposed Model .....	25
3.4 A Simulation Model for Generic Toolsets .....	26
3.5 Chapter Summary and Conclusions .....	27
4 Approximating Non-exponential Distributions .....	28
4.1 Fitting Erlang Distributions .....	28
4.2 Fitting Hyper-Erlang Distributions .....	32
4.3 Solving The Steady State Equations Using QBD .....	34
4.4 Chapter Summary and Conclusions .....	38
5 Extension To More Complex Toolsets .....	39
5.1 Toolsets with Heterogeneous Tools .....	39
5.2 Toolsets with Multiple Failure Types .....	40
5.3 Chapter Summary and Conclusions .....	42
6 Numerical Results .....	43
6.1 A Method for Performance Evaluation .....	43
6.2 Case 1 - Toolset with Erlang Underlying Distributions .....	44
6.3 Case 2 - Toolset with Hyper-Erlang Underlying Distributions .....	47
6.4 Case 3 - Toolset with Lognormal Underlying Distributions .....	48
6.5 Case 4 - Toolset with Gamma Underlying Distributions .....	50
6.6 Case 5 - Toolset with Two Heterogeneous Tools .....	52
6.7 Case 6 - Toolset with Two Failure Types .....	53
6.8 Comparison with G/G/m Approximations .....	54
6.9 Computational Requirements .....	57
7 Concluding Remarks .....	59
7.1 Summary of Completed Work .....	59
7.2 Directions for Future Research .....	61
References .....	62
Appendices .....	67

## LIST OF TABLES

Table 6.2.1	Parameters of $a(t)$ and $b(t)$ for Case 1.....	45
Table 6.2.2	Operational Rule Parameters .....	45
Table 6.2.3	$E_{avg}$ with Erlang Distributions.....	46
Table 6.3.1	Parameters of $a(t)$ for Case 2 .....	47
Table 6.3.2	Parameters of $b(t)$ for Case 2.....	47
Table 6.3.3	$E_{avg}$ with Hyper-Erlang Distributions.....	48
Table 6.4.1	Parameters of the Lognormal $a(t)$ and $b(t)$ .....	48
Table 6.4.2	$E_{avg}$ with Lognormal Distributions and Exponential Fit.....	49
Table 6.4.3	$E_{avg}$ with Lognormal Distributions and Erlang Fit .....	49
Table 6.5.1	Parameters of the Gamma $a(t)$ and $b(t)$ .....	50
Table 6.5.2	$E_{avg}$ with Gamma Distributions and Exponential Fit .....	51
Table 6.5.3	$E_{avg}$ with Gamma Distributions and Hyper-Erlang Fit .....	51
Table 6.6.1	Parameters of $a(t)$ , $b_1(t)$ and $b_2(t)$ for Case 5.....	52
Table 6.6.2	$E_{avg}$ for Heterogeneous Toolset Similar to Case 1 .....	53
Table 6.7.1	$E_{avg}$ for Toolset with Two Failure Types.....	54
Table 6.8.1	Comparison with $G/G/m$ Approximations - Case 1 .....	55
Table 6.8.2	Comparison with $G/G/m$ Approximations - Case 2 .....	55
Table 6.8.3	Comparison with $G/G/m$ Approximations - Case 3 .....	55
Table 6.8.4	Comparison with $G/G/m$ Approximations - Case 4 .....	56
Table 6.8.5	Comparison with $G/G/m$ Approximations - Case 5 .....	57
Table 6.8.6	Comparison with $G/G/m$ Approximations - Case 6 .....	57
Table 6.9.1	Computational Requirements of Case 1 .....	58
Table 6.9.2	Computational Requirements of Case 2 .....	58

## LIST OF FIGURES

Background	
Figure 1.	Major Steps of Processing A Single Layer of Oxide ..... 8
A State-Dependent Markov Chain Model	
Figure 2.	Partial Rate Diagram of The Proposed Model ..... 24
Figure 3.	Rate Diagram of Example 1 ..... 25
Approximating Non-exponential Distributions	
Figure 4.	Example of A Two Phase Erlang Distribution ..... 29
Figure 5.	Different Shapes of Erlang Distribution ..... 30
Figure 6.	Lognormal Distribution Fitted by Erlang and Exponential ..... 31
Figure 7.	Rate Diagram of Example 2 ..... 32
Figure 8.	The Three-Branch Hyper-Erlang Distribution ..... 32
Figure 9.	Gamma Fitted by Exponential and Hyper-Erlang ..... 33
Figure 10.	Rate Diagram of Example 3 ..... 35
Figure 11.	Rate Diagram of Example 2 with No Breakdown ..... 36
Extension To More Complex Toolsets	
Figure 12.	Rate Diagram of Toolset with Two Heterogeneous Tools ..... 41
Figure 13.	Rate Diagram of Toolset with Two Failure Types ..... 42
Numerical Results	
Figure 14.	Calculation of Relative Error ..... 44
Figure 15.	Utilization Graph for One Instance in Table 6.2.3 ..... 46
Figure 16.	Utilization Graph for One Instance in Table 6.4.3 ..... 50
Figure 17.	Utilization Graph for One Instance in Table 6.5.3 ..... 52
Figure 18.	Utilization Graph for One Instance in Table 6.6.2 ..... 53
Figure 19.	Utilization Graph for One Instance in Table 6.8.3 ..... 56
Appendix IV	
Figure 20.	Utilization Graph 1 ..... 74
Figure 21.	Utilization Graph 2 ..... 74
Figure 22.	Utilization Graph 3 ..... 75
Figure 23.	Utilization Graph 4 ..... 75
Figure 24.	Utilization Graph 5 ..... 76
Figure 25.	Utilization Graph 6 ..... 76
Figure 26.	Utilization Graph 7 ..... 77
Figure 27.	Utilization Graph 8 ..... 77

# 1 Introduction

## 1.1 Motivation and Problem Statement

Accurate and efficient Cycle Time (CT) approximation has become an increasingly critical issue in semiconductor manufacturing (SM) industry over the past few years. This can be attributed to several factors including the constantly decreasing price of semiconductor products and rising competition among manufacturers. The faster a manufacturer can deliver its products to the market, the more expensive they can be sold and the more satisfied become the customers of timely delivery.

The prices of direct and indirect semiconductor products including central processing units, memory chips, digital cameras and cell phones decline very quickly. Leachman *et al.* [1] show by data that prices decline exponentially at fairly constant rates over the first two-thirds of product life, and then sometimes decline more slowly during the last third. The semiconductor industry faces fast and stable rates of price decline, whereby any delay on production output leads to lowered average selling prices for that output [2].

Speed of manufacturing is a very important performance metric for semiconductor fabrication facilities (*fabs*). Semiconductor fabs are among the most capital-intensive and complex manufacturing plants in use today. Similar equipment and processes are applied to produce a variety of products including microprocessors, memories, digital signal processors and application-specific logic. Generally, a small number of companies compete on selling interchangeable products or providing foundry services with similar process technologies to customers worldwide. Operational efficiency is a critical competitive advantage and relies on maximizing the outputs from the various input resources including a large fixed capital investment.

Due to these factors manufacturers are constantly under pressure to reduce cycle time and improve delivery performance. Accurate prediction of cycle time can greatly help production planning and scheduling of fabs. It is also well known that fab capacity and fab cycle time exhibit essentially an inverse relationship. Proper capacity prediction for a given cycle time is the key to a successful fab design. Such demands raise the question of how to quickly and accurately predict cycle time. However, this question is not easy to answer due to complicated tool specifications and process flows in a semiconductor manufacturing system (SMS).

Shanthikumar *et al.* [3] consider simulation to be a common approach to cycle time approximation and performance analysis of SMS. At least in theory, simulation models can be adjusted precisely to meet various experimental purposes. They can be useful in verifying the assumptions and propositions of capacity planning and scheduling models. However, a simulation model has inherent shortcomings. It requires an enormous amount of input data, including equipment details, Work-In-Progress (WIP), management policies, and product information to produce reasonably accurate results. It also requires substantial resources to maintain and update. Based on the nature of simulation modeling, multiple replications are needed to perform proper statistical analysis. Therefore, it can be difficult and extremely time-consuming to explore what-if questions. Compared with simulation, analytical approaches based on Little's law and queueing theory can be much faster in achieving reasonable results and providing more insight for performance improvement [3].

Queueing theory is a powerful tool to evaluate the performance of a manufacturing system and can play an important role in assessing the performance of a semiconductor manufacturing line. This theory gives us insight into the trade-off between cycle times and throughput rates. Possibly because of its simplicity and qualitative insight, the  $M/M/1$  queue, as well as its variations such as  $G/G/1$ ,  $G/G/m$  and others, are widely used in this field.

However, due to the sophisticated environment of semiconductor manufacturing, application of queueing models can be very complicated. Research efforts have been on the improvement of model assumptions and model input, mainly in the first moment (averages) and the second moment (variations). However, implementation of classical queueing theory in semiconductor industry has

been unsatisfactory due to their lack of accuracy [3].

Usually fabs are constructed as job-shop systems where, similar pieces of manufacturing equipment, known as *tools*, are grouped together into *toolsets* to perform similar processes. Batches of products-to-be, known as lots, move between these toolsets according to the recipe of that product to be processed through various steps or operations.

Several factors contribute to the inaccuracy of classical queueing theory when applied to SMS toolsets. Among others, multiple products and operations on one toolset, batching and set up requirements, re-entrant processes, scrap and rework, lot split and merge and high variation in machine availability due to tool breakdown have been investigated in the literature.

We believe that one of the major contributors to the inaccuracy of classical queueing models is the violation of basic assumptions in these models. Classical queueing models assume that the sequence of inter-arrival times and the sequence of service times are mutually independent and each sequence consists of independent and identically distributed random variables.

Inter-arrival time in the context of SMS is defined as the time between the arrivals of consecutive lots to a toolset. Service time is defined as the time between the start and the completion of processing a lot at the toolset. This time includes the pure processing time, the time that the lot is actually being processed on the tool, as well as any interruptions due to tool failure or other causes and it also depends on the number of functional tools in the toolset.

The assumption of independence for inter-arrival and service times is frequently violated in SMS due to the fact that line managers intervene in the random processes of arrival and service in order to make the line flow faster. For example they sometimes increase or decrease the arrival rate at a certain toolset based on the current queue size, by controlling the operations at another toolset upstream. Other examples include allocating more or less resources to repair failed tools based on how critical the toolset is to the overall output of the factory at a given time, or postponing the preventive maintenance (PM) due to high traffic at a toolset. We refer to such informal interventions as *operational rules* throughout this document.

## 1.2 Research Objectives and Methodology

The purpose of this research is to provide a queueing model to approximate the cycle time of toolsets in SMS. In the context of SMS toolset is referred to a workstation where a number of tools or machines are grouped together. All the tools of a toolset can perform the same tasks or operations on the products and work as parallel servers. A typical SMS fab consists of a collection of toolsets that are linked together through the process flow of the products.

Our goal in this research is to accurately predict the long-run average time that a lot spends at a particular toolset to be processed for one of the operations in its process flow, given the specific parameters of the toolset including the number of tools, the arrival and service processes and the operational rule applied. We expect this model to be able to quickly and accurately evaluate the impact of operational rules on the toolset cycle time and be an effective aid to evaluate and analyze different what-if scenarios with a quick turnaround.

We choose to develop queueing models to guarantee fast result generation compared to simulation models. However, due to inherent correlations between the two stochastic processes of arrival and service in SMS, the classical queueing models fail to provide accurate results. One common type of correlation is triggered by the queue size or work in process (WIP) at the toolset that affects both arrival and service processes. This happens due to the fact that line managers tend to pay more attention to operations that have accumulated large amounts of WIP. They take countermeasures such as allocating more resources to repair failed tools or decreasing the arrival rate by performing PM on upstream tools in an attempt to quickly reduce the accumulated WIP at those operations. These types of countermeasures or operational rules are implemented spontaneously and are often informal, implicit and highly dependent on the judgement of operation managers supervising the

production. Throughout this research we consider two distinct types of such rules: *Rule I* that adjusts the mean inter-arrival time of lots and *Rule II* that adjusts the mean time to repair a tool. We assume that both rules depend on the WIP level.

We develop a Markov chain model to approximate the cycle time of a toolset in both the presence and absence of operational rules. The state space of this model includes the current number of active tools as well as the WIP level in the system. The transition rates between the states of this model are determined based on the operational rule adopted for the toolset. Given the transition rates we solve the balance equations of the Markov chain to calculate the steady state probabilities. We then use the steady state probabilities to calculate the expected value of WIP for the system. Finally using Little's formula we find the long-run average of cycle time for the toolset.

To verify the performance of this Markov chain model in cycle time approximation we build a simulation model to mimic the behavior of a typical toolset in a real SMS environment. We compare the cycle time approximation by the Markov chain model with the outcome of the simulation model as a measure of accuracy for the proposed model.

Based on the proposed Markov chain model for a basic toolset with operational rules we also present a framework to model toolsets with non-exponential distributions of arrival and service processes, toolsets with heterogeneous tools and toolsets with multiple types of failure. This general framework can be used to compare the impact of different line management policies, operational rules and what-if scenarios on the cycle time of SMS toolsets. It can also be employed for mid-to-long range capacity planning of the toolsets as well as assessing the fab cycle time using queueing networks analysis.

### 1.3 Structure of Dissertation

A brief overview of semiconductor manufacturing processes is presented in Chapter 2 followed by a survey of the relevant literature. The focus of this survey is mainly on the review of existing models for single-stage systems and queueing networks, proposed for manufacturing and semiconductor manufacturing systems. The shortcomings of these classical models in cycle time approximation of SMS toolsets are also briefly discussed.

In Chapter 3 we propose our approach to modeling the basic toolsets in SMS using a Markov chain model in order to improve the accuracy of cycle time approximation while maintaining fast response. The proposed model is capable of reflecting the effects of operational rules on cycle time as well. We also discuss the development of a simulation model to examine the accuracy of predictions. Throughout the research the cycle time predictions of the proposed model are compared to the cycle time obtained by simulation for various scenarios.

The proposed Markov chain model in Chapter 3 assumes that all the underlying distributions of the arrival and service processes for the toolset are exponential. This assumption can be limiting in real SMS toolsets. To remove this potential limitation in Chapter 4 we expand the basic Markov chain model to include toolsets with non-exponential underlying distributions. We also present a method based on Quasi Birth-Death Processes in order to solve the steady-state equations of the extended Markov chain model for complex toolsets.

In Chapter 5 we extend the proposed Markov chain model of Chapter 3 to accommodate more complex situations in SMS, specifically toolsets with heterogeneous tools and toolsets with multiple failure types.

In Chapter 6 we present the results of a computational experiment with our proposed model. In this experiment we determine the cycle time for a given toolset under different rule scenarios with exponential and non-exponential underlying distributions and also for heterogeneous tools and tools with multiple failure types. We present several cases with different underlying distributions of arrival and service processes (*i.e.*, lognormal and gamma). In the scenarios with no operational rule we also compare the performance of our model with two commonly used  $G/G/m$  approximation

models. At the end of this chapter we study the computational efficiency of the proposed framework for cycle time approximation and compare its computational requirements with simulation.

In Chapter 7 we give a summary of completed work in this dissertation and briefly discuss two potential directions to extend this research. These steps include exploring methods to improve the accuracy and computational efficiency of the fitting methods for non-exponential underlying distributions as well as applying the proposed framework to approximate the cycle time of a group of toolsets joined together in job shop format, representing a semiconductor manufacturing facility.

## 2 Background

### 2.1 Overview of Semiconductor Manufacturing

An integrated circuit commonly referred to as an IC or a semiconductor chip, is a complex device that consists of miniaturized electronic components and their interconnections. The production of IC's is accomplished in a four-stage process that begins with raw plates of silicon or, less commonly, gallium arsenide that are called raw *wafers*. Wafers are grouped in lots, which travel together in a standard container and are destined for conversion to the same final product. The maximum lot size is usually between 20 and 100 wafers and differs from one production facility to another. It may even differ from one product to another within the same facility. The most commonly seen lot size is a set of 25 wafers.

The first stage of IC production is called wafer processing or wafer fabrication. It is conducted in a *clean room*, where special means are employed to maintain low density of airborne particles. The term *wafer fab* is commonly used to mean a clean room in which wafer fabrication is conducted. Here the intricate miniature circuits for a number of identical chips are created on each wafer. The individual chips-to-be are referred to as dice. The circuitry is created by a lengthy and complex process, and the number of dice per wafer may vary from just a few to many hundreds.

This number depends on different factors including the diameter of the raw wafers, device geometries and quality yield. The latest achievement in high volume device geometries is reported as 45nm, which indicates the width of an individual transistor on the device. Companies constantly improve wafer size and device geometry because of overall cost savings as a result of the larger number of dice per wafer. In this way by conducting the same number of process steps they can produce more dice. Wafer fabrication requires a long sequence of processing steps and involves many separate pieces of equipment, through which lots of wafers are routed in the traditional job shop fashion.

In the second stage of IC production commonly referred to as *wafer probe*, the individual dice on a wafer are tested for functionality by delicate electrical probes. Dice that fail to meet specifications are marked with an ink dot. Then the wafers are scored and broken into separate individual dice and the defective dice are discarded. In the third stage of production, called *assembly*, electrical leads are connected to the individual dice, which are then encapsulated in plastic or ceramic shells called packages. In the fourth stage of production, packaged chips are subjected to a final functional test and burn-in.

Perhaps the greatest single determinant of economic success for an IC manufacturer is the total process yield, defined as the fraction of individual dice that survives all stages of production and testing to emerge as salable packaged chips. Total yield may be 80% or higher for relatively simple circuits produced with mature technologies, but figures below 10% are not uncommon for large, highly integrated products in the early stages of production [3].

The wafer fabrication stage dominates the economics of IC production, and it is here that semiconductor manufacturers concentrate their research and development efforts. Wafer fabrication requires an enormous investment in plant and equipment. Because capital costs are high and variable processing costs are relatively low, high utilization of wafer fabrication equipment is a generally accepted goal in the semiconductor industry. Most competitive wafer fabs are operated on a two shift (day and night) basis for 7 days per week, but the amount of time spent actually processing wafers is limited by several factors such as preventive maintenance, setup, absence of qualified operators, end of shift effects, and frequent episodes of unscheduled downtime.

Some of this unscheduled downtime is due to the failure of equipment, but "process tuning" is often a more important downtime category. If a manufacturer reduces any of these sources of equipment unavailability, or the time required for actually processing wafers on any given piece of equipment, a higher service or throughput rate, and hence, a lower unit cost can be achieved, provided that process yields are not adversely affected.

A piece of equipment is *idle* if it is available for processing but is starved for work. Equivalently, the equipment is idle if it is neither processing wafers nor rendered unavailable for one of the reasons named above. The idleness rate for a piece of equipment is defined as the overall fraction of working hours that it spends in the idle condition. The conventional wisdom among semiconductor manufacturers is that the idleness rate for critical fabrication equipment should be no larger than 10%. Thus it will come as no surprise that wafers spend most of their time waiting in queues rather than being processed.

To set terminology, we define the fab cycle time for a lot as the total number of working hours that elapse between its entry into the clean room and its exit. This same quantity will occasionally be called the manufacturing cycle time or manufacturing interval for wafer fabrication. The toolset cycle time is defined as the total working hours that elapse between the entry of a lot into the queue of a particular toolset and its exit after being processed by that toolset.

To better understand the magnitude of the queueing effects in IC manufacturing, consider a wafer fab dedicated to production of a single, reasonably complicated product such as a laptop processor. Production of these processors involves a total of perhaps 600 distinct fabrication steps or operations. If we add the total times required to complete all these steps, assuming tools and people are always available for production, the total time might come to about one week. However, realistically it would typically take between 5 to 10 weeks to fabricate the same product. In the semiconductor industry it is common to describe this state of affairs by saying that the actual-to-theoretical ratio is between 5 and 10, or that the manufacturing cycle time is 5 to 10 times the theoretical.

This is widely recognized as a major problem for semiconductor manufacturers. In the case of customized products, the nature of the problem is obvious, since the order lead-time imposed on customers must be at least as large as the total manufacturing cycle time. On the other hand, standardized products can be made to stock, but here again, long manufacturing cycle times cause trouble because production must be based on forecasts of market demand for many months in the future, and major demand shifts are commonplace.

Moreover, product life cycles are short in the semiconductor industry, so the risk of obsolescence for finished goods inventory is always present. Finally, research shows potential negative correlation between manufacturing cycle time and yield in wafer fabrication, which provides another strong motivation for reduction of throughput times.

Thus, it is essential that the designer of a wafer fabrication line has a means to predict key performance measures including average cycle time, given only the system characteristics that are known or can reasonably be approximated before the system goes into operation.

There exist two classes of wafer fabrication facilities in the industry. Research and Development (R&D) fabs, are dedicated to development of new products and processes and are usually a smaller version of a High Volume Manufacturing (HVM) fab. HVM fabs are designed for production of salable chips but the equipment, operating procedures and process flows are essentially the same as an equivalent R&D fab but in a larger scale.

In this research, we mainly focus on the manufacturing cycle time of toolsets in HVM factories since the queueing systems formed in those fabs are far more complicated than in an R&D fab due to increased number of tools within each toolset as well as higher volume of lots flowing through the fab. Effective queueing models for HVM fabs can be easily modified for an R&D fab too. In the following sections we will explain in more details the process of wafer fabrication in an HVM fab along with a brief introduction to major fabrication equipment, processing steps and flow of the lots throughout the fab.

### 2.1.1 Wafer Processing

As stated earlier, wafer fabrication is done in a clean room, which is typically divided into U-shaped bays. A bay generally contains a major piece of equipment or a whole toolset on which one or

multiple operations are performed, plus ancillary equipment or facilities involved in closely related operations. Operators process lots on the inside of the U, and most equipment is positioned so that maintenance can be performed on the outside of the U. This area is called a chase, which is usually outside of the clean room. Companies have recently come up with newer layout plans such as a ballroom setting in which there are no bay/chase divisions and all pieces of equipment are put next to each other in a very large ballroom shaped space.

Each lot entering the clean room has an associated process flow, often called a recipe, which consists of precisely specified operations executed in a prescribed sequence on designated toolsets. If all goes well, this exact sequence of operations is performed, but sometimes inspections reveal that an operation was not executed to specification, in which case, some or all of the wafers in the lot are either scrapped or reworked.

Integrated circuit fabrication involves the creation of multiple layers on a silicon plate, hence using the term wafer to refer to the silicon plates. The operations involved in the creation of each successive layer are essentially the same, so lots can, and typically do, return repeatedly to the same toolsets for completion of the subsequent layers.

### 2.1.2 Equipment Categories and Major Process Steps

As noted before, each piece of equipment can perform one or multiple operations on the wafers. On the other hand, a fab can possess multiple pieces of the same equipment, commonly referred to as *tools*. The term *toolset* refers to a set of similar tools that are capable of performing the same operations and are grouped together in a single workstation. These toolsets usually perform one of the five major categories of operations involved in creating the successive layers on the silicon wafers. Figure 1 shows a simplified graph of these steps and changes that they make on the wafer. A brief description of each category of operations follows and more details are given in [29] and [6].

1. **Oxidation** - A thin layer of material is deposited on the surface of the wafer to form a layer of the integrated circuit. The wafers must be cleaned within a specified time before the operation is performed, to avoid particle contamination. Different oxide deposition technologies include chemical vapor deposition, spin on glass and physical vapor deposition (sputtering).
2. **Lithography** - This operation is also called photolithography, masking or patterning. The wafer is coated with a light-sensitive material called photoresist, which is then exposed to ultraviolet light through a mask (reticle) that contains a pattern reflecting the intended geometry of the circuit. The exposure step, generally referred to as photo-exposure, is the most complex and delicate operation in the wafer fabrication process, and it involves the most expensive pieces of equipment in the fabs. The importance comes from the fact that the whole geometry of device is being mapped in this step. Even a minor distortion or misalignment with previous layers can lead to malfunction of the transistors and thus result in a defective device. Many factors play role in the lithography step, including the quality of the lens that focuses the ultraviolet light on the mask as well as positioning and quality of the masks. At this stage of wafer fabrication manufacturers encounter the most common and important type of rework. If the inspection reveals that the pattern exposed in the photo-resist does not meet the specifications (because of mask misalignment, for example), then the entire photo-resist layer must be stripped off and the wafer must be prepared for a repetition of the lithography operation. If the pattern is acceptable, then exposed photoresist is removed by a developer, some minor additional operations are performed and the wafer goes forward with a protective layer of hardened, unexposed photoresist covering its surface selectively in a pattern dictated by the mask.

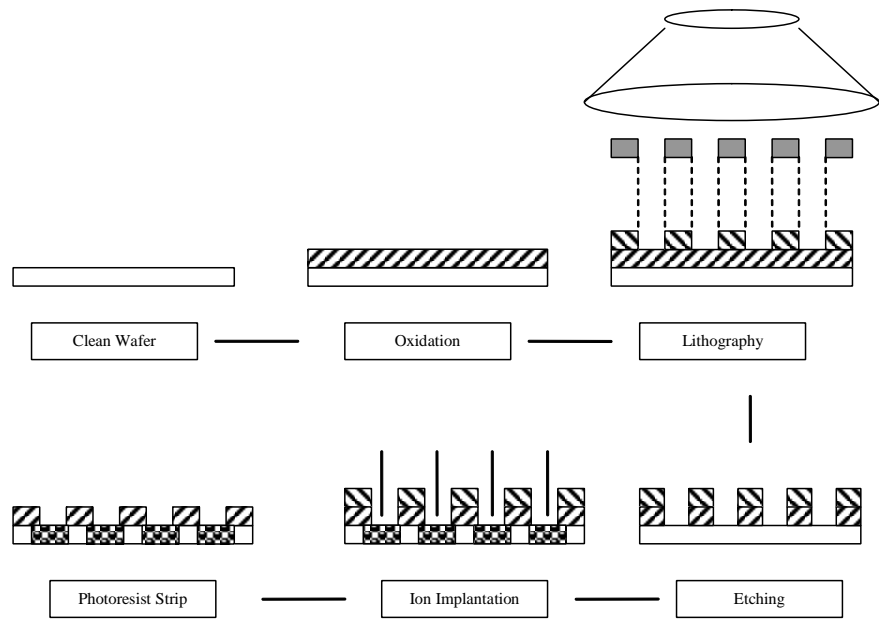


Figure 1: Major Steps of Processing A Single Layer of Oxide

3. **Etching** - Circuits are defined by etching away the portion of the oxide layer that is not protected by photoresist. There are two etching technologies used in the fabs; wet etching and plasma etching, also known as dry etching. The latter (more precise) technology is more heavily used, but wet etching is more complicated and needs more attention because of the possible inaccuracies that can happen.
4. **Ion Implantation** - In order to change the electrical properties of the surface not protected by the photoresist, the wafer is exposed to accelerated ions which are implanted to a predetermined depth and concentration. Boron, phosphorus, and arsenic ions are most frequently implanted because even a small number of their ions will dramatically alter the electrical properties of the underlying silicon.
5. **Photoresist Strip** - Finally, the pattern of photoresist that remains on the wafer after lithography is removed (stripped) using a process similar to etching. Equipment which carry out this operation are commonly referred to as planar tools.

In addition to these five types of operations, many cleaning, measurement and inspection operations are performed throughout the fab. The cleaning operations prevent contamination of the wafer, and the inspection and measurement operations are performed to identify defective wafers, which are then scrapped or reworked. Different kinds of integrated circuits require different processing steps during fabrication, and thus, have different process flows through the fab. Production facilities (as opposed to R&D labs) usually make more than one type of product. The processing technology changes quite frequently as well and therefore there is substantial diversity in product routing when the operations of any given fab are examined over a period of several years.

As stated earlier, an individual lot of wafers may visit one or more pieces of equipment repeatedly during the course of its fabrication. This phenomenon is referred to as re-entrant flow. In particular, it is often necessary to use the same photo-lithography machine (stepper or aligner) in the creation of each layer. Ion implanters and inspection stations may also be visited repeatedly. Deposition equipment and plasma etchers are often dedicated to a single operation in a single process; it is therefore common for them to be visited only once during the entire fabrication sequence.

### 2.1.3 Flow Control in HVM Fabs

Most HVM fabs are designed as classic job shop operations, in which lots with diverse characters are routed through a collection of general purpose work centers for execution of prescribed operations. The performance of a job shop is affected by the flow control mechanisms employed, including the input control, routing and sequencing rules built into the shop's operating system. Some of these routing and sequencing rules include lot-to-lens dedication for lithography operations and ion-implant run rules that are described here briefly.

Lot-to-lens dedication states that if the lithography process is done on a particular lot by a certain tool that lot has to go through the exact same tool for its subsequent lithography operations on higher layers. The reason is that lenses which are used in lithography to focus ultraviolet light on the mask, and subsequently onto the wafer through the patterns on the mask, cannot be built quite identically in a microscopic scale. The difference, although minute, can cause misalignment between successive layers. Therefore, when processing the critical layers on a wafer, each lot has to go through the exact same piece of equipment that has laid the first layer onto the wafers in that lot.

Ion-implant run rules call for certain sequencing of lots that go through an implanter machine. This sequence depends on the material that is being implanted onto the wafer. For example, there is a limit on the number of lots implanted by fluoride in a sequence or cascade. After the limit is

reached, the recipe should be changed on the tool and lots with a different implant ion should be started on that machine.

Another factor that causes the flow of lots through the fab to be stochastic is the processing of monitor lots or engineering lots. Due to high quality standards in most semiconductor companies, after any maintenance of the tools or any modification in recipe a number of monitor lots or engineering lots are run through the tool and possibly the tools that conduct subsequent operations. These lots are not considered as production lots and will be scrapped eventually. Their role is to make sure that the tools are qualified to manufacture within the required specifications after maintenance or recipe change.

Also for operations that require high accuracy such as lithography, after processing a number of lots one look-ahead (or send-ahead) lot is processed and tested before the rest of the lots in the queue are processed on that tool. This action is taken to make sure that the tool is still maintaining its qualifications to process the lots.

Hot lots are another category of production lots. These are lots with higher priority than any other lot waiting in the queue. Their priority may be caused by many different reasons, including split lots that are going to be merged with the mother lot in a few steps from the existing operation. This type of lots is especially common in R&D fabs where many experiments are performed on the process.

Processing of lots in a queue at a certain toolset is not always done on a First-In-First-Out (FIFO) basis. However, within a queue of lots from the same product type and same operation FIFO is conducted as the general policy.

#### 2.1.4 Implicit Operational Rules

The operational rules are integrated with the operation of the SMS toolsets. These rules are often implicit and unwritten, decided and implemented by the line managers and operation managers based on their observation of the factory flow and their knowledge of the situation at the upstream and downstream toolsets. These rules can be divided into two categories of *arrival-control* rules and *PM-control* rules.

The arrival-control or dispatching rules refer to the adjustment of arrival rate based on the WIP level or the toolset availability. The PM-control rules are those that change the time of performing a maintenance procedure or expedite the implementation of a repair procedure to provide more availability during high traffic situations. Usually the preventive maintenance (PM) checklists in SMS can be performed within a time window and hence the operational managers have the flexibility of pulling-in or delaying the PM. They can also pull in more resources from low traffic toolsets to carry out a repair procedure faster when the traffic at their toolset is high.

In many cases in SMS the operation managers decide to pull-in a preventive maintenance on an upstream toolset due to high traffic in the downstream. Pulling in a PM on an upstream toolset results in the reduction of arrival rate to the congested downstream toolset temporarily, and hence creates an arrival-control rule. This provides the opportunity for the downstream toolset to reduce its queue size. On the other hand, if a PM is due on one or more tools or if a tool fails within a toolset with high WIP or low availability the operation managers can expedite the repair process by adding extra resources to quickly repair the failed tools. This type of decision constitutes a PM-control rule which increases the effective service rate through increasing the availability of the toolset in high WIP or low availability.

## 2.2 Literature Review

In this section we review a number of queueing models that are either fundamental to the literature of queueing theory or customized for semiconductor manufacturing. These models aim at estimating

the cycle time of a queueing system given its characteristics. Here we consider two types of systems, namely single-stage queueing systems and queueing networks. Toolsets in SMS are usually considered as single-stage queueing systems with single or multiple tools working in parallel. On the other hand, fabs consist of a group of toolsets connected together through the process flow that form a network of single-stage queueing systems. Their cycle time can be analyzed using queueing network analysis.

### 2.2.1 Single-Stage Queueing Systems

**A. Single Server Toolsets** The simplest queueing model is the  $M/M/1$  system, where the first  $M$  represents the exponential distribution of inter-arrival time, the second  $M$  represents the exponential distribution of the service time, and the number 1 represents a single tool system. This model assumes that the arrival process and the service process are independent. The average cycle time ( $CT$ ) can be calculated exactly as,

$$CT^{M/M/1} = t_q^{M/M/1} + s = \frac{\rho}{1-\rho} s + s, \quad (1)$$

$$\rho = \lambda s \quad (2)$$

where  $s$  is the average service time,  $\lambda$  is the arrival rate,  $\rho$  denotes system utilization,  $t_q^{M/M/1}$  is the average waiting time, and  $CT^{M/M/1}$  is the average cycle time of the  $M/M/1$  system. The assumption of exponential distribution of service times in the  $M/M/1$  model is ideal, which renders it invalid in most manufacturing systems. Nonetheless, it gives the basic relationship between the arrival and service processes and provides intuitive directions on how to improve the system performance.

One way to generalize the  $M/M/1$  model is assuming a general distribution for the service process. The result will be an  $M/G/1$  model, in which letter  $G$  indicates that the service time follows a general distribution. Pollaczek and Khinchin [10] derived the following exact formula for the cycle time of the  $M/G/1$  queues, which is commonly referred to as the P-K formula,

$$CT^{M/G/1} = t_q^{M/G/1} + s = \frac{\rho(1 + C_s^2)}{2(1 - \rho)} s + s \quad (3)$$

where  $C_s^2$  is the squared coefficient of variation ( $SCV$ ) of the service times.

For more realistic queueing models, such as those with a general arrival process, analytical solutions become very difficult to derive. Therefore, researchers usually use approximate models to assess the cycle time. The accuracy of these approximation models depends greatly on the actual distribution of the inter-arrival times and the service times. Accordingly, various approximations may exist for the same queueing model. For example, Buzacott and Shanthikumar [7] list three approximations for the  $G/G/1$  queues, where both the inter-arrival and service times follow general distributions. The most widely used approximation, developed by Kingman is given as,

$$CT^{G/G/1} = t_q^{G/G/1} + s \approx \frac{\rho(C_a^2 + C_s^2)}{2(1 - \rho)} s + s \quad (4)$$

where  $C_a^2$  is the  $SCV$  of inter-arrival times.

The models shown above are not sufficient to accurately describe fab performance due to their over-simplified modeling assumptions. However, these models provide general guidance on the relationship between WIP level, capacity and cycle time. For example, Wu [4] applied Little's law and the  $G/G/1$  queueing formula to approximate the cycle time of a simple factory with single server toolsets. He found some basic properties that provide managerial insight on how the variability affects production performance. He also pointed out that the variability of bottleneck toolsets should be reduced to achieve fast cycle time.

**B. Multi-Server Toolsets with General Arrival and Service Distributions** Theoretically, given the probability distributions of the arrival and service processes the performance measures of any  $G/G/m$  system can be found with an exact analytical formula. However, derivation of such formula is not practical due to the high level of complexity. There also exist many relatively accurate but complicated approximations such as those proposed by Buzacott and Shanthikumar [7] and Connors *et al.* [8].

Based on Kingman's  $G/G/1$  approximation and the  $G/G/m$  approximation developed by Whitt [9], Hopp and Spearman [10] developed the following approximation,

$$CT^{G/G/m} = t_q^{G/G/m} + s \approx \left( \frac{C_a^2 + C_s^2}{2} \right) \left( \frac{\rho^{\sqrt{2(m+1)}-1}}{m(1-\rho)} \right) s + s. \quad (5)$$

Note that Equation (5) reduces to Equation (4) in case of a  $G/G/1$  system, and to Equation (3) for an  $M/G/1$  system. Shanthikumar *et al.* [3] show an example of the results obtained from Equation (5) for toolsets with various  $C_a^2$  and  $C_s^2$  combinations. In general higher variance of either inter-arrival times or service times increase the waiting time. If the number of servers increases, even if the overall workload increases proportionally and the average utilization of each server remains the same, the average waiting time will decrease significantly.

Buzacott and Shanthikumar [7] propose the following approximation for the  $G/G/m$  queue in manufacturing systems. Their approximation is based on the queue time of an  $M/M/m$  system, a multi server system whose arrival and service processes both follow the exponential distributions denoted by  $M$ , such that

$$CT^{G/G/m} = t_q^{G/G/m} + s \approx \frac{C_a^2(1 - (1-\rho)C_a^2)/\rho + C_s^2}{2} t_q^{M/M/m} + s, \quad (6)$$

where

$$CT^{M/M/m} = t_q^{M/M/m} + s = \left( \frac{1}{m\mu - \lambda} \right) \left( \frac{m^m \rho^m}{m!(1-\rho)} \right) p(0) + s,$$

and  $p(0)$  denotes the steady state probability of having zero lots in the queue,  $t_q^{G/G/m}$  denotes the queue time of the  $G/G/m$  system, and  $t_q^{M/M/m}$  denotes the queue time of the  $M/M/m$  system.

It is worth mentioning that the  $G/G/m$  system is a reasonably realistic model for many toolsets in semiconductor manufacturing systems (SMS). Based on classical  $G/G/m$  approximations a number of customized models for SMS have been developed in the literature as well, with the goal of improving the accuracy of cycle time approximation in SMS.

Whitt [9] approximates the expected waiting time in queue for a  $G/G/m$  system based on the proportional relationship with the exact solution of the  $M/M/m$  queue.

$$t_q^{G/G/m}(\rho, C_a^2, C_s^2, m) \approx \Phi(\rho, C_a^2, C_s^2, m) \left( \frac{C_a^2 + C_s^2}{2} \right) t_q^{M/M/m} \quad (7)$$

where  $\Phi(\rho, C_a^2, C_s^2, m)$  is given in [30].

Morrison and Martin [11] extended the  $G/G/m$  model to include production parallelism, tool idleness with work in process, travel time to the queue and defect due to failed server and propose the following approximation.

$$CT^{G/G/m} = t_q^{G/G/m} + s \approx \left( \frac{C_a^2 + C_s^2}{2} \right) \left( \frac{\rho^m}{1-\rho^m} \right) s + s \quad (8)$$

However, real problems can be more complicated due to non-identical performance of tools in the same toolset. In many instances, similar tools in a toolset are manufactured by different vendors.

Although these tools are designed to perform the exact same operations, they may show different patterns of processing time, dedication requirement, setup time and maintenance or repair durations. In this case the  $m$  tools in a toolset cannot be considered identical. Also none of these formulas consider the dependency between the arrival and service processes caused by the operational rules. In the following subsections some models that are specifically developed for such complex systems, are discussed.

**C. Tool Availability** A common event in semiconductor fabs is “tool down” due to a scheduled maintenance or an unforeseen failure. The issue of tool breakdown has been well studied in literature. Aissani and Artalejo[12] deal with a single server retrial queueing system subject to independent tool failures. Based on the assumption that tool failure and repair processes are stochastically independent, Whitt [9] deals with the case when breakdown happens only when tools are in service. Mitrani and Avi-Itzhak [13] study the many-server system with independent tool breakdowns. The exact solution can be obtained numerically, but as the number of servers increases so does the computational complexity. Mitrani and Puhalskii [14] obtain a closed-form approximation for a system under heavy traffic limit. They find that queue time under such conditions is asymptotically exponentially distributed. Hopp and Spearman [10] introduce the concept of *effective processing time* to handle the more general case of independent tool failure and repair. Let  $t_e$  and  $C_e^2$  be the average and the squared coefficient of variation (*SCV*) of the effective service times, respectively, then:

$$A = \frac{m_f}{m_f + m_r}, \quad (9)$$

$$t_e = \frac{s}{A}, \quad (10)$$

$$C_e^2 = C_s^2 + (1 + C_r^2)(1 - A)A \frac{m_r}{s} \quad (11)$$

where  $A$  is the tool availability,  $m_f$  is the mean time between failures,  $m_r$  is the mean time to repair, and  $C_r^2$  is the *SCV* of the repair times. The effective processing time can be applied in any cycle time approximation formula by simply replacing  $s$  and  $C_s^2$  by  $t_e$  and  $C_e^2$ , respectively. Jacobs *et al.* [15] and Jacobs *et al.* [16] further discuss how to incorporate SMS specifications into the variation term of effective processing time.

In Equation (11) time to failure for a tool is assumed to be exponentially distributed. However, this cannot be generally true in reality since each type of tool has multiple modes of failure with distinct characteristics and parameters. Modeling the time to failure of a tool through aggregating all the failure types into one distribution is a crude approximation. Hence this assumption may cause problems if the tool’s up and down times are critical to the queueing system. It should also be noted that when a tool is not down due to failure or preventive maintenance it does not necessarily mean that it is also available for production. Activities such as setup or engineering time may contribute to tool unavailability for production even when the tool is neither in a failed state nor under a scheduled maintenance.

**D. Batch Tools** In SMS, up to one-third of the steps require lots to be processed in batches. The batch services and the resulting batch arrival at downstream toolsets create another area of research in queueing theory. Buzacott and Shanthikumar [7] discuss approximations for the  $M^x/G/1$ ,  $G^x/G/1$ , and  $G^x/G/m$  models, where  $x$  represents the distribution of batch size for arrival process. Specific to SMS, Connors *et al.* [8] developed an approximation for the model with flexible batch size and proposed an upper limit on the batch size.

Huang *et al.* [17] provide an approximation for a model with batch arrivals, batch service, and multiple recipes based on SMS specifications. Let  $p_{m,n}$  be the probability that  $c$  servers are busy and  $n$  lots are queued,  $m$  be the number of tools,  $L$  be the average number of lots in the queue and  $\omega$  be the maximum batch size, the proposed formula is as follows,

$$L = \frac{\left(\frac{z_0}{(z_0-1)^2}\right)}{\frac{z_0}{z_0-1} + \frac{m!}{(\lambda/\mu)^m} \sum_{c=0}^{m-1} \frac{(\lambda/\mu)^c}{c!}} \quad (12)$$

where  $z_0$  is a single root, lying in the interval  $(1, m\omega\mu/\lambda)$  of the following equation,

$$\frac{\lambda}{m\mu}z - \left(1 + \frac{\lambda}{m\mu}\right) + \sum_{k=1}^{\omega} e^k z^{-k} = 0. \quad (13)$$

While this approximation slightly overestimates the average queue size, they report the approximation error to be below 12% for systems with two or three process recipes.

**E. Systems with multiple products and setups** Variety of products is another important feature in SMS. Many problems arise during analysis of multi-product queueing systems. Of these problems the most discussed is how to choose dispatching and control policies to minimize queue size. Such topics usually appear in literature concerning queueing networks, that are discussed in the next section.

Another related problem is how to account for the setup time when switching from one product to another on each tool. Using the same idea of effective processing time for breakdowns, Hopp and Spearman [10] and Gel *et al.* [18] discuss approximations for four different setup rules: First-In-First-Out (FIFO), Setup Avoidance, Setup Minimization and Type Priority.

In general, the presence of multiple products makes current queueing models more complicated and sometimes difficult to analyze. For example, Huang *et al.* [17] report acceptable results for up to three recipes. However, their model is not good for more recipes.

**F. Empirical Approximations** Many factors contribute to intractability of a closed-form formula for SMS toolsets. Industry researchers often attempt to develop approximations based on experience.

If the production line remains in steady state for a long duration, statistical analysis of historical data can be used to predict future cycle time. For example, Raddon and Grigsby [19] apply a regression model to cycle time forecasting in the fab. Chien *et al.* [20] and Backus *et al.* [21] conduct cycle time prediction and control based on production line status and data mining. Hung and Chang [22] assume a piecewise linear relationship between utilization and cycle time, with parameters calculated by an optimization method. Park *et al.* [23] develop a simulation approach to approximate the shape of throughput/queue size curve for batch tools. However, none of these approaches consider the underlying stochastic relationships in queueing models and thus the corresponding solutions are not generally applicable.

One reasonable approach is to make minor adjustments to existing simple models. Martin [24] develops the following formula derived from the  $M/M/1$  model,

$$CT^{M/M/1} = t_q^{M/M/1} + s \approx \frac{1 - \frac{\rho}{2}}{1 - \rho} s + s. \quad (14)$$

This approximation eliminates certain terms from the  $M/M/1$  model to make it applicable to more general cases. Based on experience these terms are believed to cause overestimation. Another similar approach is to find the ratio between real performance and  $G/G/m$  model output based on historical

inputs. Then for future approximations one can apply this ratio as a correction multiplier to the  $G/G/m$  model output.

In general, the application of empirical approximations is limited to “ideal” fabs with consistent product-mix, process flow, and volume. However, in the dynamic environment of semiconductor fabs, empirical approximations cannot always give satisfactory results.

### 2.2.2 Queueing Networks

Semiconductor manufacturing fabs consist of a number of toolsets that process the lots of wafers. The lots arriving at different toolsets have to wait in queues if the tools in those toolsets are busy or not available for production. Each toolset can be treated as a single-stage queueing system where its arrival process is determined by the departure processes of its upstream toolsets, which are determined by the process flow. Thus, there are numerous single-stage queueing systems in the fab at the various toolsets, and there are interactions between their queues.

In a system that contains a number of such stochastically linked toolsets the arrival patterns of lots to each toolset are stochastic. The service or throughput rate of the toolsets are also stochastic because toolsets may exhibit a natural variation in their processing time and they are also subject to unforeseen failures. These characteristics of fabs make them suitable for modeling as networks of queues where the lots wait for processing at different toolset across this network. In such a model the toolsets constitute the nodes of the network, connected by arcs representing flow of the lots between these nodes.

If in a queueing network new lots never enter and existing lots never depart the system, or departing lots are immediately replaced, this system is called a closed network. If the lots are able to enter and depart the network, such a system is considered an open network. Both models have been extensively studied in the literature. In closed queueing networks, the lot release decision is based upon some performance indicators of the manufacturing systems e.g., WIP. For instance, consider CONWIP, a WIP management policy that keeps fab-wide WIP level and cycle time relatively constant. It is expected that a queueing network with given lot releasing rules can lead to relatively fixed WIP levels and cycle time. Such a queueing network can be considered “closed” because the lot arrival process is purposely adjusted to match the departure process. According to [7], closed networks can be converted into open networks. In this section we only discuss open queueing networks.

Since in semiconductor fabs functionally similar machines are grouped together in one area, they can be considered job shop systems. In such systems the material movement is achieved through transporters and high WIP is usually accumulated at the stations. In this section we review those approaches that are suitable for modeling job shops only.

Jackson [25] first developed and analyzed a queueing network model as a dynamic job shop under the assumption of a Poisson external arrival process, exponentially distributed service times, and Markovian job transfers between tools. In such a system, the steady state probability of the system being in each state can be derived by stationary equilibrium. He also studied the performance of closed networks with constant WIP. Although Jackson’s model provides an exact solution, the assumptions in this model are very simple and not realistic for complicated manufacturing systems. Advanced methodologies based on approximation have been developed to handle more general models.

**A. Decomposition approach** Researchers have extended Jackson’s model to incorporate general service times and inter-arrival times as a renewal process with general distributions. Such a model can be analyzed by the decomposition method. In the decomposition approach, the network is broken into its constituent nodes (toolsets) with two basic assumptions: (a) the nodes can be treated as being stochastically independent; and (b) the input to each queue is the renewal process

characterized by the mean and variance of the lot inter-arrival time distributions. The input and output of each node is linked through lot routings determined by the process flow.

There are three main steps in the approximation of the decomposition approach:

1. decomposition of the network into individual nodes;
2. analysis of each node and the interaction between nodes; and
3. re-composition of the individual results to compute the network performance.

The interaction between the nodes is analyzed by considering the three basic processes at each node, *i.e.*, queue output process, output splitting, and output merging. The traffic and variations are then carried over through the network structure.

The decomposition method for queueing networks was first introduced in Kuehn [26]. Shanthikumar and Buzacott [27] first applied the decomposition approach to manufacturing systems. Whitt [9] further refined the decomposition method and developed a classical model called queueing network analyzer (QNA). Approximations for the second moment of the sojourn time were reported in Shanthikumar and Buzacott [28].

Modeling SMS with this approach is usually more complicated than many other manufacturing systems because many details such as scrap, rework, and batch tools must be considered. Thorough understanding of the fab operations is critical to queueing network modeling. Chen *et al.* [29] apply the queueing network model to evaluate fab cycle time in an ideal fab with  $M/M/1$  system toolsets. Connors *et al.* [8] incorporates various types of tools into the fab queueing network, such as batching tools. For example, lots removed due to scrap would reduce the traffic of the downstream toolsets and thus the cycle time of these toolsets would decrease. Compared with simulation, they show that the prediction accuracy of their model is relatively high, with 47 out of 72 toolsets falling in the 95% confidence interval. Hopp *et al.* [30] develop an Optimized Queueing Network (OQNET) capacity planning model to support the design of new and re-configured fabs. They consider the SMS with features such as batch processes, re-entrant flows, multiple product types, and machine setups. Various queueing models are utilized to handle different types of tool configurations. The accuracy of this model is reported to be within 30% of the simulated results.

Further refinements with respect to SMS specifications are conducted by many other researchers. Meng [31] extends the decomposition model to incorporate batch tools with operation-specific batch size instead of machine-specific batch size. Their experiments shows that significant improvements in cycle time approximation are achieved compared to the QNA model. Curry and Deuermeyer [32] develop renewal process approximations for the inter-departure processes of batching tools. Based on the decomposition method, Hu and Chang [33] design a backward queueing network analysis (BQNA) model to derive flow control parameters to meet given performance targets in SMS.

Note that all the above mentioned models are developed based on certain fab scenarios and verified by simulation. The authors do not conduct comparisons with real fab performance. Miltenburg *et al.* do this comparison in [34]. They compare the performance of four open queueing network models (including Whitt [9] and Bitran and Tirupati [35]) using the data from three fabs. In the first fab, the values of cycle time predicted by each model are about 10% longer than real production performance. They explain this difference by additional actions taken by operators to expedite production whenever WIP levels are too large. In the second fab, the values of cycle time and WIP predicted by each model are about 5% higher. The authors fail to apply these models successfully to the third fab because of the violation of the assumptions caused by dedication of tools to products.

Bitran and Tirupati [35] analyze the multi-class network as a two-class network and propose a splitting formula that outperforms the original QNA model. They specifically perform experiments based on the data of SMS. Kim [36] proposes refinements of the decomposition approximation methods by considering the heavy traffic bottleneck phenomenon in open queueing networks with

deterministic routing and highly variable arrivals. He shows that both Bitran and Tirupati's [35] and Whitt's [9] approaches in multi-class decomposition approximation could be combined with Whitt's variability functions [37] to better explain the heavy traffic bottleneck phenomenon. Proposed approximations show great improvements in all three numerical examples. For a comprehensive review of job-shop modeling by open queueing networks, see Bitran and Dasu [38].

**B. Fluid Networks** In the context of SMS a queueing network can be modeled as a fluid network where each process step is considered as fluid and the corresponding toolsets as pumps to clear out the fluid. Fluid networks cannot be applied directly to approximate the cycle time, but they can evaluate important performance metrics for a multi-product stochastic queueing network. As pointed out by Chen and Yao [39], a linear fluid network model assumes that both the inflow and outflow processes are linear functions of time. Analysis of fluid networks is based on the functional strong law of large numbers [39].

Dai [40] shows that a fluid network is stable if starting with any initial buffer level the network drains in finite time. His work also developed methods for establishing the stability of queueing networks. Connors *et al.* [41] expressed the stability condition in a different form. Many researchers have studied the stability conditions for manufacturing systems or SMS via fluid network, such as Kumar and Kumar [42]. However, the network structures of interest are for some extreme cases. As for SMS, such critical network structures will be avoided by reasonable release and scheduling policies. Thus the stability problem based on fluid network might not be a major issue in practice.

In a microscopic view, problems can be set to find the optimal control policy with a starting inventory level. There are two types of objective functions: make-span (the time to completion of the last job), and weighted flow time (the integral of weighted buffer contents over time) where the weights are usually the WIP holding or inventory costs, according to Chen and Yao [39]. More detailed work in applying fluid networks to production scheduling of manufacturing systems can be found in Gershwin [43] and Li [44]. Connors *et al.* [41] present a method to schedule fab production based on a fluid network model. They disregard the complexity inherent in the manufacturing process, and claim that it is more appropriate to view scheduling as a resource allocation problem rather than a detailed lot sequencing problem. Their scheduling algorithm is both dynamic and global. More specifically, their fluid network model takes the current state of SMS as input and uses such dynamic input information to continuously generate up-to-date schedules.

The fluid network models are usually suitable for fab scheduling decisions as well as for long-term planning decisions such as preventive maintenance scheduling and capacity planning. The fluid model requires minimal input data and regarding the arrival and service processes, only first-order quantities are needed.

**C. Diffusion Approximation** Diffusion approximations for both open and closed stochastic congested networks are described in terms of the networks' bottlenecks under heavy traffic. The approximations arise as limits of functional central limit theorems (Reiman [45], and Harrison and Williams [46]) and are valid when the traffic intensities at the toolsets are close to one. Traffic intensity or utilization of servers is defined as the ratio of the arrival rate to the service rate. Such approximations are based on the fact that the queueing system can stochastically be modeled as a Reflected Brownian Motion (RBM). Hence they require a large number of partial differential equations to be solved, in a computationally expensive procedure.

An early representative work in this type of analysis was presented by Harrison and Reiman [47]. Harrison and Nguyen [48] presented a method called QNET for the steady state analysis of open queueing networks. The QNET method approximates the exact model by an approximate Brownian model. These Brownian models are the limits of the conventional models when the traffic intensities are very high. In most cases, QNET compares with W. Whitt's QNA [9] even under

conditions of light or moderate loading. This model has been extended by follow-up research work. For example, Dai and Harrison [49] develop algorithms of numerical analysis. Harrison and Nguyen provide a survey in [50]. Harrison and Pich [51] extend the approximation to networks with multiple unreliable servers and server setups.

Recently, there has been much research interest in the use of diffusion approximation to construct scheduling policies for queueing networks. Dai *et al.* [53] apply the QNET method to approximate the performance of two priority policies, First Buffer First Served (FBFS) and Last Buffer First Served (LBFS), in a reentrant queueing network that is particularly relevant to semiconductor manufacturing lines. An analytical method is developed to approximate the long-run average workload at each toolset and the mean sojourn time of the network. They conduct experiments on a six-toolset re-entrant network. They find that the approximation errors were less than 15% for 21 out of the 24 cases. Chen, Shen and Yao [39] extend the case to incorporate multi-class queueing networks with priority scheduling and breakdowns. They used a Semi-Martingale Reflected Brownian Motion (SRBM) approximation for the performance processes, which is verified by simulation with good accuracy in most cases.

A favorable feature of diffusion approximation is that they require no information beyond the first and second moments of the model's distributions. The shortcoming is that some features may be cancelled out because of the heavy traffic assumption. Diffusion approximation also assumes that the control policy is based on priority scheduling, for which a conventional heavy traffic limit theorem holds. However, in the case of multi-class queueing systems with complicated scheduling policies, the SRBM approximation might be the only viable solution for generalized queueing networks [53].

**D. Other Queueing Network Analyses** A basic problem in operating a multi-class queueing system is to find a control rule that schedules the servers among the different job types. For a given measure of interest, performance of the system can be characterized by the measure values achieved for these job types. Gelenbe and Mitrani [54] show that using conservation laws, the performance region of a multi-class queue can be described as a polyhedron. Shanthikumar and Yao [55] showed that a wide variety of multi-class queueing systems have the polymatroidal structure in the state space of the performance vector. Mathematical programming with the objective to minimize a certain system-wide performance measure can be built to find the optimal control policy. Bertsimas *et al.* [56] further obtain lower bounds on achievable performance by optimizing over mathematical programming formulations for stochastic systems.

Some researchers employ Lyapunov quadratic functions to study the stability and performance of queueing networks based on a unique stationary distribution for the underlying Markov chain. Kumar [57] studies stability of re-entrant queueing networks. Kumar and Meyn [58] develop a mathematical programming procedure to establish the stability of queueing networks. This approach is further extended to SMS context by Kumar and Kumar [42]. Given that the system is stable and has a bounded second moment, Kumar and Kumar [59] derive mathematical programming formulations for obtaining upper and lower performance bounds for a multi-class queueing network. This method is further extended to re-entrant queueing networks in SMS by Kumar and Kumar [42].

Another line of research is to determine the throughput of irreducible closed Markovian queueing networks. For a two-station closed network, Harrison and Wein [60] provide a scheduling policy by solving a dynamic control problem involving Brownian motion and conjectured that it is asymptotically optimal. Jin, Ou and Kumar [61] solve the problem for multi-station irreducible closed Markovian networks with the functional bounds on the throughput.

### 2.2.3 Shortcomings of Classical Models in Approximating The Cycle Time of Toolsets

As discussed previously extensive research is done on various queuing models that are specially developed for SMS. However, application of those models in real semiconductor fabrication facilities (fabs) is rather limited due to unsatisfactory results. Generally, classical queuing models tend to misestimate the cycle time of toolsets in SMS. There has been some investigation in the literature to analyze the root causes of this inaccuracy [3]. The results of the investigation can be divided into the categories that are presented below.

**A. Multi-Class Environment** In the environment of multi-product queuing networks, control policy has great impact on cycle time of each product type. Most existing models are restricted to the FIFO discipline and priority scheduling, which is often violated when batching and cascading are concerned. Specific tool configurations in SMS such as tool dedication and lot cascading also affect the accuracy of cycle time approximations.

Tool dedication is the specific relationship where certain tools in a toolset are designated to only process certain products or operation steps, such as lithography lot to lens dedication. In this case, the basic assumption of identical tools in classical  $G/G/m$  models is violated.

Lot cascading is another problem in queuing modeling for SMS that has not been well investigated. For critical tools such as lithography steppers, it is preferred to run the same type of lots in a sequence to save tool setup times. There are some requirements for lot cascading, such as a minimum or maximum value for cascade size.

Other issues related to multi-product configuration include send-ahead lots, hot lots and engineering lots. As mentioned before, send-ahead lots enforce the tools to wait until metrology results prove that the tool is qualified for that product family. A hot lot has higher priority than any other production lot. An engineering lot may be inserted into the production sequence to check the tool qualification status. All these issues can add to the inaccuracy of a given cycle time approximation.

**B. Re-entrance** The feature of re-entrant process flows in SMS is a hot topic in recent research on queuing networks [3]. The focus has been on system stability and performance bounds related to control policies of multi-class queuing networks. Kumar and Kumar [42] used both fluid network and quadratic forms to study stability of a queuing network with re-entrance. They also studied the conditions with tool setup and breakdown. They noted that an important open problem is to accurately quantify the performance of such policies.

From the perspective of single-stage queuing models, the re-entrance may not have a significant influence. Previous research ignored the fact that there may be many processing steps between two consecutive operations on the same toolset, the variation of which may weaken the correlation in the arrival process.

**C. Data-Driven Analysis** Data collection and model verification for SMS queuing modeling remains to be a major challenge. Compared to the first moments, the second moments of the model input are generally more difficult to obtain. In a fast changing environment, empirical models such as that of Hung and Chang [17] could be quite efficient. This indicates that researchers of queuing modeling should pay more attention to the collection and analysis of fab data.

One solution is to use historical data to obtain the parameters of the queuing model. Jacobs *et al.* quantified the variability of the effective processing times for the  $G/G/m$  approximation [15] and further extended the approximation to batch services [16]. They developed an algorithm that approximates the mean effective processing time,  $t_e$ , and its coefficient of variation,  $C_e$ , for a multiple toolset production line. The algorithm uses fab historical data to quantify the claims of tool capacity by lots, which include time losses due to downtime, setup time, and other irregularities.

**D. Transient State** Analysis of queueing theory is based on the assumption that the status of the system is stationary. However, in real life fab operations are never in a truly stationary status. This is one of the major issues researchers face in practical applications of queueing theory. To analyze such problems, it seems fluid network is the only viable analytical method.

Queueing theory can not answer questions like, if two tools will be taken down for two weeks, in what way this down time will affect cycle time during the specified period. It can answer the following question: if there exists certain capacity, arrival rate, service rate, and tool up and down process in the following two weeks with some variation, what will be the average cycle time if such period of time will be repeated forever. Since such average performance evaluation is a key factor for fab operations, it is still worthwhile to analyze the fab with queueing theory.

**E. Dependency Between the Arrival and the Service Processes Caused by Implicit Operational Rules** A common problem of classical queueing models for SMS is that cycle times of highly utilized tools are usually misestimated [17]. Shanthikumar *et al.* [3] consider violation of the basic assumptions in classical queueing theory as the major reason for inaccuracy of classical queueing models in cycle time approximation of toolsets. They believe that the following basic assumptions do not hold for SMS.

1. The sequence of inter-arrival times,  $\{A_n, n = 1, 2, \dots\}$ , is a sequence of independent and identically distributed (*i.i.d*) random variables.
2. The sequence of service times,  $\{S_n, n = 1, 2, \dots\}$ , is a sequence of *i.i.d* random variables.
3.  $\{A_n, n = 1, 2, \dots\}$  and  $\{S_n, n = 1, 2, \dots\}$  are mutually independent.

The dependency between arrival and service processes becomes important due to the extensive scheduling and dispatching controls in highly automated fabs and applying the operational rules. In SMS it is a common practice to change the dispatching schedule dynamically to reduce WIP variation [34]. Arrivals in multi-class networks are often highly correlated [53]. Because of the variation in arrival rates of different product types, the service process is adjusted to match the arrival pattern. For example, preventative maintenance (PM) is usually performed when the toolset workload is low. A planned maintenance may be postponed if the toolset workload is high, conditioned on the flexibility of the plan. Opportunistic maintenance may be carried out when there are few lots in the queue. In this case a PM that is due in the future may be done earlier to avoid interruptions during high WIP situations.

Shanthikumar *et al.* [3] show the relationship between WIP in the queue at certain operations and the unavailability of the toolset that performs the operation. They present a graph based on the performance of a real fab that clearly shows a correlation between the arrival process and the tool availability. Note that tool availability directly controls the service process; therefore, this is a strong evidence of the correlation between arrival process and service process.

The violation of basic queueing assumptions is mainly due to the fact that in high utilization the application of operational rules becomes very popular. Operation managers resort to such unwritten rules in order to control the cycle time at their toolset. Application of operational rules causes dependency between the arrival and service processes and violates the assumption of independence between these to stochastic processes.

### 2.3 Chapter Summary and Conclusions

In the first part of this chapter the wafer fabrication process in SMS is briefly introduced, and in the second part the relevant literature on queueing models for SMS is reviewed. Specifically, the single stage queueing models and queueing network models that are commonly used for cycle time

approximation in SMS are discussed. At the end of this section some shortcomings of the queueing models in accurate approximation of cycle time for SMS toolsets are presented. The implicit and informal operational rules in SMS are identified as one of the factors that create dependency between the arrival and service processes and hence render the classical queueing models inaccurate.

The rest of this dissertation focuses on developing a Markov chain model that is capable of incorporating the operational rules and reflecting them in the cycle time approximation of toolsets. Hence, this model is expected to produce more accurate results compared with the current queueing models. In the following chapter we discuss the basic structure of the proposed Markov chain model and explain the steps to develop a simulation model that serves as the real SMS toolset in our computational study. The performance of the proposed Markov chain model in cycle time approximation is later compared with the results of this simulation model.

### 3 A State-Dependent Markov Chain Model

In this chapter we propose a queueing model for approximating the cycle time of individual toolsets in the semiconductor manufacturing systems (SMS). This model is based on a state-dependent continuous-time Markov chain model, and allows for inclusion of various operational rules. The chapter starts by defining the appropriate parameters and notation associated with various entities within the toolset, and by introducing two specific operational rules that we consider in this research. The chapter then continues with a description of the structure of the state-dependent Markov chain and an explanation of the method we use to approximate the cycle time within the model. Finally the steps to develop a simulation model of SMS toolset are described. This simulation model is used for comparison purposes and for evaluating the accuracy of the proposed model.

#### 3.1 Definitions and Notation

We employ the following assumptions and use the following notation to describe various characteristics associated with different entities in the SMS toolsets.

**Toolset Characteristics** - We assume that the toolset consists of  $k^{\max}$  identical parallel tools. The toolset can hold up to a total of  $w^{\max}$  lots that are either being processed on the tools or waiting to be processed in a single queue formed at the toolset. The queue discipline is assumed to be first come first serve.

**Arrival Process** - We assume that the inter-arrival times of lots to the toolset are continuous *iid* random variables with a given probability density function  $a(t)$ . We denote the expected inter-arrival time by  $\frac{1}{\lambda}$ , and refer to the parameter  $\lambda$  as the *average arrival rate*.

**Service Process** - We assume that the pure processing time (without interruptions due to tool failure) of each lot on the tool is a continuous random variable with a given probability density function  $b(t)$  that is identical for all tools. We denote the expected pure processing time by  $\frac{1}{\mu}$  and refer to the parameter  $\mu$  as the *average processing rate*. We also assume that the active tools are subject to preemptive random failure and the time to fail for each tool is an *iid* continuous random variable with a given probability density function  $f(t)$ , that is identical for all tools. We denote the expected time to fail by  $\frac{1}{d}$  and refer to the parameter  $d$  as the *average failure rate*. If the failure occurs during processing of a lot, that lot is assumed to go back to the queue and restart its processing time as a new lot.

We also assume that the repair times are continuous *iid* random variable with a given probability density function  $r(t)$ , identical for all tools. We denote the expected time to repair by  $\frac{1}{u}$  and refer to parameter  $u$  as the *average repair rate*.

Throughout this chapter we assume that  $a(t)$ ,  $b(t)$ ,  $f(t)$  and  $r(t)$  are exponential density functions with corresponding rates. Later in Chapter 4 we relax this assumption for  $a(t)$  and  $b(t)$ , however throughout this dissertation  $f(t)$  and  $r(t)$  are always assumed to be exponential.

**Operational Rules** - We employ the following two types of operational rules.

Rule *I* - When the total number of lots in the system is greater than a given value  $\tilde{w}$  (*i.e.*, when  $WIP > \tilde{w}$ ) the arrival process is changed so as to decrease the average arrival rate to  $\tilde{\lambda} < \lambda$ . We refer to  $\lambda$  as the *initial arrival rate* and to  $\tilde{\lambda}$  as the *congestion arrival rate*.

Rule *II* - When  $WIP > \tilde{w}$  for a given value of  $\tilde{w}$ , the repair process is changed to increase the average repair rate to  $\tilde{u} > u$ . We refer to  $u$  as the *initial repair rate* and to  $\tilde{u}$  as the *congestion repair rate*.

## 3.2 The Basic State-Dependent Markov Chain Model

### 3.2.1 Model Structure

We consider a toolset with exponential probability density functions for  $a(t)$ ,  $b(t)$ ,  $f(t)$ , and  $r(t)$ , with corresponding average rates of  $\lambda$ ,  $\mu$ ,  $d$  and  $u$ , respectively. We propose a continuous-time Markov chain model for this toolset with the stochastic process  $\{[K(t), W(t)], t \geq 0\}$ , where  $K(t)$  represents the number of active tools at time  $t$ , and  $W(t)$  denotes the number of lots in process or in queue at time  $t$ . Variables  $K(t)$  and  $W(t)$  can take on values in the set of nonnegative integers  $K(t) = 0, \dots, k^{\max}$  and  $W(t) = 0, \dots, w^{\max}$ . We define  $[k, w]$  as the state variable for this stochastic process at any arbitrary time  $t_0$ , such that  $K(t_0) = k$  and  $W(t_0) = w$ .

At any time epoch a one-step transition from the current state  $[k, w]$  can be triggered by the occurrence of one of the following events.

1. A new lot arrives to the toolset, which takes the model from state  $[k, w]$  to  $[k, w + 1]$ .
2. A lot finishes its processing and leaves the toolset. The transition is from  $[k, w]$  to  $[k, w - 1]$ .
3. One of the  $k$  active tools fails and becomes unavailable for production. In this case the model transits from state  $[k, w]$  to state  $[k - 1, w]$ .
4. One of the currently failed tools is repaired and becomes active for production. In this case the model transits from  $[k, w]$  to  $[k + 1, w]$ .

The transition rates can depend on both of the state variables,  $k$  and  $w$  and hence are state-dependent. For example in presence of either one of the two operational rules stated above some of the transition rates become dependent on the value of  $w$ . When Rule *I* is in effect the arrival rate becomes a function of  $w$  and for Rule *II* the repair rate depends on  $w$ . The overall processing and failure rates of the toolset also depend on the number of active tools,  $k$ , as with more active tools the overall toolset processing rate is faster and the overall failure rate is also higher. We use the following general notation for the state-dependent transition rates at the state  $[k, w]$ .

1. Arrival rate:  $\lambda^{k,w}$
2. Processing rate:  $\mu^{k,w}$
3. Failure rate:  $d^{k,w}$
4. Repair rate:  $u^{k,w}$

Figure 2 shows a partial view of the transition rate diagram for this Markov chain. The total number of states for such a structure is calculated as  $(w^{\max} + 1) \times (k^{\max} + 1)$ .

Given the system parameters, *i.e.*, the values of  $k^{\max}$ ,  $w^{\max}$ , and the state-dependent transition rates for a toolset of interest, the steady state probability of each state can be calculated through solving the balance equations of this Markov chain. The steady state probability for state  $[k, w]$  is denoted by  $\pi_{(k,w)}$  and we have  $\sum_{k=0}^{k^{\max}} \sum_{w=0}^{w^{\max}} \pi_{(k,w)} = 1$ .

### 3.2.2 Example 1

Consider a toolset with  $k^{\max} = 1$ ,  $w^{\max} = 4$  and exponential probability density functions for  $a(t)$ ,  $b(t)$ ,  $f(t)$ , and  $r(t)$  with their corresponding parameters. Suppose that the operational rule of this toolset is Rule *II* with  $\tilde{w} = 1$ , and  $\tilde{u} = 2u$ , *i.e.*, it dictates that when the queue size is greater than one lot the repair rate of the failed tool is doubled. The characteristics of this toolset are summarized as follow.

$a(t)$	$b(t)$	$f(t)$	$r(t)$	$k^{\max}$	$w^{\max}$	Rule Parameters
$\lambda$	$\mu$	$d$	$u$	1	4	Rule <i>II</i> , $\tilde{w} = 1, \tilde{u} = 2u$

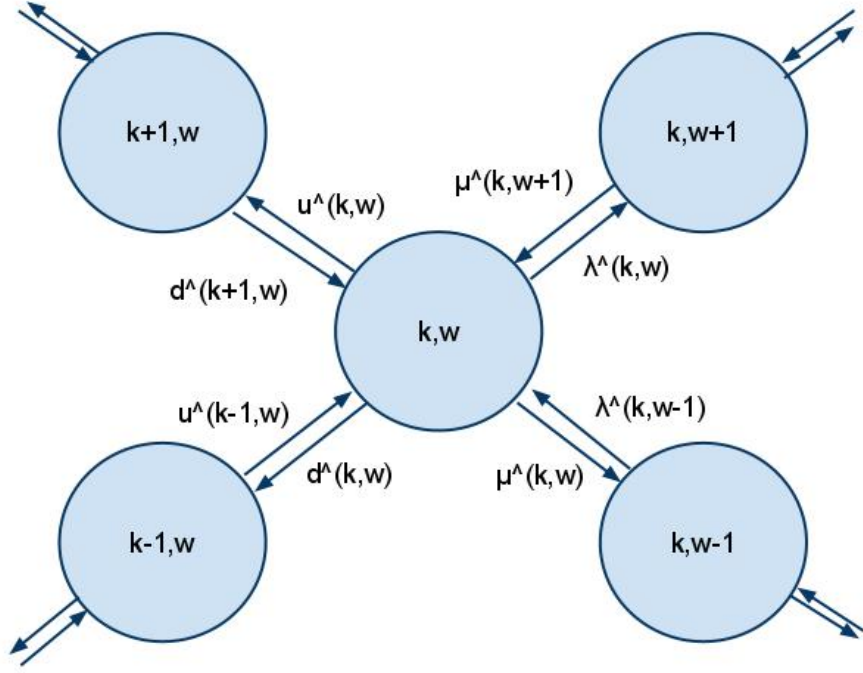


Figure 2: Partial Rate Diagram of The Proposed Model

Figure 3 depicts the transition rate diagram of such a system where each state is described by a set of two variables  $[k, w]$  that represent the number of active tools and the number of lots in system, respectively. Due to the operational rule when  $w > 1$  the repair rate of the tool doubles. This can be interpreted as allocating twice the resources to perform the repair such as assigning an extra technician to the job. This situation is modeled through changing the transition rates from subset of states  $[0, w]$  to  $[1, w]$  for states with  $w > 1$ . The total number of states in the Markov chain is calculated as  $(w^{\max} + 1) \times (k^{\max} + 1)$  or  $5 \times 2 = 10$  states.

The infinitesimal generator matrix  $Q$  of this Markov chain is shown below where  $a = \lambda + u$ ,  $b = \lambda + 2u$ ,  $c = \lambda + d$ ,  $e = \lambda + \mu + d$  and  $f = \mu + d$ .

$$\begin{bmatrix}
 -a & \lambda & 0 & 0 & 0 & u & 0 & 0 & 0 & 0 \\
 0 & -a & \lambda & 0 & 0 & 0 & u & 0 & 0 & 0 \\
 0 & 0 & -b & \lambda & 0 & 0 & 0 & 2u & 0 & 0 \\
 0 & 0 & 0 & -b & \lambda & 0 & 0 & 0 & 2u & 0 \\
 0 & 0 & 0 & 0 & -2u & 0 & 0 & 0 & 0 & 2u \\
 d & 0 & 0 & 0 & 0 & -c & \lambda & 0 & 0 & 0 \\
 0 & d & 0 & 0 & 0 & \mu & -e & \lambda & 0 & 0 \\
 0 & 0 & d & 0 & 0 & 0 & \mu & -e & \lambda & 0 \\
 0 & 0 & 0 & d & 0 & 0 & 0 & \mu & -e & \lambda \\
 0 & 0 & 0 & 0 & d & 0 & 0 & 0 & \mu & -f
 \end{bmatrix}$$

The row vector of transition probabilities is defined as vector  $\pi$  where,

$$\pi = [\pi_{00} \quad \pi_{01} \quad \pi_{02} \quad \pi_{03} \quad \pi_{04} \quad \pi_{10} \quad \pi_{11} \quad \pi_{12} \quad \pi_{13} \quad \pi_{14}].$$

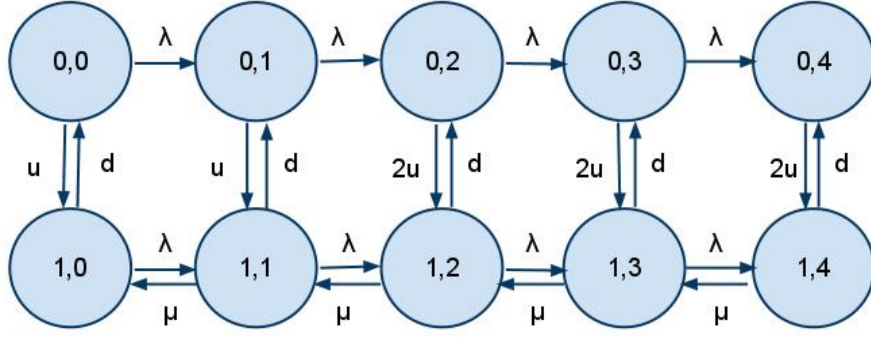


Figure 3: Rate Diagram of Example 1

Given the values for the transition rates we can solve the system of balance equations  $\pi Q = \mathbf{0}$  such that  $\sum_{k=0}^{k^{\max}} \sum_{w=0}^{w^{\max}} \pi_{(k,w)} = 1$ , to find the steady state probabilities.

### 3.3 Cycle Time Approximation Using The Proposed Model

By solving the balance equations of the system the steady state probability of being in each state  $[k, w]$  is found and denoted by  $\pi_{(k,w)}$ . Given this probability and the level of WIP at each state we can find the cycle time of the toolset as follows.

We define  $\overline{WIP}$  as the time (weighted) average number of lots at the toolset. Given  $\pi_{(k,w)}$  for  $k = 0, 1, \dots, k^{\max}$  and  $w = 0, 1, \dots, w^{\max}$  we calculate  $\overline{WIP}$  as

$$\overline{WIP} = \sum_{k,w} w \pi_{(k,w)}. \quad (15)$$

We define  $\bar{\lambda}$  as the long-run average of the state-dependent arrival rate and calculate its value as follows.

$$\bar{\lambda} = \sum_{k,w} \lambda^{k,w} \pi_{(k,w)} \quad (16)$$

The performance measure of interest for this system is the long-run average cycle time or the long-run average time spent in the system denoted by  $\overline{CT}$ . Given  $\overline{WIP}$  and  $\bar{\lambda}$  we can apply Little's formula to approximate the long-run average cycle time for the toolset.

Little's formula states that *the time average number of units in system equals the long-run average arrival rate of units to the system times the average time-in-system per unit*. Using the introduced notation this formula is defined as follows.

$$\overline{WIP} = \bar{\lambda} \times \overline{CT} \quad (17)$$

Appendix II shows that Little's formula is a distribution-free relation between the performance measures of a black box system and it holds for any stochastic input-output system with no assumptions about the number of servers or their availability. Buzacott and Shanthikumar [7] show that the only assumptions necessary to derive this formula are the uniform finiteness of the time spent by jobs in the system and the existence of the appropriate limits for  $\overline{WIP}$  and  $\overline{CT}$ . Applying

Little’s formula, the long-run average cycle time of the proposed state-dependent Markov chain is calculated as

$$\overline{CT} = \frac{\overline{WIP}}{\bar{\lambda}} = \frac{\sum_{k,w} w\pi_{(k,w)}}{\sum_{k,w} \lambda^{k,w}\pi_{(k,w)}}. \quad (18)$$

### 3.4 A Simulation Model for Generic Toolsets

To verify the performance of the proposed Markov chain model we develop a simulation model of the SMS toolsets with operational rules. In this section we discuss the framework of the simulation model to serve as the base for comparison. In the next chapter we apply the Markov chain model to approximate the cycle time of the simulated toolset under different scenarios.

#### 3.4.1 Simulation Framework

Each toolset in SMS can be modeled as a multi-server queueing system with failure prone servers. Lots arrive to this system according to a general arrival process with inter-arrival time probability density function (*pdf*) of  $a(t)$ . Upon arrival they either join the single queue at the toolset or go directly to one of the available tools to be processed. The required processing time of each lot also follows a general distribution with *pdf* of  $b(t)$ . The tools of this toolset are failure prone and alternate between states of *active* and *inactive*. A tool in the active state can either be *idle* or *busy*. The time to fail and time to repair distributions have respective *pdf*'s of  $f(t)$  and  $r(t)$  with corresponding parameters as discussed in Section 3.1.

To be consistent with the assumption of the Markov chain model we assume that the failure preempts the lot that is being processed on the tool and the preempted lot restarts its processing time when it goes to be processed for the second time. This assumption in the simulation model prevents inconsistency with the Markov model and keeps the source of deviation limited to the approximation methods for non-exponential distributions.

#### 3.4.2 Model Inputs

There are two types of input necessary for this model, entity related and resource related. At time zero in the simulation one entity is created and the time until creation of the next entity is a user-specified random variable with *pdf* of  $a(t)$  and the corresponding rate parameter. Also when the entity seizes a resource the length of time that the resource is seized is a random variable sampled from the user-defined distribution with *pdf* of  $b(t)$  and its corresponding rate parameter.

The resource related inputs are also user-specified and include the number of tools in the toolset  $k^{\max}$ , as well as the *pdf*'s of time between failures  $f(t)$ , and time to repair  $r(t)$ , for each tool and their corresponding parameters. Depending on the operational rule applied in the model the average arrival rate and the average failure rate can be determined based on the WIP level.

#### 3.4.3 Statistical Analysis of Output

Since we are interested in the long-run average cycle time of lots at the toolset we need to collect the output statistics when the model reaches steady state. In this simulation model the first entity is created when the system is in empty-and-idle state; thus we offset the initial conditions by defining a warm-up period. The appropriate length for the warm-up period can be found by monitoring one of the performance measures of the system such as WIP. The time until the level of WIP reaches steady state is used as the warm-up period.

Using this simulation model we want to estimate the mean cycle time of individual lots that we denote by  $\bar{X}_n$ . However the cycle time of consecutive lots in the queue can be positively correlated. This correlation causes problems in approximating the mean cycle time by violating the assumption of independent and identically distributed (*iid*) random variables. To overcome this issue we run a sufficient number of truncated replications of the simulation each with a long run length. Each of these replications uses a separate portion of the random number stream and hence they can be considered as mutually independent.

Assume that we run  $m$  independent replications of sufficient length where replication  $i$  produces the output cycle time of  $n$  lots  $X_{i1}, X_{i2}, \dots, X_{in}$ . Then the sample mean of this replication is defined as  $Y_i = \frac{1}{n} \sum_{j=1}^n X_{ij}, i = 1, \dots, m$ . Since the replications are mutually independent their sample means

are also *iid* random variables and hence  $\bar{Y}_m = \frac{1}{m} \sum_{i=1}^m Y_i$  is also an unbiased estimator of the mean cycle time [63]. The sample variance of  $Y_i$ ,  $S_m^2(Y) = \frac{1}{m-1} \sum_{i=1}^m (Y_i - \bar{Y}_m)^2$  is also an unbiased estimator of  $Var(\bar{X}_n)$ . If  $m$  and  $n$  are sufficiently large an approximate  $1 - \alpha$  confidence interval for the mean cycle time is  $\bar{Y} \pm t_{m-1, 1-\alpha/2} \frac{S_m(Y)}{\sqrt{m}}$ .

### 3.4.4 Modeling The Operational Rules in Simulation

In the simulation model we create the operational rules by conditioning the arrival or failure rate on the WIP level,  $w$ . Specifically, Rule *I* changes the mean inter-arrival time from  $1/\lambda$  to  $1/\tilde{\lambda}$  when the WIP is greater than the threshold,  $\tilde{w}$ . In this case whenever the toolset is in a state  $[k, w]$ ,  $w > \tilde{w}$  the mean of inter-arrival time distribution is set to  $1/\tilde{\lambda}$  and as soon as the state of the toolset is back to  $w \leq \tilde{w}$  it is changed to  $1/\lambda$  again. A similar procedure is applied for Rule *II* where the mean time to repair is changed from  $1/u$  to  $1/\tilde{u}$  when the WIP level is greater than  $\tilde{w}$ .

## 3.5 Chapter Summary and Conclusions

In this chapter the structure of a state-dependent Markov chain model is discussed. This model is capable of incorporating the operational rules through changing its state-dependent transition rates. Using the steady-state probabilities of this model the steps for approximating the cycle time of a given toolset through this model are explained and the development of a simulation model to mimic the same system as the base of comparison is also discussed.

The proposed Markov model of this chapter assumes that all the transition probabilities are exponential, an assumption that is often violated in SMS. In the following chapter we introduce methods to approximate non-exponential distributions with mixtures of exponential phases. Using such distribution approximation schemes we can model any general probability distribution as a combination of exponential phases, making it possible to work with general distributions for the inter-arrival times or the processing times in the context of the proposed Markov chain model.

## 4 Approximating Non-exponential Distributions

The proposed Markov chain model of Chapter 3 is capable of approximating the cycle time of toolsets with exponential underlying distributions in the presence of operational rules. The Markovian property of this model requires that all the underlying distributions of  $a(t)$ ,  $b(t)$ ,  $f(t)$ , and  $r(t)$  be exponential. This assumption can be limiting in the context of SMS toolsets since in reality the random variables of the arrival and service processes can follow general continuous distributions with a variety of probability density functions (*pdf*). In fact study shows that in SMS toolsets the distributions of arrival and service processes are often non-exponential [3].

In order to remove the limitation of exponential assumption for underlying distributions, in this chapter we discuss an extension to the basic Markov chain model of Chapter 3 to include toolsets with non-exponential underlying distributions of inter-arrival and processing times. In this extended model we approximate the non-exponential distributions of inter-arrival and processing time with one or more phases of exponential distribution.

There are variety of approaches proposed in the literature to fit a single exponential distribution or a mixture of exponential phases to a non-exponential continuous distribution. These approaches can be classified into two categories of Maximum Likelihood Estimates (MLE) and moment matching techniques. This subject is extensively discussed in [64], [65], [70], [71], [72], [73], and [74].

In general these methods use two types of mixture distributions based on their squared coefficient of variation (*SCV*). Hyperexponential distribution is a class of exponential mixtures with  $SCV > 1$  and hypoexponential distribution is the class of distributions with  $SCV < 1$ . The exponential distribution itself has  $SCV = 1$  and the  $SCV$  does not depend on the parameter of the distribution. In this research we introduce specific distribution fitting methods of approximating non-exponential distributions by phases of exponential distribution based on both the moment matching and MLE techniques.

Sections 4.1 and 4.2 of this chapter are dedicated to presenting three different fitting methods for non-exponential distributions, namely *Exponential Fit*, *Erlang Fit* and *Hyper-Erlang Fit*. In each section we also discuss the necessary extensions to the Markov chain model in order to include the fitted distributions. In Section 4.3 we introduce a method to solve the steady-state equations of the extended Markov chain model based on Quasi Birth-Death Processes.

### 4.1 Fitting Erlang Distributions

The approximation methods of this section apply the Erlang distribution as a class of hypoexponential distributions. An *Erlang* –  $k$  distribution with rate parameter  $\lambda$  consists of  $k$  sequential phases of exponential distribution each with rate  $\lambda$  and it is a special case of the gamma distribution with integer shape parameter,  $k$ . For example if the time until occurrence of an event is distributed as  $Erlang(k, \lambda)$ , then the time until occurrence of the next event is the sum of  $k$  exponential random variables, each with rate  $\lambda$ . The probability density function for the *Erlang* –  $k$  distribution with rate  $\lambda$  is defined as

$$f(x; k, \lambda) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}, x, \lambda \geq 0,$$

with mean  $\frac{k}{\lambda}$  and variance  $\frac{k}{\lambda^2}$ ; hence its squared coefficient of variation (*SCV*) equals  $\frac{1}{k}$  which is always less than or equal to 1. Note that the exponential distribution is a special class of Erlang distribution with  $k = 1$  and  $SCV = 1$ . Figure 4 shows the two phases of an *Erlang* – 2 distribution with rate  $\lambda$ .

Since Erlang distribution is composed of a series of phases of exponential distribution it retains the memoryless property of the exponential distribution while producing a variety of shapes depending on the value of  $k$ . Therefore when any of the underlying distributions of  $a(t)$  or  $b(t)$  is modeled as

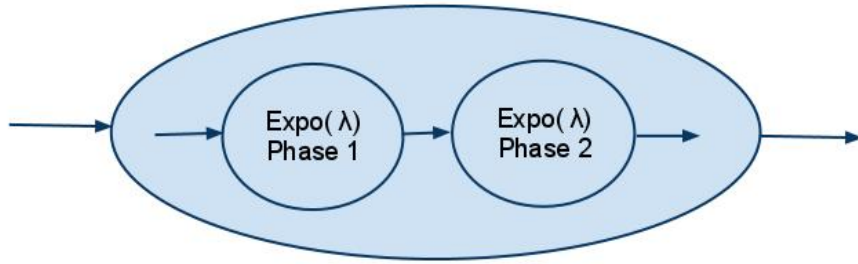


Figure 4: Example of A Two Phase Erlang Distribution

Erlang distribution they can be easily incorporated into the proposed Markov chain model. Figure 5 shows the variety of shapes that can be produced by the *Erlang* -  $k$  distribution.

In this section we propose two fitting methods based on fitting the Erlang distributions, namely *Exponential Fit* and *Erlang Fit*. The Exponential Fit approximates a non-exponential distribution with any *SCV* by matching the first moment of the non-exponential distribution with the expected value of the exponential distribution (*Erlang* - 1). For example assume that  $b(t)$  is a Normal probability density function (*pdf*) with parameters  $N(\frac{1}{\mu}, \sigma^2)$ . Using the Exponential Fit method we can approximate  $b(t)$  with an exponential distribution with expected value  $\frac{1}{\mu}$  and rate  $\mu$ . This method although quick and simple, can produce inaccurate approximations since it does not take into account any information regarding the higher moments or the shape of the *pdf* of the actual underlying distribution. Note that when applying the Exponential Fit method the Markov chain model of the toolset does not change and is the same as Figure 2.

The *Erlang Fit* method approximates a continuous distribution with  $SCV \leq 1$  with an *Erlang*- $k$  distribution. The value of  $k$  should be selected such that the *SCV* of the resulting Erlang distribution is as close to the initial distribution as possible. Throughout this dissertation for convenience we choose the value of  $k$  among three values of  $k = 1$ ,  $k = 2$ , or  $k = 10$ . Each of these values produces an Erlang distribution with a different shape, namely *Erlang*-1 with squared coefficient of variation  $SCV = 1$  (which is the same as what the Exponential Fit method approximates), *Erlang* - 2 with  $SCV = 0.5$  and *Erlang* - 10 with  $SCV = 0.1$ . Finally the rate parameter of the Erlang distribution  $\lambda$ , is chosen such that the mean of the Erlang distribution ( $\frac{k}{\lambda}$ ) be equal to the mean of the initial underlying distribution.

When either of the underlying distributions of  $a(t)$  or  $b(t)$  in a toolset is non-exponential with  $SCV \leq 1$  we measure the squared coefficient of variation (*SCV*) of the distribution and choose the shape parameter  $k$ , between the three values of 1, 2 or 10 such that the *SCV* of the resulting Erlang distribution is closest to the *SCV* of the actual underlying distribution. Then we determine the value of  $\lambda$  based on the equality of the means of the two distributions. Throughout the rest of this document we refer to this fitting method as the *Erlang Fit*.

For example suppose that one of the underlying distributions is lognormal with mean 1 and variance 0.32. The Exponential Fit method will fit a distribution with mean 1, and hence the variance of of the fitted exponential distribution will also be 1.

To apply the Erlang Fit method to the same lognormal distribution we first need to find the *SCV* for this distribution, which is calculated as  $\frac{0.32}{1} = 0.32$ . Among the three Erlang distributions that we consider here (*i.e.*, among  $k = 1, 2$ , or 10), the one that has the closest *SCV* to this lognormal distribution is *Erlang* - 2 with  $SCV = 0.5$ ; hence we set  $k = 2$ . After determining the value of  $k$  we

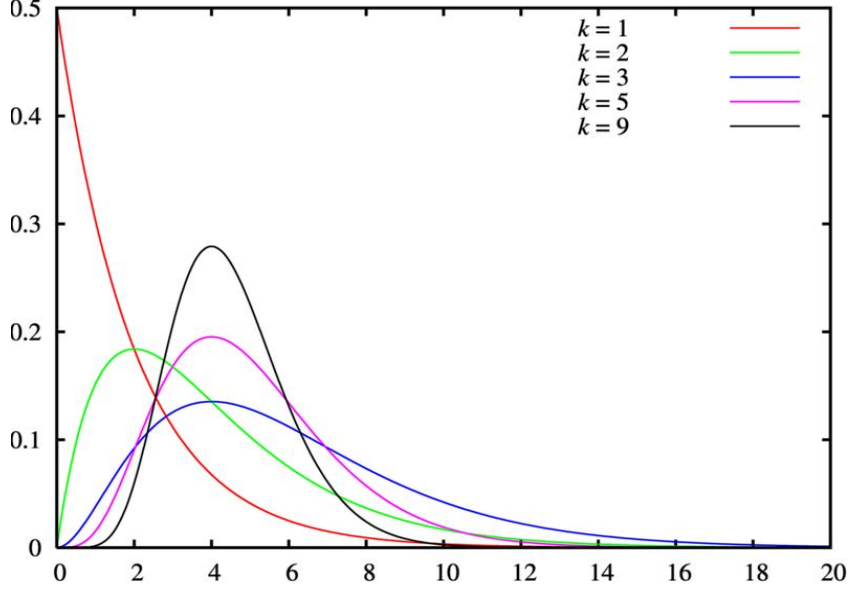


Figure 5: Different Shapes of Erlang Distribution

choose the rate parameter  $\lambda$  such that the mean of the *Erlang* – 2 distribution (which is calculated as  $\frac{2}{\lambda}$ ) be equal to the mean of the original distribution; hence the value of  $\lambda$  is chosen as 2. This results in an *Erlang* – 2 with mean of 1 and variance of 0.5.

Figure 6 also shows the *pdf* of the lognormal distribution with mean 1 and variance 0.32 fitted by *exponential* (1) and *Erlang* – 2 with rate 2. As it is observed in this example applying the Exponential Fit method is simpler and requires fewer steps compared with the Erlang Fit. Also the two fitting methods result in fitted distributions with the same mean, however the variance of the Erlang distribution is much closer to that of the original distribution compared with the variance of the resulting Exponential Fit. Also the *pdf* of the Erlang Fit is closer to that of the lognormal distribution than the Exponential Fit.

#### 4.1.1 The Markov Chain Model with Erlang Fit

To model the Erlang Fit distributions of inter-arrival or processing time for a toolset we modify the stochastic process underlying the Markov chain model of Chapter 3 by introducing new variables. These new variables indicate the phases of Erlang distributions for  $a(t)$  and  $b(t)$  at any time  $t$ . We define the modified stochastic process by  $\{[K_i(t), W(t), A(t)], t \geq 0, i = 1, \dots, k^{\max}\}$ , where variables  $K_i(t)$  and  $A(t)$  are newly introduced.

By  $A(t) \in \{0, 1, 2, \dots, A^{\max} - 1\}$  we denote the phase of the arrival process at time  $t$  and  $A^{\max}$  shows the maximum number of phases for the Erlang distribution of inter-arrival times. In the case of *Erlang* – 2 distribution  $A^{\max} = 2$ , in the case of *Erlang* – 10 distribution  $A^{\max} = 10$  and  $A(t) = i, 0 \leq i \leq A^{\max} - 1$  shows that the arrival process is in its  $i^{\text{th}}$  phase.

Similarly for the processing time distribution  $K_i(t) \in \{F, I, 0, 1, 2, \dots, P^{\max} - 1\}$ ,  $i = 1, \dots, k^{\max}$  denotes the status of the  $i^{\text{th}}$  tool, where  $K_i(t) = F$  indicates that tool  $i$  is in a *failed* state at time  $t$ ,  $K_i(t) = I$  indicates that tool  $i$  is in *idle* state and  $K_i(t) = l, 0 \leq l \leq P^{\max} - 1$  shows that the tool is in a *busy* state and the processing time for the current lot on the tool is in its  $l^{\text{th}}$  phase. Here  $P^{\max}$  denotes the maximum number of phases for the Erlang distribution of processing

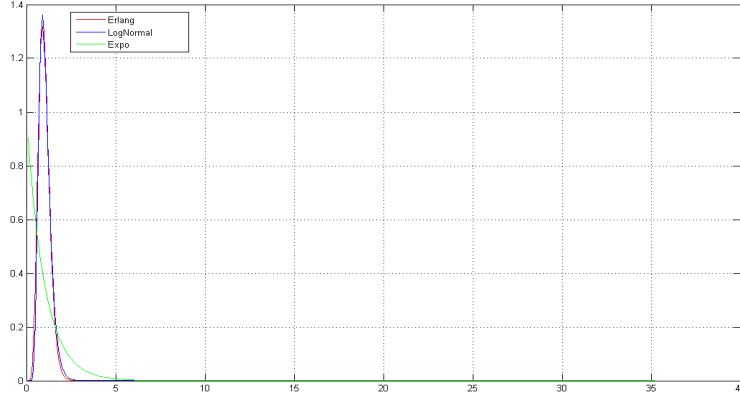


Figure 6: Lognormal Distribution Fitted by Erlang and Exponential

time.

This extension of the state-space increases the number of states of the Markov model to keep track of the phase of Erlang distributions of  $a(t)$  and  $b(t)$ . Appendix III discusses a method to calculate the total number of states for a Markov chain with Erlang underlying distributions of  $a(t)$  and  $b(t)$ , given the values of  $w^{\max}$ ,  $k^{\max}$ ,  $A^{\max}$  and  $P^{\max}$ .

#### 4.1.2 Example 2 - Example 1 Revisited with Erlang Fit

For Example 1 presented in Chapter 3, suppose that we model the inter-arrival time distribution with *Erlang* – 2 instead of exponential but the rest of the density functions remain the same. The parameters of the toolset for this example are summarized below.

$a(t)$	$b(t)$	$f(t)$	$r(t)$	$k^{\max}$	$w^{\max}$	$A^{\max}$	Rule Parameters
$\lambda$	$\mu$	$d$	$u$	1	4	2	Rule II, $\tilde{w}=1, \tilde{u}=2u$

To reflect this change in the Markov model we need to add a new dimension to each state to track the phase of the inter-arrival distribution. The state-space for the new model is defined as  $\{[K(t), W(t), A(t)], t \geq 0, A(t) = 0, 1\}$ . Figure 7 shows a partial view of the rate diagram for the Markov chain of this example with the inter-arrival time distribution modeled as *Erlang* – 2. To accommodate the two phases of arrival into the state-space of Example 1 each state in which an arrival occurs is split into two.

For example state  $[0, 1]$  in Example 1 is split into two states of  $\{[0, 0, 1], [0, 1, 0]\}$ , where state  $[0, 0, 1]$  indicates the completion of the first phase of arrival and  $[0, 1, 0]$  indicates the completion of the second phase which also means that the arrival has taken place, hence the value of WIP level is increased by one and the phase of arrival is reset to zero. The rest of the states in Example 1 are also expanded accordingly. The total number of states in this example increases to 18, compared with 10 that we originally had in Example 1.

The failure and repair processes remain the same except that the tool can fail in each phase of arrival and hence there are transitions between  $[0, 0, 1]$  and  $[1, 0, 1]$  as well as between  $[0, 1, 0]$  and  $[1, 1, 0]$ . When the tool is active the processing of lots can take place in both phases of arrival. The operational rule is also in effect in the same manner as Example 1 and for states with WIP level greater than one the repair rate is doubled.

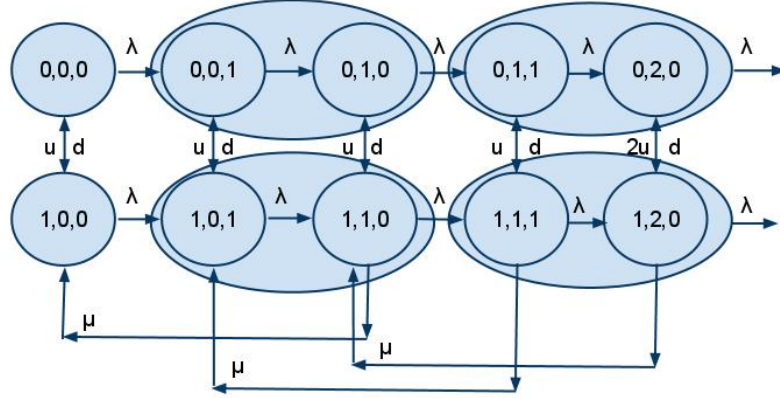


Figure 7: Rate Diagram of Example 2

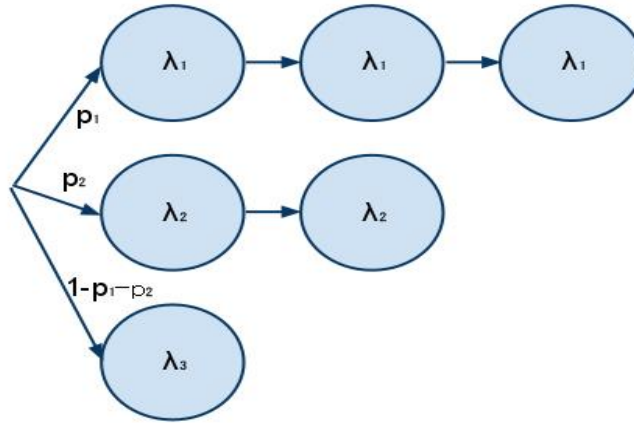


Figure 8: The Three-Branch Hyper-Erlang Distribution

## 4.2 Fitting Hyper-Erlang Distributions

As discussed in Section 4.1 neither the Exponential Fit nor the Erlang Fit can produce distributions with  $SCV > 1$ . In this section we propose a mixture of exponential phases that can produce probability density functions with  $SCV > 1$ . This mixture distribution is composed of three branches of Erlang distribution with three, two and one phases, respectively. Each branch of this distribution is followed by a certain probability and the sum of the probabilities of the three branches add to one. We refer to this mixture distribution as the Hyper-Erlang Distribution. Figure 8 shows a diagram of phases for such a distribution.

Thummler *et al.* [85] give the probability density function of the Hyper-Erlang mixture as  $f(x) = \frac{p_1}{2} \lambda_1^3 x^2 e^{-x\lambda_1} + p_2 \lambda_2^2 x e^{-x\lambda_2} + (1 - p_1 - p_2) \lambda_3 e^{-x\lambda_3}$ . Given a set of data or a continuous distribution function our aim in this section is to fit the Hyper-Erlang mixture distribution to it and then apply it in the Markov chain model.

In order to fit the Hyper-Erlang distribution to a continuous distribution the values of  $p_1, \lambda_1, p_2, \lambda_2$  and  $\lambda_3$  are to be estimated first. Thummler *et al.* [85] propose an iterative algorithm to find these

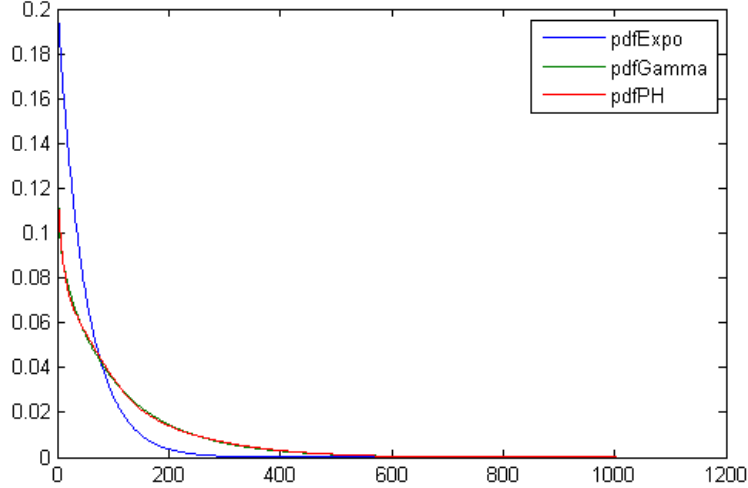


Figure 9: Gamma Fitted by Exponential and Hyper-Erlang

parameters using the Expectation Maximization (EM) algorithm.

The EM algorithm first proposed by [74], is a general procedure that finds a proper mixture of exponential phases (with a general structure) by matching the MLE of a dataset with that of the mixture distribution. The EM algorithm is based on minimizing the difference between the Maximum Likelihood Estimate (MLE) of the data generated from the initial distribution and the fitted exponential mixture.

Thummler *et al.* [85] tailor this algorithm and propose an iterative procedure that gives the parameters of the Hyper-Erlang distribution for a dataset or generated random variables from any continuous distribution, based on the EM algorithm. This tailored algorithm starts with a set of arbitrary initial values for the parameters of the Hyper-Erlang distribution, namely  $p_1, \lambda_1, p_2, \lambda_2$  and  $\lambda_3$ . Then it compares the MLE of the resulting Hyper-Erlang distribution with the MLE of the dataset and in each iteration improves the estimation for the parameters  $p_1, \lambda_1, p_2, \lambda_2$  and  $\lambda_3$  by maximizing the log-likelihood of the the data generated from the initial distribution.

Thummler *et al.* [85] prove that each iteration of this algorithm improves the estimation of parameters  $p_1, \lambda_1, p_2, \lambda_2$  and  $\lambda_3$  and that the iterative algorithm converges. The stopping criteria of the algorithm [85] is based on the difference between the MLE of the initial distribution and the resulting Hyper-Erlang distribution. When this difference becomes smaller than a chosen value  $\epsilon$ , the algorithm is stopped and the last estimation for values of  $p_1, \lambda_1, p_2, \lambda_2$  and  $\lambda_3$  is taken as the final estimate.

We refer to this fitting method as the Hyper-Erlang Fit method and apply it to non-exponential underlying distributions of  $a(t)$  or  $b(t)$  with  $SCV > 1$ . In order to implement the Hyper-Erlang Fit method we first generate  $10^4$  random samples of the initial distribution and form a dataset. We choose  $\epsilon = 10^{-6}$  and apply the algorithm proposed by [85] to estimate the parameters of the Hyper-Erlang distribution, namely  $p_1, \lambda_1, p_2, \lambda_2$  and  $\lambda_3$ . Figure 9 shows the *pdf* of a  $gamma(0.91, 12.6)$  distribution approximated by both the Exponential Fit method and the Hyper-Erlang Fit method.

#### 4.2.1 The Markov Chain Model with Hyper-Erlang Distribution

To model the inter-arrival or processing times in a toolset as the Hyper-Erlang distribution we modify the stochastic process of the basic Markov chain model of Chapter 3 to  $\{[K_i(t), W(t), B(t)], t \geq 0, i =$

$1, \dots, k^{\max}$ , where variable  $K_i(t) \in \{F, I, (0, 0), (2, 1), (1, 1), (1, 2)\}$ ,  $i = 1, \dots, k^{\max}$  denotes the status of the  $i^{\text{th}}$  tool. In this notation  $K_i(t) = F$  indicates that tool  $i$  is in a *failed* state at time  $t$ ,  $K_i(t) = I$  indicates that tool  $i$  is in *idle* state and  $K_i(t) = (0, 0)$  show that tool  $i$  has just finished processing a lot through one of the three branches of the Hyper-Erlang distribution. When  $K_i(t) = (2, 1)$  it indicates that the processing time of the lot on the  $i^{\text{th}}$  tool is following the second branch of the Hyper-Erlang distribution with two phases of exponential in series, each with parameter  $\lambda_2$  and that the processing time is in its first phase of the second branch. Similarly  $K_i(t) = (1, 1)$  denotes that the process is in the first phase of the first branch of the Hyper-Erlang distribution with parameter  $\lambda_1$  and  $K_i(t) = (1, 2)$  locates the process in the second phase of the first branch of the Hyper-Erlang distribution, also with parameter  $\lambda_1$ .

Variable  $B(t)$  in the stochastic process  $\{[K_i(t), W(t), B(t)], t \geq 0\}$ ,  $i = 1, \dots, k^{\max}$  keeps track of the Hyper-Erlang mixture in the arrival process. By  $B(t) \in \{(0, 0), (2, 1), (1, 1), (1, 2)\}$  we denote the branch and phase of the Hyper-Erlang distribution for the arrival process at time  $t$ .

Similar to the Erlang Fit method this extension of state-space also increases the number of states of the Markov model. To calculate the resulting number of states with this method we can apply the procedure suggested in Appendix III with the parameters  $A^{\max} = 3$  and  $P^{\max} = 3$ .

#### 4.2.2 Example 3 - Example 1 Revisited with Hyper-Erlang Distribution

For Example 1 presented in Chapter 3, suppose that we model the inter-arrival time distribution with the Hyper-Erlang distribution instead of exponential but the rest of the parameters of Example 1 remain the same. The parameters of the toolset for this example are summarized below.

$a(t)$	$b(t)$	$f(t)$	$r(t)$	$k^{\max}$	$w^{\max}$	Rule Parameters
$p_1, \lambda_1, p_2, \lambda_2, \lambda_3$	$\mu$	$d$	$u$	1	4	Rule II, $\tilde{w} = 1, \tilde{u} = 2u$

To reflect this change in the Markov model we need to add a new dimension to each state in the original formulation of Example 1 to track the phase of the inter-arrival distribution. The state-space for the new model is defined as  $\{[K(t), W(t), B(t)], t \geq 0, B(t) = (0, 0), (2, 1), (1, 1), (1, 2)\}$ . Figure 10 shows a partial view of the rate diagram for the Markov chain of this example with the inter-arrival time distribution modeled as Hyper-Erlang mixture.

For example between the state  $[0, 0]$  and state  $[0, 1]$  in Example 1 three states of  $\{[0, 0, (1, 1)], [0, 0, (1, 2)], [0, 0, (2, 1)]\}$  are inserted to keep track of the branch and phases of the Hyper-Erlang distribution. In the new rate diagram states  $[0, 0, (1, 1)]$  and  $[0, 0, (1, 2)]$  indicate the completion of the first and second phases of arrival in branch one, respectively. Similarly, state  $[0, 0, (2, 1)]$  indicates the completion of the first phase of branch two. Finally state  $[0, 1, (0, 0)]$  indicates that the arrival has taken place and the value of WIP is increased by one; hence the branch and phase of arrival process is reset to zero. The rest of the states in Example 1 are also expanded accordingly.

The failure and repair processes remain the same with the exception that the tool can fail in each phase of arrival and hence for example there are transitions between  $[0, 0, (1, 1)]$  and  $[1, 0, (1, 1)]$ . When the tool is active the processing of lots can take place in all phases of arrival too. The operational rule is also in effect in the same manner as Example 1 and for states with the level of WIP greater than 1 the repair rate is doubled. The total number of states in this example increases to 34, compared with 10 states in the original case of Example 1.

### 4.3 Solving The Steady-State Equations Using QBD

As it can be observed in this chapter, the extensions of the state-space to accommodate toolsets with non-exponential distributions cause a growth in the state-space of the Markov chain model and

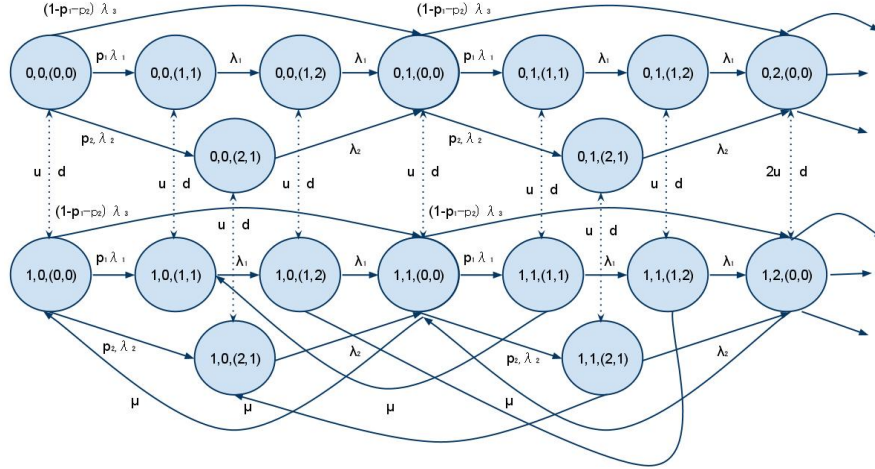


Figure 10: Rate Diagram of Example 3

can make the solution procedure inefficient. In this section we introduce an efficient algorithm to solve the steady state equations of the extended Markov chain model with a large number of states based on the Quasi Birth-Death Processes (QBD). To this end we first introduce the definition and characteristics of a QBD. Then we show that our proposed model also shares these characteristics and therefore can be classified as a QBD, allowing us to apply the efficient QBD algorithms to solve the balance equations in our model.

#### 4.3.1 Definition of Quasi Birth-Death Processes

Latouche and Ramaswami [80] give the following definition for a Quasi-Birth-Death (QBD) Process.

Definition - A continuous time homogeneous QBD is a Markov process with the following properties:

- It has a two-dimensional state space  $[i, j]$ ,  $i = 0, \dots, l$ ,  $j = 1, \dots, m$  where  $[i, j] \in \{(0, 1), (0, 2), \dots, (0, m), \dots, (l, 1), (l, 2), \dots, (l, m)\}$  for all  $l \geq 1$ ;  $m$  and  $l$  may be infinite. The subset of all the states that their first dimension is  $i$ , ( $i = 0, \dots, l$ ) is called level  $i$ .
- One-step transitions from a state are restricted to states in the same or in the two adjacent levels. In other words, a transition from  $(i, j)$  to  $(i', j')$  is not possible if  $|i' - i| \geq 2$ .
- For  $l \geq 1$  the instantaneous transition rate between two states in the same level or between two adjacent levels might not depend on  $l$ .

The infinitesimal generator  $\mathbf{Q}$  for such a process has the following structure:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{B}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

where  $\mathbf{A}_0, \mathbf{A}_1$  and  $\mathbf{A}_2$  are squared matrices of order  $m$  and  $\mathbf{B}_0, \mathbf{B}_1$  and  $\mathbf{B}_2$  are also matrices of appropriate size.

For example consider the case of Example 2, which is a queueing system with a single tool, inter-arrival times distributed as *Erlang* - 2 with rate  $\lambda$  and processing time exponentially distributed

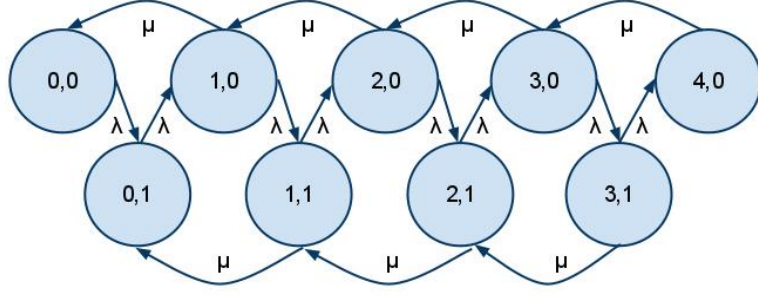


Figure 11: Rate Diagram of Example 2 with No Breakdown

with rate  $\mu$ . For simplicity we assume that the tool never fails and hence no failure or repair process is defined. We define the state space of such a system as  $[w, a]$  where  $w$  indicates the number of jobs in the system and  $a$  shows the current phase of the arrival process and hence  $w = 0, 1, \dots, 4$  and  $a = 0, 1$ . Note that in this special case of Example 2 since no failure and repair process is assumed we do not need to keep track of the number of active tools and it is assumed that the single tool of this toolset is available for production at all times. Hence the state space can be defined using only two variables of  $w$  and  $a$ . If the states are enumerated and ordered lexicographically then the state-space can be shown as

$$S = \{(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1), (3, 0), (3, 1), (4, 0)\}.$$

The transition rate diagram of this system is shown in Figure 11.

In this example state  $(0, 0)$  pertains to the situation where there are no jobs in the system and the arrival process is in its first phase. Similarly state  $(0, 1)$  pertains to the situation where the first phase of the arrival is finished and the second phase has been started with no jobs in the system. Finally, state  $(1, 0)$  indicates that the first arrival has taken place and the phase of the next arrival is reset to zero. The rest of the states are interpreted in a similar fashion. The infinitesimal generator matrix  $\mathbf{Q}$  of this process is defined as

$$\begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda & \lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & 0 & -\lambda - \mu & \lambda & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & -\lambda - \mu & \lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 & -\lambda - \mu & \lambda & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & -\lambda - \mu & \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 & -\lambda - \mu & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu & 0 & -\lambda - \mu & \lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mu & 0 & -\lambda - \mu & \lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mu & 0 & -\mu \end{bmatrix}.$$

We define the following sub-matrices.

$$\mathbf{B}_1 = \begin{bmatrix} -\lambda & \lambda \\ 0 & -\lambda \end{bmatrix}, \mathbf{B}_0 = \begin{bmatrix} 0 & 0 \\ \lambda & 0 \end{bmatrix}, \mathbf{B}_2 = \begin{bmatrix} \mu & 0 \\ 0 & \mu \end{bmatrix}$$

$$\mathbf{A}_1 = \begin{bmatrix} -\lambda - \mu & \lambda \\ 0 & -\lambda - \mu \end{bmatrix}, \mathbf{A}_0 = \begin{bmatrix} 0 & 0 \\ \lambda & 0 \end{bmatrix}, \mathbf{A}_2 = \begin{bmatrix} \mu & 0 \\ 0 & \mu \end{bmatrix}$$

Based on these definitions the infinitesimal generator matrix  $Q$  can be rewritten as follows.

$$\mathbf{Q} = \begin{bmatrix} B_1 & B_0 & 0 & 0 & 0 & \dots \\ B_2 & A_1 & A_0 & 0 & 0 & \dots \\ 0 & A_2 & A_1 & A_0 & 0 & \dots \\ 0 & 0 & A_2 & A_1 & A_0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

The new structure of  $\mathbf{Q}$  resembles the infinitesimal generator for an  $M/M/1$  queue which is known to have a birth-death process in its underlying Markov chain; hence the name Quasi Birth-Death is used for such systems. Latouche and Ramaswami [80] show that this method of modeling QBD's can also be extended to more dimensions.

For example in the case of Example 2 with non-exponential underlying distribution of inter-arrival times and in presence of failure and repair processes the state variable needs to at least have three dimensions  $[k, w, a]$ , where  $k$  indicates the number of active tools,  $w$  indicates the number of lots at the toolset and  $a$  indicates the current phase of the *Erlang* – 2 distribution of the arrival process. Latouche and Ramaswami [80] show that by grouping two of the three variables (e.g.,  $w$  and  $a$ ) we can construct a two dimensional QBD within the initial stochastic process and the combination of the first variable, namely  $k$  with the QBD consisting of  $[w, a]$  also classifies as a QBD. In other words they show that the stochastic process  $\{[K(t), W(t), A(t)], t \geq 0\}$  with the two-dimensional process of  $[W(t), A(t)]$  within it also possesses the form of a QBD. This applies to all the forms of extended state-space that we have introduced so far in this chapter and will also apply to the extended Markov chains of the following chapter with extensions to heterogeneous tools and multiple failure types.

#### 4.3.2 A Solution Algorithm for QBD Processes

As discussed in Chapter 3 the solution for the system of equations  $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$  and  $\boldsymbol{\pi}\mathbf{1} = 1$  will find the steady state probabilities of a given Markov chain with the state probability vector  $\boldsymbol{\pi}$ . In this notation  $\boldsymbol{\pi}$  corresponds to a row vector with elements  $\pi_{ij}$  in the increasing order of  $j$  and then  $i$ . For example in the case of Example 1 in Chapter 3 the vector  $\boldsymbol{\pi}$  will be written as  $\boldsymbol{\pi} = [\pi_{00} \ \pi_{01} \ \pi_{02} \ \pi_{11} \ \pi_{12} \ \dots]$ . For brevity we group the elements of this vector in sub-vectors such that sub-vector  $\boldsymbol{\pi}_k$  correspond to a column vector with all elements  $\pi_{k,j}$  in  $\boldsymbol{\pi}$  for  $j = 0, 1, 2$ . Therefore we denote vector  $\boldsymbol{\pi}$  as  $\boldsymbol{\pi} = [\boldsymbol{\pi}_0 \ \boldsymbol{\pi}_1 \ \boldsymbol{\pi}_2 \ \dots]$ , where each element in this vector represents a group of states that are common in their first element of their state variable.

Evans [83] shows that a solution exists that satisfies the set of equations  $\boldsymbol{\pi}_{i+1} = \boldsymbol{\pi}_i\mathbf{R}$  for  $i = 1, \dots, l$  and matrix  $\mathbf{R}$  is a square matrix of order  $m$ . Matrix  $\mathbf{R}$  is the solution to the matrix-quadratic equation  $\mathbf{A}_0 + \mathbf{R}\mathbf{A}_1 + \mathbf{R}^2\mathbf{A}_2 = \mathbf{0}$ .

Various algorithms have been proposed to compute matrix  $\mathbf{R}$  including [80], [65], [84] and references therein. In this research we use an iterative algorithm introduced by [65] that starts with an arbitrary initial guess  $\mathbf{R}_0$  and obtains a series of  $\mathbf{R}_k$  matrices through iterations of the form  $\mathbf{R}_{k+1} = -(\mathbf{A}_0 + \mathbf{R}_k^2\mathbf{A}_2)\mathbf{A}_1^{-1}$ . It is shown that this process converges and that  $\mathbf{A}_1^{-1}$  exists. Once  $\mathbf{R}$  is determined then  $\boldsymbol{\pi}_0$  and  $\boldsymbol{\pi}_1$  are determined by solving the following linear system of equations. Then the rest of the steady-state probability vectors are computed iteratively through the equation  $\boldsymbol{\pi}_{i+1} = \boldsymbol{\pi}_i\mathbf{R}$  for  $i = 1, \dots, l$ .

$$\begin{aligned} [\boldsymbol{\pi}_0 \ \boldsymbol{\pi}_1] \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_0 \\ \mathbf{B}_2 & \mathbf{A}_1 + \mathbf{R}\mathbf{A}_2 \end{bmatrix} &= [\mathbf{0} \ \mathbf{0}] \\ \boldsymbol{\pi}_0\mathbf{1} + \boldsymbol{\pi}_1(\mathbf{I} - \mathbf{R})^{-1}\mathbf{1} &= 1. \end{aligned}$$

Given the vector  $\boldsymbol{\pi} = [\pi_0 \ \pi_1 \ \pi_2 \ \dots]$  and the long-run average inter-arrival time we can apply the Little's Law as explained in Chapter 3 in order to compute the long-run average cycle time of the toolset with non-exponential underlying distributions.

#### 4.4 Chapter Summary and Conclusions

In this chapter methods to approximate non-exponential distributions with mixtures of exponential phases are discussed. Specifically, three distribution fitting methods of Exponential Fit, Erlang Fit and Hyper-Erlang Fit are presented. These fitting methods allow for the integration of the non-exponential underlying distributions of arrival and service processes in the proposed Markov chain model of Chapter 3. However, applying these fitting methods to the Markov chain model expands the state-space and renders the solution time longer. To improve the computational efficiency of the Markov chain framework the solution algorithms for Quasi Birth-Death (QBD) processes are introduced. These algorithms provide efficient computational procedures to solve the steady-state equations of the proposed framework.

In the next chapter the Markov model of Chapter 3 is further extended to include more complexities of the SMS toolsets. Two models are specifically developed for toolsets with heterogeneous tools and toolsets with multiple types of failure.

## 5 Extensions To More Complex Toolsets

The basic Markov chain model of Chapter 3 is constructed for a toolset with  $k^{\max}$  parallel tools that are identical in their service process attributes. In this chapter we introduce an extension to this basic model to include toolsets with similar but not identical (heterogeneous) tools. Heterogeneous tools can perform the same operation on the wafers but are different in their processing time, time to failure or time to repair attributes. This difference in the tools complicates the cycle time estimation of a toolset.

The proposed model of Chapter 3 also considers only one type of failure and repair processes for each tool, however in reality SMS toolsets are subject to various types of failure with distinct repair procedures. In this chapter we also extend the basic Markov chain model of Chapter 3 to include two distinct types of failure with their corresponding repair procedures for each tool within a toolset.

### 5.1 Toolsets with Heterogeneous Tools

Existence of heterogeneous tools within the same toolset is very common in SMS. This happens when the same type of tools from different manufacturers are grouped together or when an older generation tool is upgraded to perform in a new technology and is grouped with new generation tools in the same toolset. Heterogeneous tools often differ from each other in their distributions of processing time, time to fail or time to repair.

For example, usually newer generations of tools are designed to open and close easier and faster in order to perform a PM and they are often faster processors than their predecessors. However, it is not usual to have more than two groups of heterogeneous tools within one toolset. In a heterogeneous toolset the operation managers usually have a preference for one group of tools over the other due to faster processing time, relative ease of maintenance, or less chance of failure during the process.

We define the preferred tool group as group 1 and the non-preferred group as group 2. We assume that group 1 and group 2 each have  $K_1^{\max}$  and  $K_2^{\max}$  tools, respectively. We define the service related underlying distributions of group 1 as  $b_1(t)$ ,  $f_1(t)$  and  $r_1(t)$  and for group 2 we define these distributions as  $b_2(t)$ ,  $f_2(t)$  and  $r_2(t)$  and assume that in a toolset with heterogeneous tools at least one of these underlying distribution pairs of  $\{b_1(t), b_2(t)\}$ ,  $\{f_1(t), f_2(t)\}$  or  $\{r_1(t), r_2(t)\}$  are different between group 1 and group 2.

To model a toolset with heterogeneous tools using the Markov chain model of Chapter 3 we define  $K_1$  as the number of active tools in group 1 (the preferred tool group) and  $K_2$  as the number of active tools in group 2. Due to this preference all arriving lots first occupy the active and idle tools in the first group, and the active tools in the second are only utilized if all active tools in the first group are busy.

To reflect the existence of heterogeneous tools in the Markov chain model of the toolset we add an auxiliary variable showing how many tools of each tool group are busy. Hence we extend the state-space of the Markov chain model to  $\{[K_1(t), K_2(t), K_1^b, K_2^b, W(t)], t \geq 0\}$ , where  $K_i(t)$  shows the number of active tools in group  $i$  and  $K_i^b$  indicates the number of busy tools of tool group  $i$ , for  $i = 1, 2$ . This definition of state-space allows us to track the processing of lots on each tool group and assign the lots to active tools according to the tool preferences.

In the case of non-exponential underlying distributions of  $a(t)$ ,  $b_1(t)$  and  $b_2(t)$  the state-space of the heterogeneous toolset is further expanded to track the phases of arrival or service processes. The state-space for a toolset with two heterogeneous tools and non-exponential arrival process is defined as  $\{[K_1(t), K_2(t), K_1^b, K_2^b, W(t), A(t)], t \geq 0\}$  where variable  $A(t)$  is added to the basic structure of the heterogeneous state-space in order to keep track of the phases of the arrival process according to an Erlang distribution as defined in Section 4.1, for the Erlang distribution with  $A^{\max}$  phases where  $A(t)$  can take on values between 0 and  $A^{\max} - 1$ . Similarly in the case of Hyper-Erlang underlying

distribution of  $a(t)$  the stochastic process is extended to  $\{[K_1(t), K_2(t), K_1^b, K_2^b, W(t), B(t)], t \geq 0\}$  with  $B(t) = (0, 0), (2, 1), (1, 1), (1, 2)$ .

Likewise, when the service process of the toolset is non-exponential the state-space of the basic model for heterogeneous tools should be expanded to keep track of the phases of the processing time. In this case the variables  $K_1^b$  and  $K_2^b$  are changed to  $K_1^i$  and  $K_2^j$ ,  $i = 1, \dots, K_1^{\max}$  and  $j = 1, \dots, K_2^{\max}$  where the superscripts  $i$  and  $j$  indicate the specific tools within each group. In this notation variable  $K_1^i$  can take on values between 0 and  $P^{\max} - 1$  for an Erlang distribution with  $P^{\max}$  phases or  $K_1^i = (0, 0), (2, 1), (1, 1), (1, 2)$  for the Hyper-Erlang distribution. More specifically  $K_1^i$  shows the phase of the processing time distribution for tool  $i$  of group 1 and in a similar manner  $K_2^j = 0, \dots, P^{\max} - 1$  or  $K_2^j = (0, 0), (2, 1), (1, 1), (1, 2)$  show the phases of the processing time for tool  $j$  of group 2. Therefore, for a toolset with heterogeneous tools and non-exponential underlying distributions of both  $a(t)$  and  $b(t)$  the state-space is defined as  $\{[K_1(t), K_2(t), K_1^i, K_2^j, W(t), A(t)], t \geq 0, i = 1, \dots, K_1^{\max}, j = 1, \dots, K_2^{\max}\}$  or  $\{[K_1(t), K_2(t), K_1^i, K_2^j, W(t), B(t)], t \geq 0, i = 1, \dots, K_1^{\max}, j = 1, \dots, K_2^{\max}\}$  depending on the fitting method used for  $a(t)$ .

### 5.1.1 Example 4 - A Toolset with Two Heterogeneous tools

Consider a toolset with two heterogeneous tools, *i.e.*,  $K_1^{\max} = 1$ ,  $K_2^{\max} = 1$  and  $w^{\max} = 4$ , arrival rate  $\lambda$ , processing rates  $\mu_1$  for tool 1 and  $\mu_2$  for tool 2, failure rate  $d$  and repair rate  $u$  for both tools, and Rule *II* with  $\tilde{w} = 1$ ,  $\tilde{u} = 2u$ . Suppose that all the underlying distributions of the toolset are exponential. Also assume that tool 1 is preferred over tool 2, so that in the case where both tools are idle and an arrival takes place the arriving lot occupies tool 1. The state variable of such a system is defined as  $[k_1, k_2, k_1^b, k_2^b, w]$ . In this notation  $k_1 = 0, 1$  shows the number of active tools of group 1,  $k_2 = 0, 1$  shows this number for group 2,  $k_1^b$  shows the number of busy tools for group 1 and  $k_2^b$  shows the similar number for group 2. Finally  $w$  shows the number of lots at the toolset. The transition rate diagram of this toolset is shown in Figure 12. The total number of states in this example is 21 states.

## 5.2 Toolsets with Multiple Failure Types

SMS tools are subject to different types of failure such as scheduled versus unscheduled maintenance or short term versus long term unforeseen failures. To model this phenomenon we assume that each tool within the toolset is subject to two categories of failures, type *I* and type *II* with distinct time to fail and time to repair distributions. We denote by  $f_I(t)$  the time to failure probability density function for type *I* failure with expected time to failure of  $\frac{1}{d_I}$  and by  $r_I(t)$  the time to repair of a type *I* failure with the expected repair time of  $\frac{1}{u_I}$ . Similarly we denote the type *II* time to failure distribution by  $f_{II}(t)$  and its time to repair distribution by  $r_{II}(t)$  with expected values of  $\frac{1}{d_{II}}$  and  $\frac{1}{u_{II}}$ , respectively.

Like single failure tools, for toolsets with two failure types we also assume that the time to fail and time to repair for each type of failure, namely  $f_I(t)$ ,  $r_I(t)$ ,  $f_{II}(t)$  and  $r_{II}(t)$  are exponentially distributed with parameters  $d_I, u_I, d_{II}$ , and  $u_{II}$ , respectively.

To separately model the two types of failure and repair in the Markov chain model of Chapter 3 we modify its underlying stochastic process as follows. To track the repair rate corresponding to a specific type of failure at each state the state indicator needs to contain the information on the type of failure for each of the failed tools. Therefore, we extend the state-space of the basic Markov chain model to  $\{[K(t), S(t), W(t)], t \geq 0\}$  where the variable  $S(t)$  indicates the number of tools that are down with type *I* failure at time  $t$ . Since we know the total number of tools in the toolset ( $k^{\max}$ ) as well as the number of active tools,  $K(t)$  and the number of tools that are down with type *I* failure, finding the number of tools that are down with type *II* failure is trivial.

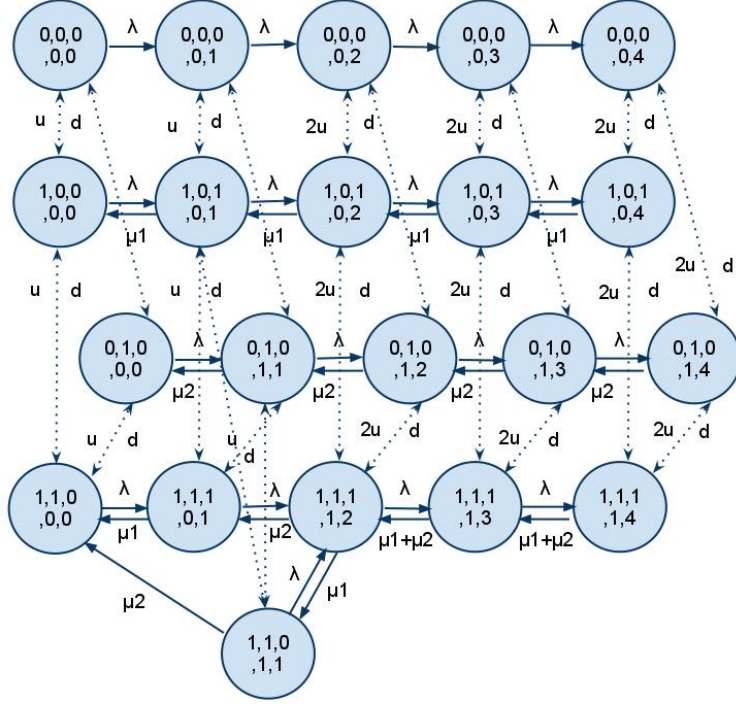


Figure 12: Rate Diagram of Toolset with Two Heterogeneous Tools

Similar to the model for heterogeneous tools, in the case of non-exponential underlying distributions of  $a(t)$  or  $b(t)$  the state-space of the multi-failure model is also expanded to keep track of the phases of arrival or processing time distributions. The state-space for a toolset with two failure types and non-exponential arrival process is defined as  $\{[K(t), S(t), W(t), A(t)], t \geq 0\}$  where variable  $A(t)$  tracks the phases of the arrival process according to an Erlang distribution with  $A^{\max}$  phases, where  $A(t) = 0, \dots, A^{\max} - 1$ . Similarly in the case of Hyper-Erlang distribution the stochastic process is modified to  $\{[K(t), S(t), W(t), B(t)], t \geq 0\}$ , where  $B(t) = (0, 0), (2, 1), (1, 1), (1, 2)$ .

Likewise, when the service process of the toolset is non-exponential the state-space of the basic model for multi-failure tools should be expanded to keep track of the phases of the processing time. In this case the variable  $K(t)$  is changed to  $K_i(t)$  for  $i = 1, \dots, k^{\max}$  where for the Erlang distribution with  $P^{\max}$  phases  $K_i(t) = 0, \dots, P^{\max} - 1$  and for the Hyper-Erlang distribution  $K_i(t) = (0, 0), (2, 1), (1, 1), (1, 2)$ . Therefore, for a toolset with non-exponential distributions of both  $a(t)$  and  $b(t)$  with two types of failure the state-space of the system is define as  $\{[K_i(t), S(t), W(t), A(t)], t \geq 0, i = 1, \dots, k^{\max}\}$  with  $A^{\max}$  phases of Erlang distribution in the arrival process or to  $\{[K_i(t), S(t), W(t), B(t)], t \geq 0, i = 1, \dots, k^{\max}\}$  with the Hyper-Erlang distribution in the arrival process.

### 5.2.1 Example 5 - Example 1 Revisited with Two Failure Types

Consider the toolset in Example 1 with the addition of a second category of failure. The parameters and characteristics of this toolset are shown below and Figure 13 shows the transition rate diagram of the Markov chain model for Example 5.

$a(t)$	$b(t)$	$f(t)$	$r(t)$	$k^{\max}$	$w^{\max}$	Rule Parameters
$\lambda$	$\mu$	$d_1, d_2$	$u_1, u_2$	1	4	Rule II, $\tilde{w} = 1, \tilde{u} = 2u$

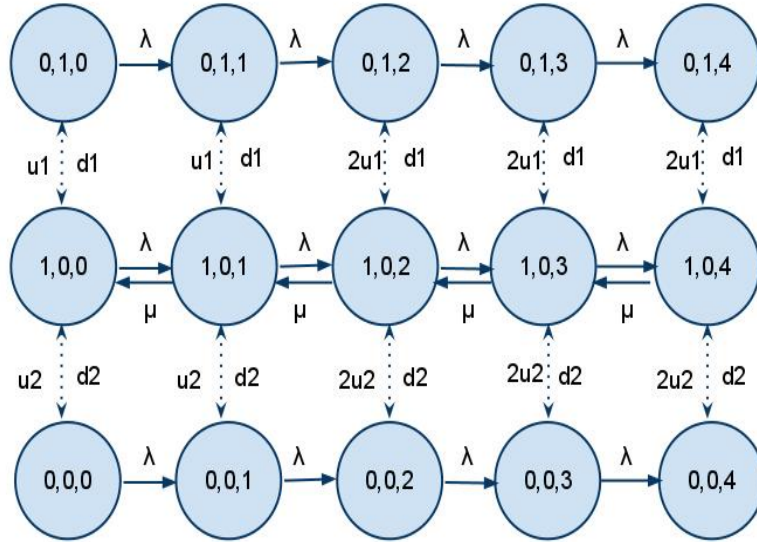


Figure 13: Rate Diagram of Toolset with Two Failure Types

The state descriptor for such a system is shown as  $[k, s, w]$  where  $k = 0, 1$  shows the number of active tools in each state,  $s = 0, 1$  indicates if the only tool of this toolset is down with type  $I$  failure and finally  $w = 0, \dots, 4$  indicates the WIP level at the toolset. The total number of states for this example is 15 with a 50% increase compared to Example 1 with 10 states.

### 5.3 Chapter Summary and Conclusions

In this chapter the basic structure of the Markov chain model in Chapter 3 is expanded to include toolsets with heterogeneous tools and multiple types of failure. These conditions are common in the real SMS toolsets and can highly impact the cycle time. Hence, integration of such situations in the Markov chain model can increase the accuracy of cycle time approximation for SMS toolsets.

In the past three chapters the details of the basic Markov chain model as well as extensions for non-exponential distributions, heterogeneous tools and multiple failure types are discussed. In the following chapter some numerical results are presented through a series of cases. In the first two cases the underlying distributions are considered to be exponential, Erlang or Hyper-Erlang and hence no fitting method is applied. These two cases are developed for model verification. The next four cases discuss various situations such as non-exponential underlying distributions approximated by the fitting methods of Chapter 4 and the complexities presented in Chapter 5. Finally in all these cases the performance of the proposed Markov chain framework is compared with the classical  $G/G/m$  approximations under no operational rules and the computational requirements of the proposed framework are discussed.

## 6 Numerical Results

In this chapter we first develop a measure to evaluate the performance of the proposed cycle time approximation method. Then we present some numerical results on the performance of this method by applying it to a simulated toolset with two parallel tools. We first develop a simulation model of this toolset as described in Chapter 4 and examine it under a series of scenarios that are different in their inter-arrival time distribution  $a(t)$ , processing time distribution  $b(t)$ , or the operational rule applied. For each instance we run the simulation model for 30 replications and each replication for 20,000 simulation hours and a warm-up period of 500 hours.

We consider two different forms of the probability distributions for each of  $a(t)$  and  $b(t)$ . In the first form we use exponential or exponential mixture density functions for both  $a(t)$  and  $b(t)$ , and in the second form we consider non-exponential density functions (specifically, we use lognormal and gamma density functions to represent the inter-arrival or processing time distributions). The objective of building cases with exponential and exponential mixture underlying distributions is to validate the proposed framework, since in such cases there is no approximation involved. Therefore, we expect the cycle time prediction of the Markov model to be very close to that of the corresponding simulated toolset.

The purpose of cases with non-exponential underlying distributions is to study the performance of the proposed framework with various shapes of distributions and observe the effect of approximating the non-exponential distributions. For these cases with non-exponential underlying distributions we apply the methods of approximation discussed in Chapter 4, namely *Exponential Fit*, *Erlang Fit* and *Hyper-Erlang Fit* as appropriate.

We also examine the framework with two cases of a toolset with heterogeneous tools and a toolset with two types of failure, both with Erlang underlying distributions. Additionally, in all the above-mentioned cases for the instances with no operational rule we compare the results with the cycle time approximation of two  $G/G/m$  models that are commonly used in manufacturing and are previously discussed in Chapter 2.

In all cases in this chapter the distribution of time to fail and time to repair, *i.e.*,  $f(t)$  and  $r(t)$  are assumed to be exponential. For each toolset we apply three policies regarding the operational rules, namely absence of operational rules (No Rule), Rule *I* and Rule *II*. Recall from Chapter 3 that Rule *I* changes the average arrival rate from  $\lambda$  to  $\tilde{\lambda} < \lambda$  when the WIP level is greater than  $\tilde{w}$  and Rule *II* changes the average repair rate from  $u$  to  $\tilde{u} > u$  when the WIP level is greater than  $\tilde{w}$ .

To conclude this chapter we examine the computational requirements of the proposed model and compare it with simulation. The results show that in the majority of instances the Markov model outperforms the simulation in terms of solution time.

### 6.1 A Method for Performance Evaluation

Before presenting the numerical results it is necessary to develop a method to evaluate the performance of the proposed cycle time approximation framework with respect to the simulation model, which is considered to be the actual SMS toolset throughout this research. Since the operational rules that we consider in this research are both WIP dependent we expect to observe a higher impact of the rules in high utilization where the level of WIP is expected to be higher. The effects of failure and repair times on cycle time are also more visible in high traffic.

In order to remove the possible bias caused by closer approximations at low utilization levels we compare the results of the Markov model with those of the simulation only at higher utilization levels. Specifically we consider three levels of utilization namely,  $\rho = 70\%$ ,  $\rho = 80\%$  and  $\rho = 90\%$ . At these three levels we evaluate the cycle time of a given toolset through simulation and the proposed Markov model.

Figure 14 shows an example where the cycle time of a toolset is evaluated in different levels of

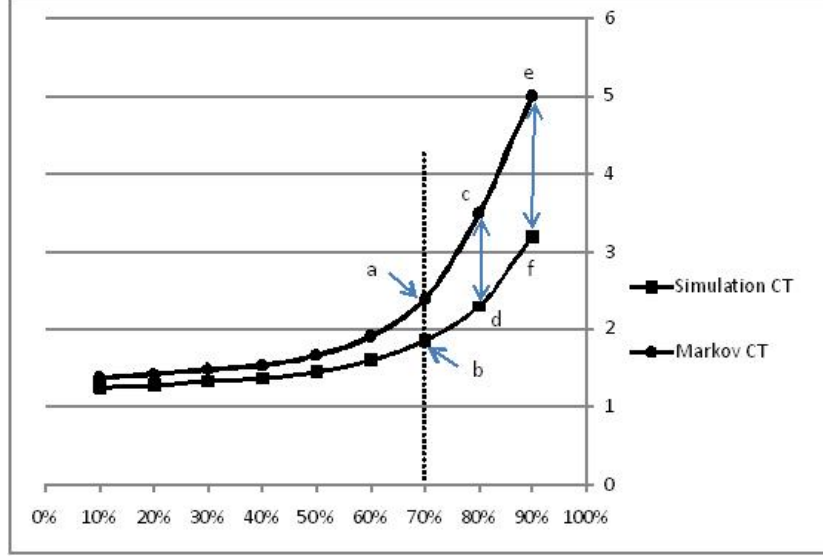


Figure 14: Calculation of Relative Error

utilization through both models. In this figure the X-axis shows the levels of utilization and the Y-axis shows the cycle time. A vertical dotted line marks the 70% utilization level. The Markov chain model evaluates the cycle time of this toolset at 70%, 80%, and 90% utilization as points  $a$ ,  $c$ , and  $e$ , respectively, and the simulation model finds points  $b$ ,  $d$ , and  $f$  for the same levels of utilization.

To compare the performance of the Markov chain model with simulation we introduce the following measure. For each of the three levels of utilization mentioned above (*i.e.*,  $\rho = 70\%$ ,  $\rho = 80\%$  and  $\rho = 90\%$ ) we calculate the absolute value of the relative error for the Markov chain approximation based on the results of the simulation model, and denote it by  $E_\rho$ . Then we take the average of these three values, namely  $E_{0.7}$ ,  $E_{0.8}$  and  $E_{0.9}$ . For the example of Figure 14 we calculate the relative error as follow.

$$\begin{array}{c|c|c} \rho = 70\% & \rho = 80\% & \rho = 90\% \\ \hline E_{0.7} = \left| \frac{a-b}{b} \right| & E_{0.8} = \left| \frac{c-d}{d} \right| & E_{0.9} = \left| \frac{e-f}{f} \right| \end{array}$$

We then calculate the average of  $E_{0.7}$ ,  $E_{0.8}$  and  $E_{0.9}$  as  $E_{avg} = \frac{E_{0.7} + E_{0.8} + E_{0.9}}{3}$  and consider the *average relative error*, denoted by  $E_{avg}$ , as the measure of performance of the proposed model. In the following tables and discussions we consistently multiply the value of  $E_{avg}$  by 100 and refer to it as *average error percentage*. Note that smaller values of  $E_{avg}$  indicate closer approximation of cycle time by the proposed model and hence better performance. Throughout the rest of this document we use  $E_{avg}$  as the measure of performance for the proposed cycle time approximation model.

## 6.2 Case 1 - Toolset with Erlang Underlying Distributions

In this case we assume that the underlying distributions of inter-arrival and processing times of the toolset or  $a(t)$  and  $b(t)$  are Erlang. Since the Erlang distribution retains the memoryless property, there is no need to employ a fitting method and we can directly apply the rate parameters of  $a(t)$  and

$b(t)$  as the transition rates in the corresponding state-dependent Markov chain model. As a result no approximation is involved, and we expect the approximated cycle time of the Markov model to be very close to the mean cycle time found by the simulation model of the same toolset. The objective of this case study is to validate the proposed model when no approximation of underlying distributions is involved.

We examine the system under three different sets of distributions for  $a(t)$  and  $b(t)$ , respectively. These distributions include *Erlang* – 1 (exponential), *Erlang* – 2, and *Erlang* – 10. The rate parameters of  $a(t)$  and  $b(t)$  are denoted by  $\lambda$  and  $\mu$ , respectively, and their values are shown in Table 6.2.1. To achieve different levels of utilization, *i.e.*,  $\rho = 70\%$ ,  $80\%$  and  $90\%$  the value of  $\lambda$  is changed as shown in Table 6.2.1. There are two parallel tools in this toolset ( $k^{\max} = 2$ ) that are subject to random failures with time to fail and time to repair following exponential distributions (*i.e.*,  $f(t)$  and  $r(t)$  are exponential *pdf's*) and initial parameters  $d = \frac{1}{4}$  and  $u = 1$ , respectively.

Table 6.2.1: Parameters of  $a(t)$  and  $b(t)$  for Case 1

Case 1		$\lambda$			$\mu$
<i>Distribution</i>	<i>SCV</i>	70%	80%	90%	
<i>Erlang</i> – 1	1	1.02	1.16	1.31	1
<i>Erlang</i> – 2	0.5	2.04	2.33	2.62	2
<i>Erlang</i> – 10	0.1	10.18	11.64	13.09	10

The maximum number of lots that are allowed in the system is  $w^{\max} = 59$ . The initial number of states in the Markov chain model of this toolset is  $(k^{\max} + 1) \times (w^{\max} + 1) = 180$ , when  $a(t)$  and  $b(t)$  both have exponential (*Erlang* – 1) probability density functions. However if we model the inter-arrival or processing time distributions with an Erlang distribution with higher number of phases then the number of states is calculated according to the method introduced in Appendix III. These values are later presented in Section 6.9 at the end of this chapter.

We consider three scenarios for the operational rules. In the first scenario the toolset has no operational rule (No Rule), the second scenario is with Rule *I* and the third scenario is with Rule *II*. The general characteristics of each rule are summarized in Table 6.2.2.

Table 6.2.2: Operational Rule Parameters

Rule Type	Parameter
No Rule	–
Rule <i>I</i>	$\tilde{w} = 2, \tilde{\lambda} = 0.5 \times \lambda$
Rule <i>II</i>	$\tilde{w} = 2, \tilde{u} = 2 \times u$

Since we consider three distinct distributions for each of  $a(t)$  and  $b(t)$ , each scenario consists of 9 instances with distinct combinations of arrival and processing time distributions. We verify the performance of the state-dependent Markov chain model in each instance by calculating the  $E_{avg}$  as discussed earlier. The results are presented in Table 6.2.3.

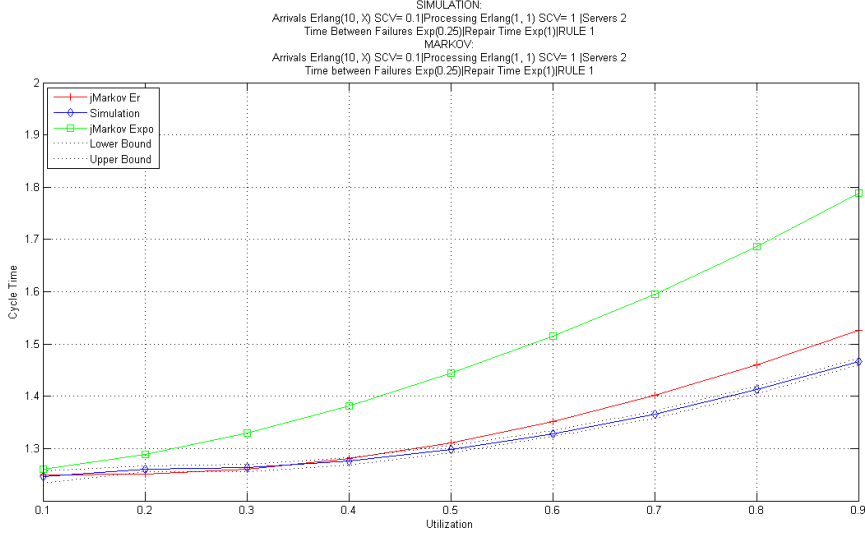


Figure 15: Utilization Graph of One Instance in Table 6.2.3

Table 6.2.3:  $E_{avg}$  with Erlang Distributions

Inter-arrival Time Distribution	Processing Time Distribution	Rule Scenario		
		No Rule	Rule I	Rule II
<i>Erlang</i> – 1	<i>Erlang</i> – 1	0.5%	4.5%	0.4%
<i>Erlang</i> – 2	<i>Erlang</i> – 1	0.8%	4.1%	2.1%
<i>Erlang</i> – 10	<i>Erlang</i> – 1	0.3%	3.4%	2.7%
<i>Erlang</i> – 1	<i>Erlang</i> – 2	0.5%	4.9%	0.3%
<i>Erlang</i> – 2	<i>Erlang</i> – 2	0.5%	4.5%	1.1%
<i>Erlang</i> – 10	<i>Erlang</i> – 2	0.5%	3.5%	2.5%
<i>Erlang</i> – 1	<i>Erlang</i> – 10	0.7%	5.0%	0.3%
<i>Erlang</i> – 2	<i>Erlang</i> – 10	0.5%	4.5%	0.7%
<i>Erlang</i> – 10	<i>Erlang</i> – 10	0.1%	3.7%	3.0%

For all instances in Table 6.2.3 the cycle time approximation of the Markov model at each utilization level is very close to the mean of cycle time for the simulation model. As observed in Table 6.2.3 the average error percentage,  $E_{avg}$ , does not exceed 5% in any of the instances.

Figure 15 shows the utilization graph for the instance with Rule I where  $a(t)$  is *Erlang* – 10 and  $b(t)$  is *Erlang* – 1 (exponential). The Y-axis shows the values for average cycle time of the toolset at different levels of utilization (X-axis) ranging from 10% to 90%. The values of average cycle time are evaluated both through the simulation model and the Markov model. In the case of simulation we also present the upper and lower 95% confidence interval limits by dotted lines along the simulation curve. Utilization graphs for every instance of Case 1 (*i.e.*, as in Table 6.2.3) are developed and analyzed. All graphs show a similar behavior as in Figure 15. For brevity we do not include all these graphs here. Utilization graphs for selected instances of Case 1 are presented in Appendix IV.

### 6.3 Case 2 - Toolset with Hyper-Erlang Underlying Distributions

In this case we assume that the underlying distributions of inter-arrival and processing times of the toolset, namely  $a(t)$  and  $b(t)$  are Hyper-Erlang mixtures with the structure shown in Figure 8. The Hyper-Erlang distribution also retains the memoryless property since it consists of exponential phases and hence like the Erlang underlying distributions of Case 1 there is no need to employ a fitting method in this case as well. We can directly apply the rate parameters of  $a(t)$  and  $b(t)$  as the transition rates of the extended Markov chain model as demonstrated in Chapter 4. Similar to Case 1, in this case too we expect the approximated cycle time of the Markov model to be very close to the mean cycle time found by the simulation model of the same toolset. This case is also presented with the objective of validating the proposed framework where no approximation of underlying distributions is involved.

We examine this system under three different sets of distributions for  $a(t)$  and  $b(t)$ , respectively. These distributions vary in their  $SCV$  and specifically we examine Hyper-Erlang distributions with  $SCV = 1.1$ ,  $SCV = 1.5$  and  $SCV = 2$ . The parameters of  $a(t)$  and  $b(t)$  are denoted by  $\lambda_1, \lambda_2, \lambda_3, p_1, p_2$  and  $\mu_1, \mu_2, \mu_3, q_1, q_2$  respectively, and their values are shown in Table 6.3.1 and Table 6.3.2. To achieve different levels of utilization, *i.e.*,  $\rho = 70\%$ ,  $80\%$  and  $90\%$ , the value of  $a(t)$  parameters are changed as shown in Table 6.3.1. Like Case 1 the two parallel tools in this toolset ( $k^{\max} = 2$ ) are also subject to random failures with time to fail and time to repair following exponential distributions and initial parameters  $d = \frac{1}{4}$  and  $u = 1$ , respectively.

Table 6.3.1: Parameters of  $a(t)$  for Case 2

Inter-arrival Time Parameters					
	$\lambda_1$	$\lambda_2$	$\lambda_3$	$p_1$	$p_2$
$\rho = 70\%$					
$SCV = 1.1$	0.34	0.74	1.33	0.41	0.40
$SCV = 1.5$	0.25	0.58	2.20	0.26	0.49
$SCV = 2$	0.23	0.72	4.46	0.27	0.46
$\rho = 80\%$					
$SCV = 1.1$	0.34	0.69	1.75	0.31	0.51
$SCV = 1.5$	0.29	0.67	2.72	0.26	0.50
$SCV = 2$	0.26	0.80	5.02	0.27	0.47
$\rho = 90\%$					
$SCV = 1.1$	0.44	0.92	1.72	0.41	0.40
$SCV = 1.5$	0.36	1.01	4.39	0.34	0.47
$SCV = 2$	0.30	0.90	5.73	0.27	0.47

Table 6.3.2: Parameters of  $b(t)$  for Case 2

Processing Time Parameters					
$SCV$	$\mu_1$	$\mu_2$	$\mu_3$	$q_1$	$q_2$
1.1	0.33	0.74	1.34	0.43	0.39
1.5	0.24	0.55	2.10	0.25	0.49
2	0.23	0.73	4.64	0.28	0.46

The maximum number of lots that are allowed in the system is  $w^{\max} = 59$ . The number of states for this case are calculated according to the method introduced in Appendix III. These values are later presented in Section 6.9 at the end of this chapter. We consider three scenarios for the operational rules as described in Table 6.2.2.

Since we consider three distinct distributions for each of  $a(t)$  and  $b(t)$  that vary in their  $SCV$ , each scenario consists of 9 instances or combinations of inter-arrival time and processing time dis-

tributions. We verify the performance of the state-dependent Markov chain model in each instance by calculating the  $E_{avg}$  as discussed earlier. The results are presented in Table 6.3.3.

Table 6.3.3:  $E_{avg}$  with Hyper-Erlang Distributions

Inter-arrival Time Distribution	Processing Time Distribution	Rule Scenario		
		No Rule	Rule I	Rule II
$SCV = 2$	$SCV = 2$	1.3%	1.7%	5.3%
$SCV = 1.5$	$SCV = 2$	3.8%	2.1%	4.8%
$SCV = 1.1$	$SCV = 2$	3.5%	4.3%	5.2%
$SCV = 2$	$SCV = 1.5$	3.2%	4.1%	2.8%
$SCV = 1.5$	$SCV = 1.5$	1.6%	3.4%	2.2%
$SCV = 1.1$	$SCV = 1.5$	3.4%	2.9%	4.7%
$SCV = 2$	$SCV = 1.1$	1.9%	3.5%	2.3%
$SCV = 1.5$	$SCV = 1.1$	3.1%	2.6%	3.9%
$SCV = 1.1$	$SCV = 1.1$	0.5%	1.7%	2.8%

For all instances in Table 6.3.2 the cycle time approximation of the Markov model at each utilization level is very close to the mean of cycle time for the simulation model. As observed in this table the average error percentage,  $E_{avg}$ , does not exceed 6% in any of the instances. Utilization graphs for selected instances of Case 2 are presented in Appendix IV.

Computational results presented in Case 1 and Case 2 show that when the underlying distributions of  $a(t)$  and  $b(t)$  are identical to those in the corresponding simulation model, the results that we obtain through the Markov chain framework are quite close to those obtained via the simulation model, as expected. These two cases are developed to validate the simulation model since in both of these cases the results obtained via the Markov model contain no approximation. The results in Tables 6.2.3 and 6.3.3 show that the  $E_{avg}$  for Case 1 and Case 2 does not exceed 5.5% and on average it is 2.5%. We now discuss cases in which either  $a(t)$  or  $b(t)$  or both are non-exponential distributions, so as to observe the effectiveness of the proposed framework when the fitting methods are applied.

## 6.4 Case 3 - Toolset with Lognormal Underlying Distributions

In this case we assume that  $a(t)$  and  $b(t)$  are both lognormal *pdf's* but all other aspects of the system are the same as Case 1. We denote the mean of  $a(t)$  by  $\alpha$  and its variance by  $\sigma^2$ . The mean and variance of  $b(t)$  are denoted by  $\beta$  and  $\delta^2$ , respectively. We consider three categories of lognormal distributions with distinct values of the squared coefficient of variation and distinct values for the parameters  $\alpha$ ,  $\sigma^2$ ,  $\beta$  and  $\delta^2$ , as presented in Table 6.4.1.

Table 6.4.1: Parameters of the Lognormal  $a(t)$  and  $b(t)$

Inter-arrival Time Parameters									Processing Time Parameters		
$\rho = 70\%$			$\rho = 80\%$			$\rho = 90\%$			$\beta$	$\delta^2$	$SCV$
$\alpha$	$\sigma^2$	$SCV$	$\alpha$	$\sigma^2$	$SCV$	$\alpha$	$\sigma^2$	$SCV$			
1.0	1.0	1.0	0.9	0.7	1.0	0.8	0.6	1.0	1	1	1.0
1.0	0.5	0.5	0.9	0.4	0.5	0.8	0.3	0.5	1	0.5	0.5
1.0	0.1	0.1	0.9	0.1	0.1	0.8	0.1	0.1	1	0.1	0.1

To construct the Markov chain model for each instance we approximate the corresponding lognormal distributions by the fitting methods of Exponential Fit and Erlang Fit as discussed in Chapter 4. For the Exponential Fit method we simply use the mean of the lognormal distribution as the

mean of the exponential distribution. But for the Erlang Fit we choose the appropriate shape parameter  $k$ , for the Erlang  $(k, \lambda)$  distribution with the closest  $SCV$  to the corresponding lognormal distribution. Then we choose the rate parameter  $\lambda$  such that the mean of the Erlang distribution ( $\frac{k}{\lambda}$ ) be equal to the mean of the lognormal distribution.

The  $E_{avg}$  for the Exponential Fit and Erlang Fit are presented in Tables 6.4.2 and 6.4.3, respectively. In Table 6.4.2 that pertains to the Exponential Fit we observe that the accuracy of cycle time approximation is relatively low (*i.e.*, the value of  $E_{avg}$  is relatively high) compared with Case 1. We believe this deviation is due to the fact that we approximate a non-exponential distribution with a single phase of exponential distribution in each instance, through the Exponential Fit method. The utilization graphs for selected instances of this case are presented in Appendix IV.

Table 6.4.2:  $E_{avg}$  for Lognormal Distributions and Exponential Fit

Inter-arrival Time Lognormal Dist.	Processing Time Lognormal Dist.	Exponential Fit		
		No Rule	Rule I	Rule II
$SCV = 1$	$SCV = 1$	3.0%	14.4%	3.3%
$SCV = 0.5$	$SCV = 1$	21.1%	17.2%	14.4%
$SCV = 0.1$	$SCV = 1$	41.0%	19.7%	28.6%
$SCV = 1$	$SCV = 0.5$	9.2%	9.8%	7.0%
$SCV = 0.5$	$SCV = 0.5$	9.2%	12.6%	7.6%
$SCV = 0.1$	$SCV = 0.5$	30.6%	15.2%	24.0%
$SCV = 1$	$SCV = 0.1$	22.0%	16.9%	12.8%
$SCV = 0.5$	$SCV = 0.1$	3.3%	18.8%	3.8%
$SCV = 0.1$	$SCV = 0.1$	20.9%	11.5%	23.0%

Table 6.4.3:  $E_{avg}$  for Lognormal Distributions and Erlang Fit

Inter-arrival Time Lognormal Dist.	Processing Time Lognormal Dist.	Erlang Fit		
		No Rule	Rule I	Rule II
$SCV = 1$	$SCV = 1$	3.0%	14.4%	3.3%
$SCV = 0.5$	$SCV = 1$	1.4%	8.2%	5.7%
$SCV = 0.1$	$SCV = 1$	0.1%	3.6%	7.0%
$SCV = 1$	$SCV = 0.5$	5.0%	15.9%	3.7%
$SCV = 0.5$	$SCV = 0.5$	2.4%	9.1%	6.5%
$SCV = 0.1$	$SCV = 0.5$	0.4%	4.4%	7.5%
$SCV = 1$	$SCV = 0.1$	5.4%	16.8%	3.8%
$SCV = 0.5$	$SCV = 0.1$	2.0%	9.5%	6.7%
$SCV = 0.1$	$SCV = 0.1$	0.0%	4.4%	7.6%

Table 6.4.3 pertains to the instances where Erlang Fit is applied to the lognormal distributions of Table 6.4.1 and it shows the results of  $E_{avg}$  for each instance. Comparing Tables 6.4.2 and 6.4.3, we observe that applying the Erlang Fit method improves the accuracy of cycle time approximation over the Exponential Fit in the majority of instances. However the number of computational steps for the Erlang Fit is larger and the resulting Markov chain model has a larger number of states when the Erlang distribution is applied. The issue of the number of states in the Markov chain and the corresponding computational requirements of the method are further discussed in Section 6.9 at the end of this chapter. Figure 16 shows the simulation results on the utilization graph for the instance with lognormal underlying distributions of  $a(t)$  and  $b(t)$  with  $SCV = 0.1$  in both distributions, along with the Markov model results with both Erlang Fit and Exponential Fit. Selected utilization graph for the instances of this case are presented in Appendix IV.

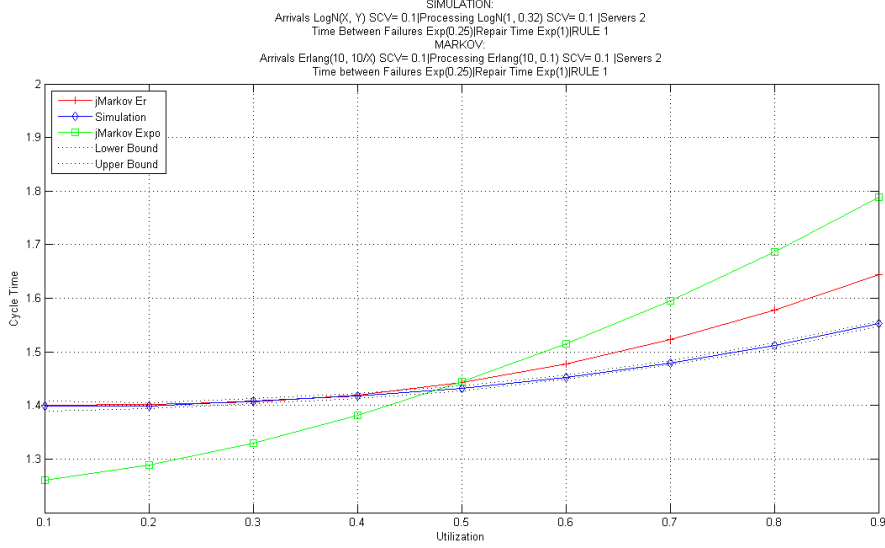


Figure 16: Utilization Graph of One Instance in Table 6.4.3

### 6.5 Case 4 - Toolset with Gamma Underlying Distributions

In this case we assume that  $a(t)$  and  $b(t)$  are both gamma probability density functions but all other aspects of the system are the same as Case 1. We denote the shape parameter of  $a(t)$  by  $\tau$  and its scale parameter by  $\theta$ . The mean of the gamma distribution is defined as  $\tau\theta$  and its variance as  $\tau\theta^2$ , hence its squared coefficient of variation is calculated as  $SCV = \frac{\tau\theta^2}{\tau^2\theta^2} = \frac{1}{\tau}$ . The shape and scale parameters of  $b(t)$  are also denoted by  $\gamma$  and  $\varphi$ , respectively and its  $SCV$  is equal to  $\frac{1}{\gamma}$ . We consider three sets of gamma distributions for  $a(t)$  and  $b(t)$  combinations with distinct values for the parameters  $\tau$ ,  $\theta$ ,  $\gamma$  and  $\varphi$ , as presented in Table 6.5.1.

Table 6.5.1: Parameters of the Gamma  $a(t)$  and  $b(t)$

Inter-arrival Time Parameters									Processing Time Parameters		
$\rho = 70\%$			$\rho = 80\%$			$\rho = 90\%$			$\gamma$	$\varphi$	$SCV$
$\tau$	$\theta$	$SCV$	$\tau$	$\theta$	$SCV$	$\tau$	$\theta$	$SCV$			
0.9	5.4	1.1	0.7	7.4	1.1	0.5	9.8	1.1	0.9	5.4	1.1
0.9	4.7	1.5	0.7	6.5	1.5	0.5	8.6	1.5	0.7	7.5	1.5
0.9	12.6	2	0.7	5.7	2	0.5	7.6	2	0.5	10	2

To construct the Markov chain model for each instance we approximate the corresponding gamma distributions by the fitting methods of Exponential Fit and Hyper-Erlang Fit as discussed in Chapter 4 and compare the results. In this case the Erlang Fit is not appropriate since the maximum  $SCV$  gained by this method is 1 and all the instances in this case have  $SCV$  higher than one.

For the Exponential Fit method we simply use the mean of the lognormal distribution as the mean of the exponential distribution. But for the Hyper-Erlang Fit we estimate the appropriate parameters of the distributions of  $a(t)$  and  $b(t)$ , namely  $\lambda_1, \lambda_2, \lambda_3, p_1, p_2$  and  $\mu_1, \mu_2, \mu_3, q_1$  and  $q_2$ . The estimation of these parameters is achieved through the algorithm proposed by [85] as discussed in Chapter 4. The estimated parameters of the Hyper-Erlang distributions fitted to the three gamma distributions of Table 6.5.1 are the same as those presented in Table 6.3.1 and Table 6.3.2 in the

context of Case 2, respectively. The resulting relative error values for Exponential Fit and Hyper-Erlang Fit are presented in Tables 6.5.2 and 6.5.3, respectively.

Table 6.5.2:  $E_{avg}$  for Gamma Distributions and Exponential Fit

Inter-arrival Time Lognormal Dist.	Processing Time Lognormal Dist.	Exponential Fit		
		No Rule	Rule I	Rule II
$SCV = 2$	$SCV = 2$	58.4%	37.9%	35.2%
$SCV = 1.5$	$SCV = 2$	71.9%	43.4%	41.6%
$SCV = 1.1$	$SCV = 2$	87.9%	49.1%	71.1%
$SCV = 2$	$SCV = 1.5$	24.4%	33.2%	38.7%
$SCV = 1.5$	$SCV = 1.5$	49.9%	40.2%	56.8%
$SCV = 1.1$	$SCV = 1.5$	22.6%	39.5%	39.2%
$SCV = 2$	$SCV = 1.1$	14.7%	35.6%	48.7%
$SCV = 1.5$	$SCV = 1.1$	26.1%	29.8%	36.9%
$SCV = 1.1$	$SCV = 1.1$	33.5%	32.7%	29.8%

Table 6.5.3:  $E_{avg}$  for Gamma Distributions and Hyper-Erlang Fit

Inter-arrival Time Lognormal Dist.	Processing Time Lognormal Dist.	Hyper-Erlang Fit		
		No Rule	Rule I	Rule II
$SCV = 2$	$SCV = 2$	13.7%	12.2%	11.0%
$SCV = 1.5$	$SCV = 2$	10.0%	13.6%	11.7%
$SCV = 1.1$	$SCV = 2$	11.2%	15.1%	13.7%
$SCV = 2$	$SCV = 1.5$	7.4%	14.6%	17.2%
$SCV = 1.5$	$SCV = 1.5$	7.7%	17.7%	17.9%
$SCV = 1.1$	$SCV = 1.5$	8.0%	19.6%	16.8%
$SCV = 2$	$SCV = 1.1$	18.4%	15.5%	14.1%
$SCV = 1.5$	$SCV = 1.1$	19.4%	18.3%	13.7%
$SCV = 1.1$	$SCV = 1.1$	21.5%	21.7%	15.2%

Comparing Tables 6.5.2 and 6.5.3, we observe that applying the Hyper-Erlang Fit results in higher accuracy of approximation compared with the Exponential Fit in all of the instances. However in the case of Hyper-Erlang Fit the number of states of the Markov chain model and the resulting computational requirements are much larger than the Exponential Fit method. This issue is further discussed in Section 6.9. Figure 17 shows the utilization graph for the instance with Rule II,  $a(t)$  gamma distribution with  $SCV = 1.1$  and  $b(t)$  gamma distribution with  $SCV = 2$  with both the Hyper-Erlang Fit and the Exponential Fit. The utilization graphs for selected instances of this case are presented in Appendix IV.

As it is observed in Case 3 and Case 4, when the underlying distributions of  $a(t)$  and  $b(t)$  are non-exponential and the fitting methods of Chapter 4 are applied the average relative error is larger than Case 1 and Case 2, where no approximation of underlying distributions is involved. However, the framework still provides reasonable accuracy in cycle time approximation. The overall average of  $E_{avg}$  in all instances in Case 3 and Case 4 with Erlang Fit or Hyper-Erlang Fit is 10.3% and the maximum value of  $E_{avg}$  is 21.7%.

The following two cases are developed for heterogeneous toolsets and toolsets with two types of failure. In both cases the underlying distributions of  $a(t)$  and  $b(t)$  are assumed to be Erlang and hence no approximation of underlying distributions is involved.

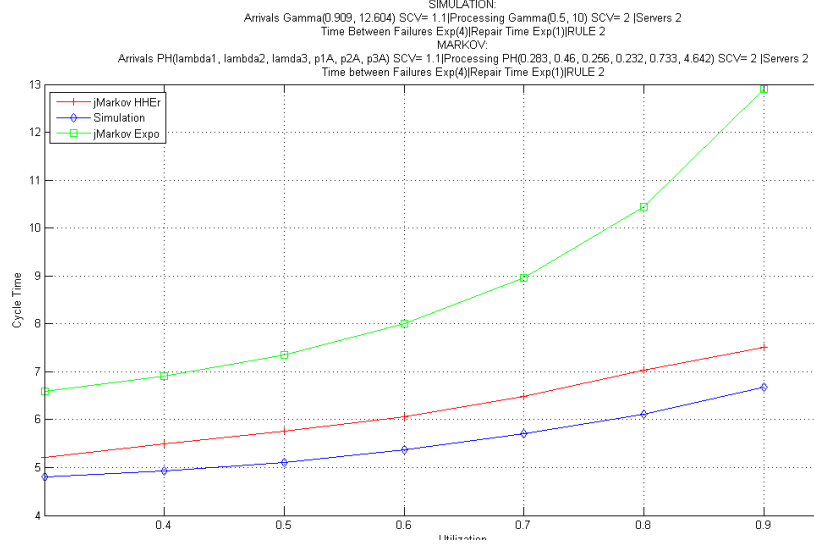


Figure 17: Utilization Graph of One Instance in Table 6.5.3

## 6.6 Case 5 - Toolset with Two Heterogeneous Tools

In this section we consider the toolset of Case 1 with the only change that in this toolset the processing rate of one of the tools is higher than the other by 25%. More specifically we assume that the underlying distribution of inter-arrival time is  $a(t)$  with Erlang distribution as shown in Table 6.2.1. The processing time distribution of tool 1 is denoted by  $b_1(t)$  with Erlang density function and rate  $\mu_1$ , and the processing time distribution of tool 2 is denoted by  $b_2(t)$  also with Erlang density function and rate  $\mu_2$ . The parameters of  $a(t)$ ,  $b_1(t)$  and  $b_2(t)$  for this toolset are presented in Table 6.6.1. The rest of the toolset characteristics remain unchanged with respect to Case 1.

Table 6.6.1: Parameters of  $a(t)$ ,  $b_1(t)$  and  $b_2(t)$  for Case 5

Case - 5		$\lambda$				
Distribution	SCV	70%	80%	90%	$\mu_1$	$\mu_2$
Erlang - 1	1	1.02	1.16	1.31	1	1.25
Erlang - 2	0.5	2.04	2.33	2.62	2	2.5
Erlang - 10	0.1	10.18	11.64	13.09	10	12.5

Since the underlying distributions are all Erlang or exponential with the memoryless property, there is no need to employ a fitting method and we can directly apply the rate parameters of  $a(t)$ ,  $b_1(t)$  and  $b_2(t)$  as the transition rates of the state-dependent Markov chain model presented in Chapter 5 for heterogeneous toolsets. Table 6.6.2 shows the  $E_{avg}$  for all the instances of this case. As we expect, the results of the Markov model are very close to those of the simulation in this case, since no approximation of underlying distributions is involved.

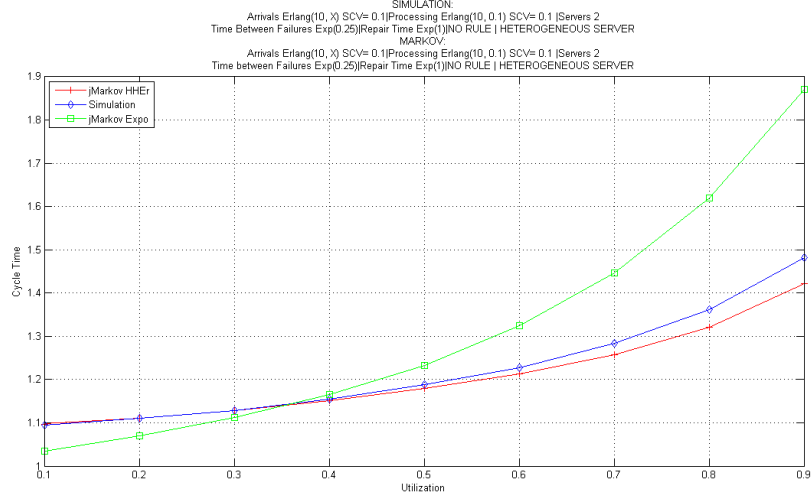


Figure 18: Utilization Graph of One Instance in Table 6.6.2

Table 6.6.2:  $E_{avg}$  for Heterogeneous Toolset Similar to Case 1

Inter-arrival Time Distribution	Processing Time Distribution	Rule Scenario		
		No Rule	Rule I	Rule II
<i>Erlang</i> – 1	<i>Erlang</i> – 1	0.5%	4.3%	0.6%
<i>Erlang</i> – 2	<i>Erlang</i> – 1	0.6%	3.6%	1.9%
<i>Erlang</i> – 10	<i>Erlang</i> – 1	0.2%	2.8%	2.5%
<i>Erlang</i> – 1	<i>Erlang</i> – 2	0.2%	4.2%	1.0%
<i>Erlang</i> – 2	<i>Erlang</i> – 2	0.5%	3.7%	1.8%
<i>Erlang</i> – 10	<i>Erlang</i> – 2	0.2%	2.9%	2.9%
<i>Erlang</i> – 1	<i>Erlang</i> – 10	0.4%	4.4%	0.9%
<i>Erlang</i> – 2	<i>Erlang</i> – 10	0.3%	3.8%	2.1%
<i>Erlang</i> – 10	<i>Erlang</i> – 10	0.3%	2.9%	3.1%

The utilization graph for one instance of Table 6.6.2 is shown in Figure 18 with Erlang underlying distributions of  $a(t)$  and  $b(t)$  and  $SCV = 0.1$  in both cases, with No Rule. The selected utilization graphs for some instances of this case are provided in Appendix IV.

## 6.7 Case 6 - Toolset with Two Failure Types

In this section we examine the performance of the Markov chain model on the toolset of Case 1 but we assume that the tools of this toolset are subject to two types of failure. The first failure type (Type I) has shorter expected time to failure with faster expected time to repair. For Type I failures we denote the *pdf* of the time to fail distribution by  $f_I(t)$  and expected time to fail  $\frac{1}{d_I}$ . The *pdf* of its corresponding time to repair is denoted with  $r_I(t)$  with expected time to repair  $\frac{1}{u_I}$ . This failure type represents short but frequent failures of the SMS tools that happen often but are fairly easy to fix. Type II failures have longer expected time to failure with longer expected time to repair. The *pdf* of the time to fail distribution for Type II failures is denoted by  $f_{II}(t)$  with expected value of  $\frac{1}{d_{II}}$ . The *pdf* of time to repair for a Type II failure is denoted by  $r_{II}(t)$  with expected time to repair  $\frac{1}{u_{II}}$ . This type of failure represents the major tool breakdowns that take a long time to be

repaired. The major breakdowns do not happen as often, however when they happen they usually require a long time to repair.

In this case we assume that  $f_I(t)$ ,  $f_{II}(t)$ ,  $r_I(t)$  and  $r_{II}(t)$  are all exponential with parameters of  $d_I = \frac{1}{4}$ ,  $d_{II} = \frac{1}{168}$ ,  $u_I = 1$  and  $u_{II} = \frac{1}{12}$ . In other words the Type *I* failure occurs on average every four hours and is fixed on average within an hour. But the Type *II* failure happens once a week and on average needs a whole shift or twelve hours to be repaired. The distributions and parameters of  $a(t)$  and  $b(t)$  for this case are the same as Case 1. The results are shown in Table 6.7.1.

Table 6.7.1:  $E_{avg}$  for Toolset with Two Failure Types

Inter-arrival Time Distribution	Processing Time Distribution	Rule Scenario		
		No Rule	Rule I	Rule II
<i>Erlang</i> – 1	<i>Erlang</i> – 1	1.3%	3.2%	1.2%
<i>Erlang</i> – 2	<i>Erlang</i> – 1	2.1%	2.6%	3.1%
<i>Erlang</i> – 10	<i>Erlang</i> – 1	1.1%	1.8%	2.9%
<i>Erlang</i> – 1	<i>Erlang</i> – 2	1.7%	1.3%	3.0%
<i>Erlang</i> – 2	<i>Erlang</i> – 2	2.0%	2.6%	1.5%
<i>Erlang</i> – 10	<i>Erlang</i> – 2	3.9%	2.7%	3.0%
<i>Erlang</i> – 1	<i>Erlang</i> – 10	2.4%	3.4%	2.1%
<i>Erlang</i> – 2	<i>Erlang</i> – 10	3.3%	2.9%	2.4%
<i>Erlang</i> – 10	<i>Erlang</i> – 10	1.3%	3.7%	1.5%

In Table 6.7.1 we observe that the results of the Markov model are very close to those of the simulation. This outcome conforms with our expectation since in this case all the underlying distributions are Erlang and hence no approximation is involved. Selected utilization graphs for this case are also provided in Appendix IV.

The objective of presenting Case 5 and Case 6 is to examine the performance of the extended Markov chain model for heterogeneous tools and for multiple failure types. In both cases the underlying distributions are considered to be Erlang, hence in these two cases no fitting method is applied. The overall average of  $E_{avg}$  in both cases is 2.1% and the maximum  $E_{avg}$  over all instances of these two cases is 4.4%. As the results suggest the average relative error is small in these cases when the underlying distributions are Erlang.

## 6.8 Comparison with $G/G/m$ Approximations

In this section we make a comparison between the accuracy of the proposed Markov chain framework and two of the  $G/G/m$  approximations commonly used in manufacturing systems. As discussed in Chapter 2 the classical queueing models have the inherent assumption of independence between the arrival and service processes and hence are incapable of modeling the operational rules. Therefore in this section the comparison is only made in the absence of operational rules.

We examine two of the  $G/G/m$  models discussed in Chapter 2, namely Equation (5) by Hopp and Spearman [10] and Equation (6) by Buzacott and Shanthikumar [7]. We apply these two approximations to the system where the inter-arrival time distribution  $a(t)$  and the processing time distribution  $b(t)$  are both Erlang and where they are Hyper-Erlang with the same parameters as presented in Case 1 and Case 2, respectively. We also do the same comparison with non-exponential underlying distributions of Case 3 and Case 4. For each model, *i.e.*, the Markov chain framework, Hopp-Spearman formula (H&S) and Buzacott-Shanthikumar formula (B&S) we compare the cycle time approximation to the cycle time of the simulation model with no operational rule and calculate the average error percentage,  $E_{avg}$ . The results are presented in Tables 6.8.1 - 6.8.6.

Table 6.8.1: Comparison with  $G/G/m$  Approximations - Case 1

$a(t)$ Erlang Distribution	$b(t)$ Erlang Distribution	Markov Model	$G/G/m$ H&S	$G/G/m$ B&S
$SCV = 1$	$SCV = 1$	0.5%	30.2%	36.0%
$SCV = 0.5$	$SCV = 1$	0.8%	30.2%	38.5%
$SCV = 0.1$	$SCV = 1$	0.3%	28.3%	34.1%
$SCV = 1$	$SCV = 0.5$	0.5%	7.7%	3.6%
$SCV = 0.5$	$SCV = 0.5$	0.5%	7.8%	1.9%
$SCV = 0.1$	$SCV = 0.5$	0.5%	9.0%	5.5%
$SCV = 1$	$SCV = 0.1$	0.7%	33.6%	31.2%
$SCV = 0.5$	$SCV = 0.1$	0.5%	33.4%	29.1%
$SCV = 0.1$	$SCV = 0.1$	0.1%	36.8%	35.0%

Table 6.8.2: Comparison with  $G/G/m$  Approximations - Case 2

$a(t)$ Hyper-Erlang Dist	$b(t)$ Hyper-Erlang Dist	Markov Model	$G/G/m$ H&S	$G/G/m$ B&S
$SCV = 2$	$SCV = 2$	1.3%	47.0%	53.5%
$SCV = 1.5$	$SCV = 2$	3.8%	33.5%	41.9%
$SCV = 1.1$	$SCV = 2$	3.5%	21.9%	27.3%
$SCV = 2$	$SCV = 1.5$	3.2%	11.0%	15.4%
$SCV = 1.5$	$SCV = 1.5$	1.6%	2.7%	3.5%
$SCV = 1.1$	$SCV = 1.5$	3.4%	14.7%	11.4%
$SCV = 2$	$SCV = 1.1$	1.9%	14.8%	11.8%
$SCV = 1.5$	$SCV = 1.1$	3.1%	28.5%	24.0%
$SCV = 1.1$	$SCV = 1.1$	0.5%	40.8%	39.1%

As we observe in Tables 6.8.1 and 6.8.2 in the case with no operational rule and Erlang or Hyper-Erlang underlying distributions of  $a(t)$  and  $b(t)$  the proposed Markov chain model performs with higher accuracy than the two  $G/G/m$  approximations of [10] and [7]. But this is not remarkable since in this case all underlying assumptions of the Markov model are indeed satisfied.

Next we compare these  $G/G/m$  formulas with the Markov chain model for underlying distributions of Case 3 (lognormal with  $SCV \leq 1$ ) and Case 4 (gamma with  $SCV > 1$ ) under no operational rule and using Exponential Fit, Erlang Fit and Hyper-Erlang Fit as appropriate. The results are presented in Tables 6.8.3 and 6.8.4, respectively.

Table 6.8.3: Comparison with  $G/G/m$  Approximations - Case 3

$a(t)$ LogN Dist.	$b(t)$ LogN Dist.	Markov Expo Fit	Markov Erlang Fit	$G/G/m$ H&S	$G/G/m$ B&S
$SCV = 1$	$SCV = 1$	3.0%	3.0%	34.4%	40.3%
$SCV = 0.5$	$SCV = 1$	21.1%	1.4%	32.1%	40.6%
$SCV = 0.1$	$SCV = 1$	41.0%	0.1%	29.2%	35.0%
$SCV = 1$	$SCV = 0.5$	9.2%	5.0%	12.8%	12.2%
$SCV = 0.5$	$SCV = 0.5$	9.2%	2.4%	5.1%	0.9%
$SCV = 0.1$	$SCV = 0.5$	30.6%	0.4%	9.2%	5.7%
$SCV = 1$	$SCV = 0.1$	22.0%	5.4%	29.6%	27.1%
$SCV = 0.5$	$SCV = 0.1$	3.3%	2.0%	32.6%	28.4%
$SCV = 0.1$	$SCV = 0.1$	20.9%	0.0%	36.6%	34.7%

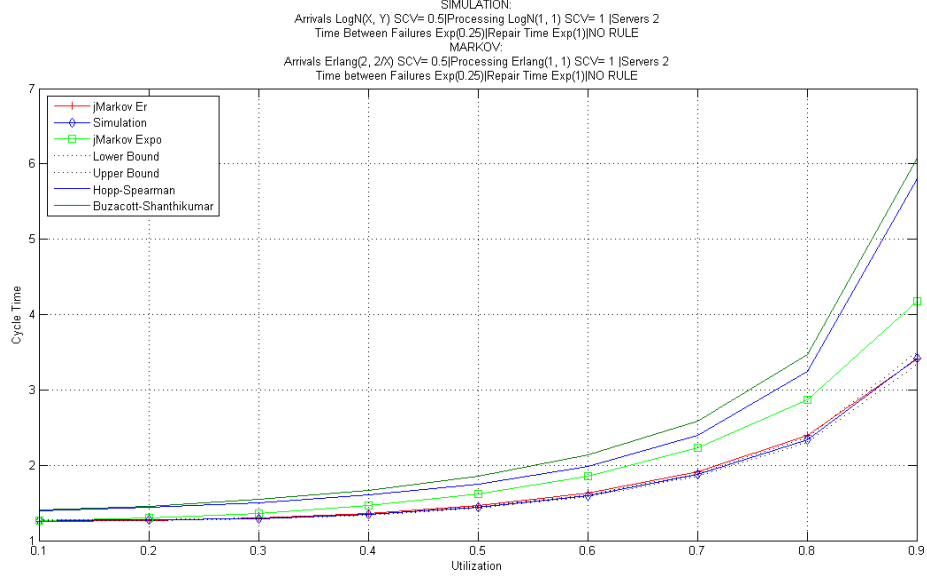


Figure 19: Utilization Graph of One Instance in Table 6.8.3

Table 6.8.4: Comparison with  $G/G/m$  Approximations - Case 4

$a(t)$ Gamma Dist	$b(t)$ Gamma Dist	Markov Expo Fit	Markov H-E Fit	$G/G/m$ H&S	$G/G/m$ B&S
$SCV = 2$	$SCV = 2$	58.4%	13.7%	47.0%	53.5%
$SCV = 1.5$	$SCV = 2$	71.9%	10.0%	33.5%	41.9%
$SCV = 1.1$	$SCV = 2$	87.9%	11.2%	21.9%	27.3%
$SCV = 2$	$SCV = 1.5$	24.4%	7.4%	11.0%	15.4%
$SCV = 1.5$	$SCV = 1.5$	49.9%	7.7%	2.7%	3.5%
$SCV = 1.1$	$SCV = 1.5$	22.6%	8.0%	14.7%	11.4%
$SCV = 2$	$SCV = 1.1$	14.7%	18.4%	18.8%	21.8%
$SCV = 1.5$	$SCV = 1.1$	26.1%	19.4%	28.5%	24.0%
$SCV = 1.1$	$SCV = 1.1$	33.5%	21.5%	40.8%	39.1%

In the case of approximating non-exponential underlying distributions of  $a(t)$  and  $b(t)$  with Exponential Fit we observe that the performance of the Markov model is not always better than the two  $G/G/m$  approximations. However when the Erlang Fit or Hyper-Erlang Fit is applied the accuracy of the Markov model becomes significantly higher in the majority of instances. Figure 19 shows the utilization graph for the instance in Table 6.8.3 with lognormal underlying distributions of  $a(t)$  and  $b(t)$  with  $SCV = 0.5$  and  $SCV = 1$ , respectively under the No Rule scenario. As it is demonstrated in the graph the Erlang Fit method gives a better approximation compared to H&S and B&S approximations.

In Tables 6.8.5 and 6.8.6 we compare the performance of the Markov model with the  $G/G/m$  approximations of H&S and B&S, respectively for Case 5 with heterogeneous tools and for Case 6 with two types of failure both with Erlang underlying distributions. Again the results show that the Markov model outperforms the classical  $G/G/m$  approximations in all these instances.

Table 6.8.5: Comparison with  $G/G/m$  Approximations - Case 5

$a(t)$ Erlang Dist	$b(t)$ Erlang Dist	Markov Model	$G/G/m$ H&S	$G/G/m$ B&S
$SCV = 1$	$SCV = 1$	0.5%	24.5%	39.6%
$SCV = 0.5$	$SCV = 1$	0.6%	23.3%	39.8%
$SCV = 0.1$	$SCV = 1$	0.2%	28.6%	32.6%
$SCV = 1$	$SCV = 0.5$	0.2%	18.2%	22.3%
$SCV = 0.5$	$SCV = 0.5$	0.5%	35.1%	16.9%
$SCV = 0.1$	$SCV = 0.5$	0.2%	19.2%	25.7%
$SCV = 1$	$SCV = 0.1$	0.4%	32.4%	23.9%
$SCV = 0.5$	$SCV = 0.1$	0.3%	33.6%	28.3%
$SCV = 0.1$	$SCV = 0.1$	0.3%	32.4%	41.7%

Table 6.8.6: Comparison with  $G/G/m$  Approximations - Case 6

$a(t)$ Erlang Dist	$b(t)$ Erlang Dist	Markov Model	$G/G/m$ H&S	$G/G/m$ B&S
$SCV = 1$	$SCV = 1$	1.3%	68.0%	73.5%
$SCV = 0.5$	$SCV = 1$	2.1%	55.5%	69.7%
$SCV = 0.1$	$SCV = 1$	1.1%	83.2%	87.3%
$SCV = 1$	$SCV = 0.5$	1.7%	51.0%	65.9%
$SCV = 0.5$	$SCV = 0.5$	2.0%	83.2%	63.5%
$SCV = 0.1$	$SCV = 0.5$	3.9%	76.3%	61.4%
$SCV = 1$	$SCV = 0.1$	2.4%	45.8%	39.6%
$SCV = 0.5$	$SCV = 0.1$	3.3%	48.5%	74.0%
$SCV = 0.1$	$SCV = 0.1$	1.3%	56.0%	49.1%

Utilization graphs for selected instances of Case 1 through Case 6 in comparison with the  $G/G/m$  approximations of H&S and B&S are presented in Appendix IV.

## 6.9 Computational Requirements

In this section we compare the computational requirements of the Markov model with those of the simulation under two different scenarios for the underlying distributions of Case 1 and Case 2. To this end we record the running time of both models in different combinations of  $SCV$  for inter-arrival and processing times for Case 1 and Case 2 underlying distributions and at different levels of utilization.

Both models are run with the same processing power (Intel CoreDuo processor, 3.0 GHz). In Tables 6.9.1 and 6.9.2 we present the execution time of the Markov model in milliseconds ( $MCT$ ) for the underlying distributions of each case (*i.e.*, Erlang and Hyper-Erlang Distributions), as well as the ratio of simulation running time to the running time of the Markov model ( $S/M$ ). We also present the number of states in the Markov chain model ( $N$ ) for each case. Note that in the case of Hyper-Erlang distributions, presented in Table 6.9.2 the number of states does not depend on the  $SCV$  since in all instances the same mixture is applied and that  $SCV$  is only changed through changing the mixture parameters.

Table 6.9.1: Computational Requirements of Case 1

$a(t)$	$SCV = 1$			$SCV = 1$			$SCV = 1$		
$b(t)$	$SCV = 1$			$SCV = 0.5$			$SCV = 0.1$		
$\rho$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$
70%	15	101	180	35	53	535	140	10	7063
80%	15	116	180	35	56	535	172	13	7063
90%	16	119	180	36	58	535	203	10	7063
$a(t)$	$SCV = 0.5$			$SCV = 0.5$			$SCV = 0.5$		
$b(t)$	$SCV = 1$			$SCV = 0.5$			$SCV = 0.1$		
$\rho$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$
70%	30	98	478	47	35	1061	657	2	14005
80%	31	55	478	48	57	1061	703	3	14005
90%	31	117	478	50	40	1061	718	3	14005
$a(t)$	$SCV = 0.1$			$SCV = 0.1$			$SCV = 0.1$		
$b(t)$	$SCV = 1$			$SCV = 0.5$			$SCV = 0.1$		
$\rho$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$
70%	1390	1.1	2374	7469	0.2	5269	684406	0	69541
80%	1391	1.2	2374	7312	0.2	5269	684719	0	69541
90%	1422	1.3	2374	7703	0.2	5269	681531	0	69541

Table 6.9.2: Computational Requirements of Case 2

$a(t)$	$SCV = 2$			$SCV = 2$			$SCV = 2$		
$b(t)$	$SCV = 2$			$SCV = 1.5$			$SCV = 1.1$		
$\rho$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$
70%	86	21	2880	83	22	2880	79	28	2880
80%	89	24	2880	88	29	2880	78	30	2880
90%	91	24	2880	87	32	2880	86	25	2880
$a(t)$	$SCV = 1.5$			$SCV = 1.5$			$SCV = 1.5$		
$b(t)$	$SCV = 2$			$SCV = 1.5$			$SCV = 1.1$		
$\rho$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$
70%	93	19	2880	71	29	2880	86	33	2880
80%	89	26	2880	80	24	2880	99	30	2880
90%	87	23	2880	67	21	2880	85	22	2880
$a(t)$	$SCV = 1.1$			$SCV = 1.1$			$SCV = 1.1$		
$b(t)$	$SCV = 2$			$SCV = 1.5$			$SCV = 1.1$		
$\rho$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$	$MCT$	$S/M$	$N$
70%	85	41	2880	86	17	2880	77	30	2880
80%	89	36	2880	92	21	2880	79	28	2880
90%	87	36	2880	89	22	2880	81	23	2880

In Table 6.9.1 we observe that the computational requirements of the Markov model generally increase as the number of states grows. For instance in combinations when the  $SCV$  of both distributions are low and hence the number of Erlang phases is higher the number of states grows significantly and causes the computation time of the Markov model to increase to the degree that in some instances it becomes less efficient compared to the simulation model. However, in Table 6.9.2 due to applying the Hyper-Erlang mixture with fixed number of states (*i.e.*, three branches with a total of six states) the total number of states does not change with changing the  $SCV$ . In this table we observe that in all instances the Markov model outperforms the simulation.

## 7 Concluding Remarks

In this chapter we first present a summary of the completed work in this dissertation and then discuss potential directions to continue and extend this work.

### 7.1 Summary of Completed Work

In this research we addressed the problem of cycle time approximation for semiconductor manufacturing toolsets with operational rules. In Chapters 1 and 2 we gave an overview of the semiconductor manufacturing systems (SMS) and reviewed two common approaches to cycle time approximation of SMS toolsets, namely simulation and queueing theory [3]. We discussed the inefficiencies of simulation models to provide fast response to what-if questions and quick decision making needs in a fast-pace industry such as SMS. Simulation models also require extensive amounts of input data and are difficult to maintain.

Queueing theory, on the other hand, provides fast results and requires relatively less input data and maintenance. However the performance of current queueing models in general and  $G/G/m$  approximations in particular have not been satisfactory [3]. The accuracy of cycle time approximation by such models is far from the reality of SMS due to different reasons.

We identified that one of the issues that complicates the modeling of SMS toolsets through queueing approximations is the existence of unwritten and informal operational rules in real SMS fabrication facilities (fabs). These rules are often put in place by the line managers to spontaneously control and smoothen the flow of the fab. Application of operational rules creates dependency between the arrival and service processes and hence makes it impossible for the existing approximate  $G/G/m$  formulas to give an accurate approximation of cycle time. This is due to the fact that such approximations have the inherent assumption of independence between the arrival and service processes.

In Chapter 3 we considered two types of operational rules that are commonly used in SMS fabs, namely arrival control (*i.e.*, Rule *I*) and repair control (*i.e.*, Rule *II*) that adjust the arrival rate or the repair rate, respectively, based on the WIP level of the toolset. For the cycle time approximation of a toolset with operational rules we proposed a continuous-time Markov chain model with state-dependent transition rates. The state-space of this Markov chain was defined by two variables that indicated the WIP level and the number of active tools at any time. This approach to modeling the system allowed us to include the two types of operational rules mentioned above. Since at each state the WIP level and the number of active tools were known, the arrival rate and the failure rate could be adjusted accordingly.

Through solving the balance equations we found the steady state probabilities for this Markov chain. Given these probabilities we calculated the time-weighted average WIP level in the system. Finally using Little's law and the average arrival rate we approximated the long-run average cycle time of the toolset. To conclude this chapter we discussed the steps to develop a simulation model of SMS toolsets, including the operational rules in order to evaluate the performance of the proposed model through comparing its results with the simulation outcome.

In Chapter 4 we extended the basic Markov chain model of Chapter 3 to include toolsets with non-exponential underlying distributions. Due to the Markovian property of the model presented in Chapter 3 we had to assume the inter-arrival, processing, failure and repair times to be exponential random variables. However this assumption was far from the reality of SMS in many cases and could limit the performance of the proposed model. To overcome this limitation we approximated the non-exponential distributions by an exponential distribution or a mixture of phases of exponential distributions which maintained the memoryless property. We proposed three distribution fitting methods of *Exponential Fit*, *Erlang Fit* and *Hyper-Erlang Fit*.

To include the approximated distributions of inter-arrival and processing times by these fitting

methods in the Markov chain model we proposed extensions to the state space of the basic model in Chapter 3 for the fitting methods and added auxiliary state variables to keep track of the phases of the mixture distributions of the arrival and service processes.

We then introduced the Quasi Birth-Death (QBD) processes and their proposed solution algorithms to find the steady-state probabilities of a Markov chain with certain characteristics. Then we demonstrated that the QBD approach can be applied to the extended Markov chain model of Chapter 4 to solve the models of complex toolsets with approximated distributions through the Exponential Fit, Erlang Fit or Hyper-Erlang Fit methods based on the QBD formulation which provided efficiency in the computational time.

In Chapter 5 we discussed extensions to the basic Markov chain model of Chapter 3 to include toolsets with heterogeneous tools and toolsets with multiple types of failure. Through examples we demonstrated the extended state-space of the basic Markov chain model to include such complexities as well. The basic model of Chapter 3 together with the extensions of Chapter 4 and Chapter 5 along with the QBD solution method made a framework for cycle time approximation of complex toolsets under operational rules.

In Chapter 6 we examined the performance of the cycle time approximation framework through a series of case studies with different features. We first introduced a method to evaluate the performance of the proposed cycle time approximation framework with respect to simulation. To this end we defined the average error percentage ( $E_{avg}$ ) as our evaluation measure. This measure first compares the performance of the proposed model with simulation in three levels of high utilization (70%; 80% and 90%) and then takes the average across these three points. Then we applied the basic Markov model of Chapter 3 and its extensions introduced in Chapter 4 to approximate the cycle time of a toolset with two parallel tools and three scenarios regarding the operational rules, namely No Rule, Rule *I* and Rule *II*.

In Case 1 we assumed that the underlying distributions of inter-arrival times and processing times are Erlang with the Markovian property and in Case 2 we assumed that the underlying distributions are Hyper-Erlang again retaining the memoryless property. Hence in both of these cases no approximation to the underlying distributions was involved and therefore, we expected the results to be very close to those obtained via the corresponding simulation models. These two cases were presented to validate the correctness and accuracy of the proposed Markov chain model and we observed that the average error percentage in these two cases were at most 6%.

In Case 3 and Case 4 we applied the model to toolsets with lognormal and gamma underlying distributions with squared coefficients of variation ( $SCV$ ) less than one and greater than one, respectively. In Case 3 we employed Exponential Fit and Erlang Fit to approximate the lognormal distributions and in Case 4 we applied Exponential Fit and Hyper-Erlang Fit since the Erlang Fit was not feasible for the distributions with  $SCV \geq 1$ . We observed that the Exponential Fit although faster and easier to apply did not provide accurate approximation compared with the Erlang Fit and Hyper-Erlang Fit methods which had higher computational requirements.

In Case 5 we applied the cycle time approximation framework to a toolset with heterogeneous tools and non-exponential underlying distributions and in Case 6 we applied the framework to approximate the cycle time of a toolset with two types of failure and non-exponential underlying distributions.

In the above mentioned cases for all instances where no operational rule was applied we also compared the performance of the proposed cycle time approximation framework with the two commonly used  $G/G/m$  approximations in SMS. The result of this comparison showed that in cases of Erlang and Hyper-Erlang underlying distributions the proposed framework performed significantly better than the  $G/G/m$  approximations. Also when either of the Erlang Fit or Hyper-Erlang Fit methods were employed to approximate the lognormal or gamma underlying distributions the Markov model outperformed the two  $G/G/m$  approximations. However when the Exponential Fit was applied the results were not necessarily better.

Finally we studied the computational requirements of the proposed Markov model and its extensions by comparing their solution time to the solution time of the simulation model. In the case of Erlang Fit we observed that in general as the number of states of the Markov model grew the efficiency of the proposed model in solution time decreased sharply. However, in the Hyper-Erlang Fit method since the number of states were constant the computational efficiency did not depend on the size of the Markov chain and in all of the instances the Markov model outperformed the simulation model in computational time.

## 7.2 Direction for Future Research

The proposed Markov chain model of Chapter 3 and its extensions in Chapter 4 are developed for a single toolset in SMS. In a high volume manufacturing fab in SMS a number of toolsets are connected with each other through the process flow of the products and usually form a job shop. Using the proposed Markov chain framework for cycle time approximation and expanding the state space to include more toolsets we can approximate the total cycle time of a fab in SMS. This expansion of the state-space however, adds to the computational complexity of the problem and more efficient solution methods for the steady-state equations are to be investigated.

For example Kempf [82] proposes a simplified model of a "Mini-FAB" with three toolsets and six process steps for academic modeling and analysis purposes. Applying the proposed Markov chain framework to model each operation at this network as part of the state-space and using the QBD algorithms to find the steady-state probabilities, the total cycle time of the Mini-FAB can be approximated.

The queueing network model for the Mini-FAB can also serve as a platform to experiment the effectiveness of various operational rules and make improved decisions regarding the parameter of each rule.

The second direction that we propose is research on efficient methods of finding appropriate mixtures of exponential phases for different probability density functions or dataset with minimum number of phases and accurate approximations. In Chapter 4 we propose three fitting methods to approximate a non-exponential distribution with one or more phases of exponential distribution. Although the proposed fitting methods can provide accurate results for most of the instances in the experimented cases of Chapter 6 there are instances in which the accuracy of approximation can be improved by employing more efficient fitting methods.

## References

- [1] R. Leachman and S. Ding (2007). "Integration of speed economies into decision-making for manufacturing management," *International Journal of Production Economics*, 107(1), 39-55.
- [2] R. Leachman, S. Ding and C. Chien (2007). "Economic Efficiency Analysis of Wafer Fabrication," *IEEE Transactions on Automation Science and Engineering*, 4(4), 501-512.
- [3] J. G. Shanthikumar, S. Ding and M. Zhang (2007). "Queueing theory for semiconductor manufacturing systems: a survey and open problems," *IEEE Transactions on Automation Science and Engineering*, 4(4).
- [4] Kan Wu, Leon F. McGinnis and Bert Zwart (2007). "Compatibility of queueing theory, manufacturing systems and SEMI standards", *Proceedings of The 3rd Annual IEEE Conference on Automation Science and Engineering*, Scottsdale, AZ, USA.
- [5] H. Chen, J. M. Harrison, A. Mandelbaum, A. V. Ackere and L. M. Wein (1988). "Empirical evaluation of a queueing network model for semiconductor wafer fabrication," *Operations Research*, 36(2), 202-215.
- [6] Peter Van Zant, *Microchip Fabrication*, 5th Edition, McGraw-Hill, 2004.
- [7] J. A. Buzacott and J. G. Shanthikumar, *Stochastic Models of Manufacturing Systems*, Prentice Hall, Englewood Cliffs, NJ, 1993.
- [8] D. Connors, G. Feigin and D. Yao (1996). "A queueing network model for semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, 9.
- [9] W. Whitt (1983). "The queueing network analyzer," *Bell Systems Technology Journal*, 62, 2779-2815.
- [10] W. J. Hopp and M. L. Spearman, *Factory Physics: Foundations of Manufacturing Management*, London, U.K.: Irwin McGraw-Hill, 2001.
- [11] R. Morrison and D. P. Martin (2007). "Practical Extensions to Cycle Time Approximations for the G/G/m-Queue with Applications", *IEEE Transactions on Automation Science and Engineering*, 4(4).
- [12] A. Aissani and J. R. Artalejo (1998). "On the single server retrial queue subject to breakdowns," *Queueing Systems*, 30(3-4), 309-321.
- [13] I. Mitrani and B. Avi-Itzhak (1968). "A many server queue with service interruptions," *Operations Research*, 16(3), 628-638.
- [14] I. Mitrani and A. Puhalskii (1993). "Limiting results for multiprocessor systems with breakdowns and repairs," *Queueing Systems Theory and Applications*, 14, 293-311.
- [15] J. H. Jacobs, L. F. P. Etman, E. J. J. van Campen and J. E. Rooda (2003). "Characterization of operational time variability using effective process times," *IEEE Transactions on Semiconductor Manufacturing*, 16(3), 511-520.
- [16] J. H. Jacobs, P.P. van Bakel, L. F. P. Etman and J. E. Rooda (2006). "Qualifying variability of batching equipment using effective process times," *IEEE Transactions on Semiconductor Manufacturing*, 19(2), 269-275.

- [17] M. G. Huang, P.L. Chang and Y. C. Chou (2001). "Analytical approximations for multi-server batch-service workstations with multiple process recipes in semiconductor wafer fabrication", *IEEE Transactions on Semiconductor Manufacturing*, 14(4).
- [18] E. S. Gel, J. W. Fowler, and K. Khawalaz, "Queueing Approximations for Capacity Planning under Common Setup Rules", *working paper*, Department of Industrial Engineering, Arizona State University.
- [19] A. Raddon and B. Grigsby (1997). "Throughput time forecasting model," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 430-433.
- [20] C. F. Chien, C. W. Hsiao, C. Meng, K. T. Hong and S. T. Wang (2005). "Cycle time prediction and control based on production line status and manufacturing data mining," *Proceedings of International Symposium on Semiconductor Manufacturing*, 327-330.
- [21] P. Backus, M. Janakiram, S. Mowzoon, G. C. Runger and A. Bhargava (2006). "Factory cycle-time prediction with a data-mining approach," *IEEE Transactions on Semiconductor Manufacturing*, 19(2).
- [22] Y. Hung and C. Chang (1999). "Using an empirical queueing approach to predict future flow times," *Computers and Industrial Engineering*, 37, 809-821.
- [23] S. Park, J. W. Fowler, G. T. Mackulak, J. B. Keats and W. M. Carlyle (2002). "D-Optimal sequential experiments for generating a simulation-based cycle time-throughput curve," *Operations Research*, 50(6), 981-990.
- [24] D. P. Martin (1997). "How the law of unanticipated consequences can nullify the theory of constraints: the case for balanced capacity in a semiconductor manufacturing line," *Proceedings of IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 380-385.
- [25] J. R. Jackson (1963). "Job-shop-like queueing systems," *Management Science*, 10, 131-142.
- [26] P.J. Kuehn (1978). "Approximate analysis of general queueing networks by decomposition," *IEEE Transactions on Communications*, 27, 113-126.
- [27] J.G. Shanthikumar and J. A. Buzacott (1981). "Open queueing network models of dynamic job shops," *International Journal of Production Research*, 19, 255-266.
- [28] J. G. Shanthikumar and J. A. Buzacott (1984). "The time spent in a dynamic job shop," *European Journal of Operations Research*, 17, 215-226.
- [29] H. Chen, J. M. Harrison, A. Mandelbaum, A. V. Ackere and L. M. Wein (1988). "Empirical evaluation of a queueing network model for semiconductor wafer fabrication," *Operations Research*, 36(2), 202-215.
- [30] W. J. Hopp, M. Spearman, S. Chayet, K. Donohue and E. Gel (2002). "Using an optimized queueing network model to support wafer fab design," *IIE Transactions*, 34, 119-130.
- [31] G. Meng and S. S. Heragu (2002). "Batch size modeling in a multi-item, discrete manufacturing system via an open queueing network", *IIE Transactions*, 36, 743-753.
- [32] G. L. Curry and B. L. Deurmeyer (2002). "Renewal approximations for the departure processes of batch systems," *IIE Transactions*, 34(2), 94-104.
- [33] M. Hu and S. Chang (2003). "Translating overall production goals into distributed flow control parameters for semiconductor manufacturing," *Journal of Manufacturing Systems*, 22(1), 46-63.

- [34] J. Miltenburg, C. H. Cheng and H. Yan (2002). "Analysis of wafer fabrication facilities using four variations of the open queueing network decomposition model," *IIE Transactions*, 34(3), 263-272.
- [35] G. R. Bitran and D. Tirupati (1988). "Multiproduct queueing networks with deterministic routing: decomposition approach and the notion of interference," *Management Science*, 34(1), 75-100.
- [36] S. Kim (2005). "Approximation of multiclass queueing networks with highly variable arrivals under deterministic routing," *Naval Research Logistics*, 52(5), 399-408.
- [37] W. Whitt (1995). "Variability functions for parametric-decomposition approximations of queueing networks," *Management Science*, 41, 1704-1715.
- [38] G. R. Bitran and S. Dasu (1992). "A review of open queueing network models of manufacturing systems," *Queueing Systems*, 12, 95-134.
- [39] H. Chen and D. D. Yao (1993). "Dynamic scheduling of a multiclass fluid network," *Operations Research*, 41(6), 1104-1115.
- [40] J. G. Dai (1995). "On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models," *the annals of applied probability*, 5(1), 49-77.
- [41] D. Connors, G. Feigin and D. Yao (1994). "Scheduling semiconductor lines using a fluid network model," *IEEE Transactions on Robotics and Automation*, 10(2), 88-98.
- [42] S. Kumar and P. R. Kumar (2001). "Queueing network models in the design and analysis of semiconductor wafer fabs," *IEEE Transactions on Robotics and Automation*, 17(5), 548-561.
- [43] S. B. Gershwin (1999), "Design and operation of manufacturing systems – the control-point policy," *IIE Transactions*, 2(2), 93-103.
- [44] J. Li (2005). "Overlapping decomposition: a system-theoretic method for modeling and analysis of complex manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, 2(1), 40-53.
- [45] M. I. Reiman (1984). "Open queueing networks in heavy traffic," *Mathematics of Operations Research*, 9, 441-458.
- [46] J. M. Harrison and R. J. Williams (1987). "Brownian models of open queueing networks with homogeneous customer populations," *Stochastics*, 22, 77-115.
- [47] J. M. Harrison and M. I. Reiman (1981). "Reflected Brownian motion on an orthant," *The Annals of Probability*, 9, 302-308.
- [48] J. M. Harrison and V. Nguyen (1990). "The QNET method for two moment analysis of open queueing networks," *Queueing Systems*, 6, 1-32.
- [49] J. G. Dai and J. M. Harrison (1992). "Reflected Brownian motion in an orthant: numerical methods for steady-state analysis", *Annals of Applied Probability*, 2, 65-86.
- [50] J. M. Harrison and V. Nguyen, "Brownian models of multiclass queueing networks: current status and open problems," *Queueing Systems*, 13, 5-40.
- [51] J. M. Harrison and M. T. Pich (1997). "Two-moment analysis of open queueing networks with general workstation capabilities", *Operations Research*, 45(4), 610-623.

- [52] J. G. Dai, D. H. Yeh and C. Zhou (1997). "The Q-NET method for re-entrant queueing networks with priority disciplines," *Operations Research*, 45(4), 610-623.
- [53] H. Chen, X. Shen and D. D. Yao (2002). "Brownian Approximations of Multiclass Open-Queueing Networks", *Operations Research*, 50(6), 1032-1049.
- [54] E. Gelenbe and L. Mitrani, *Analysis and Synthesis of Computer Systems*. Academic, London, 2010.
- [55] J. G. Shanthikumar and D. D. Yao (1992). "Multiclass queueing systems: polymatroidal structure and optimal scheduling control," *Operations Research*, 40(2).
- [56] D. Bertsimas, I. C. Paschalidis and J. N. Tsitsiklis (1995), "Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance," *Annals of Applied Probability*, 4, 43-75.
- [57] P. R. Kumar (1993). "Re-entrant lines," *Queueing Systems*, 13, 87-110.
- [58] P. R. Kumar and S. P. Meyn (1995). "Stability of queueing networks and scheduling policies," *IEEE Transactions on Automatic Control*, 40(2), 251-260.
- [59] S. Kumar and P. R. Kumar (1994). "Performance bounds for queueing networks and scheduling policies," *IEEE Transactions on Automatic Control*, 39(8), 1600-1611.
- [60] J. M. Harrison and L. M. Wein (1990). "Scheduling networks of queues: Heavy traffic analysis of a two-station closed network," *Operations Research*, 38(6), 1052-1064.
- [61] H. Jin, J. Ou and P. R. Kumar (1997). "The throughput of irreducible closed Markovian queueing networks: Functional bounds, asymptotic loss, efficiency and the Harrison-Wein conjectures," *Mathematics of Operations Research*, 22(4), 886-920.
- [62] J. Miltenburg, C. H. Cheng and H. Yan (2002). "Analysis of wafer fabrication facilities using four variations of the open queueing network decomposition model," *IIE Transactions*, 34(3), 263-272.
- [63] Jerry Banks, *Handbook of Simulation*, Engineering and Management Press, 1998.
- [64] D.R. Cox (1955). "A use of complex probabilities in the theory of stochastic processes," *Proceedings of Cambridge Philosophical Society*, 51.
- [65] Marcel F. Neuts, *Matrix-Geometric Solutions In Stochastic Models: An Algorithmic Approach*, Dover Publications, NY, 1994.
- [66] K. Wu (2005). "An Examination of Variability and Its Basic Properties for a Factory", *IEEE Transactions on Semiconductor Manufacturing*, 18(1).
- [67] K. Wu and K. Hui (2008). "The determination and indetermination of service times in manufacturing systems", *IEEE Transactions on Semiconductor Manufacturing*, 21(1).
- [68] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley Interscience, NJ, 1994.
- [69] R. Uzsoy and L. Martin-Vega (1990). "Modeling kanban-based demand-pull systems: A survey and critique", *Manufacturing Review*, 3(3).

- [70] A. Lang and J. L. Arthur, Parameter Approximation for Phase-type distributions. In S. R. Chakravarty and A. S. Alfa, *Matrix-Analytic Methods in Stochastic Models*, Lecture Notes in Pure and Applied Mathematics, Pages 151-206, Marcel Dekker Inc., 1996.
- [71] S. Asmussen, O. Nerman and M. Olsson (1996). "Fitting phase-type distributions via the EM algorithm", *Scandinavian Journal of Statistics*, 23, 419-441.
- [72] A. Riska, V. Diev and E. Smirni (2004). "An EM-based technique for approximating long-tailed data sets with PH distributions", *Performance Evaluation*, 55, 147-164.
- [73] A. Horvath and M. Telek (2000). "Approximating heavy-tailed behavior with Phase-type distributions", *In 3rd International Conference on Matrix-Analytic Methods in Stochastic Models*, Leuven, Belgium, 2000.
- [74] A. P. Dempster, N. M. Laird and D. B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1), 1-38.
- [75] M. A. Johnson and M. R. Taaffe (1991). "An investigation of Phase-distribution moment-matching algorithms for use in queueing models", *Queueing Systems*, 8, 129-148.
- [76] M. A. Johnson and M. R. Taaffe (1988). "A moment-matching algorithm for queueing approximations", *Research Memorandum*, No. 88-24, School of Industrial Engineering, Purdue University.
- [77] M. A. Johnson and M. R. Taaffe (1989). "Matching moments to phase distributions: mixtures of Erlang distributions of common order," *Communications in Statistics-Stochastic Models*, 5(4).
- [78] M. A. Johnson and M. R. Taaffe (1990). "Matching moments to phase distributions: Density function shapes", *Communications in Statistics-Stochastic Models*, 6(2).
- [79] M. A. Johnson and M. R. Taaffe (1991). "A graphical investigation of error bounds for moment-based queueing approximations", *Queueing Systems*, 8, 295-312.
- [80] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modeling*, ASA-SIAM Series on Statistics and Applied Probability, 1999.
- [81] S. K. Gupta and J. K. Goyal (1964). "Queues with Poisson input and hyper-exponential output with finite waiting space", *Operations Research*, 12(1), 75-81.
- [82] Kempf, K., Intel Five-Machine Six Step Mini-Fab Description, Intel/ASU Report, <http://www.eas.asu.edu/aar/research/intel/papers/fabspec.html>, 1994.
- [83] R. Evans (1967). "Geometric distribution in some two-dimensional queueing systems," *Operations Research*, 15(5), 830-846.
- [84] G. Latouche and V. Ramaswami (1993). "A Logarithmic Reduction Algorithm for Quasi-Birth-Death Processes," *Journal of Applied Probability*, 30(3), 650-674.
- [85] A. Thummler, P. Buchholz and M. Telek (2006). "A novel approach for phase-type fitting with the EM algorithm", *IEEE Transactions on Dependable and Secure Computing*, 3(3).

# Appendices

## Appendix I – Queueing Theory Notation

In this section we summarize the notation that is used in Chapter 2 in the context of the queueing formulas.

$\lambda$  – Arrival Rate

$\mu$  – Service Rate

$s$  – Pure Processing Time

$\rho = \frac{\lambda}{\mu}$  – Server Utilization

$m_f$  – Mean Time to Fail

$m_r$  – Mean Time to Repair

$A = \frac{m_f}{m_f + m_r}$  – Server Availability

$t_e = \frac{s}{A}$  – Effective Processing Time

$C^2 = \frac{\sigma^2}{\mu^2}$  – Squared Coefficient of Variation

$C_s^2$  – Squared Coefficient of Variation of Service Time

$C_a^2$  – Squared Coefficient of Variation of Inter-arrival Time

$C_r^2$  – Squared Coefficient of Variation of Repair Time

$C_e^2 = C_s^2 + (1 + C_r^2)(1 - A)A \frac{m_r}{s}$  – Squared Coefficient of Variation of Effective Processing Time

## Appendix II – Little’s Formula

Consider a system in which jobs arrive according to an arbitrary arrival process  $\{A_n, n = 1, 2, \dots\}$ , where  $A_n$  is the arrival time of the  $n$ -th job:  $0 \leq A_1 \leq A_2 \leq \dots$ . The  $n$ -th job to arrive will be identified by the label  $n$  and we assume that more than one job can arrive at a time but the number arriving in any finite time interval is uniformly finite.

Let  $D_n$  be the time instant at which the  $n$ -th job departs the system. Then the number of jobs departed the system during  $(0, t]$  is

$$D(t) = \sup \{n : D_n \leq t, n = 0, 1, \dots\}, D_0 = 0$$

If  $N(t)$  is the number of jobs in the system at time  $t$ , then

$$N(t) = N(0) + A(t) - D(t), \quad t \geq 0 \quad (19)$$

where,

$$A(t) = \sup \{n : A_n \leq t, n = 0, 1, \dots\}$$

with  $A(0) = 0$  being the number of jobs arrived during  $(0, t]$ .

Suppose  $\frac{A(t)}{t} \rightarrow \lambda$  uniformly as  $t \rightarrow \infty$  with probability one, where  $\lambda$  is the job arrival rate and  $\beta = \lim_{t \rightarrow \infty} \frac{D(t)}{t}$  is the departure rate. If the number of jobs in the system is uniformly finite for all time  $t \geq 0$ , then the values of  $\lambda$  and  $\beta$  will be the same and the following flow balance will hold.

$$\text{Arrival rate} = \text{Departure rate}$$

To see that the number of jobs in the system is indeed uniformly finite, or  $\lim_{t \rightarrow \infty} \frac{N(t)}{t} = 0$ , one can divide (19) by  $t$  and take the limits as  $t \rightarrow \infty$ . Let  $D_{K(n)}$  be the time when the  $n$ -th arriving job leaves the system. Then the time spent in the system by the  $n$ -th arriving job is

$$T_n = D_{K(n)} - A_n.$$

We assume that  $T_n, n = 1, 2, \dots$  is uniformly finite and that jobs’ average time in the system,  $\bar{T} = \lim_{k \rightarrow \infty} (1/k) \sum_{n=1}^k T_n$ , exists. That is

$$\bar{T} = \lim_{k \rightarrow \infty} (1/k) \sum_{n=1}^k (D_{K(n)} - A_n) \quad (20)$$

with probability 1.

Let us also assume that the time-average number of jobs in the system exists and is given by  $\bar{N} = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t N(\tau) d\tau$  with probability 1. It can be shown that under these assumptions the *Little’s Formula*

$$\bar{N} = \lambda \bar{T}$$

holds.

Let  $Z_n(t)$  be an indicator variable that takes the value 1 if job  $n$  is in the system at time  $t$  and zero otherwise. Then  $Z_n(t) = 1, A_n \leq t \leq D_{K(n)}$  and  $N(\tau) = \sum_{n=1}^{A(\tau)} Z_n(t), 0 \leq \tau \leq t$ . Hence,

$$\int_0^t N(\tau) d\tau = \int_0^t \left( \sum_{n=1}^{A(\tau)} Z_n(t) \right) d\tau = \sum_{n=1}^{A(\tau)} \{ \min(t, D_{K(n)}) - A_n \}. \quad (21)$$

Let  $K^{-1}(n)$  represent the label of the  $n$ -th departure, then  $A_{K^{-1}(n)}$  is the arrival of the  $n$ -th departure and therefore,

$$\sum_{n=1}^{D(t)} (D_n - A_{K^{-1}(n)}) \leq \sum_{n=1}^{A(\tau)} \{\min(t, D_{K(n)}) - A_n\} \leq \sum_{n=1}^{A(\tau)} \{D_{K(n)} - A_n\}. \quad (22)$$

Because  $\lim_{t \rightarrow \infty} \frac{1}{t} \sum_{n=1}^{A(\tau)} \{D_{K(n)} - A_n\} = \lim_{t \rightarrow \infty} \left(\frac{A(t)}{t}\right) \left(\frac{1}{A(t)}\right) \sum_{n=1}^{A(t)} \{D_{K(n)} - A_n\} = \lambda \bar{T}$ , from Equations (20) and (21) and the second inequality in (22) it is clear that  $\bar{N} \leq \lambda \bar{T}$ . Let  $t'$  be the time by which all the jobs arrived during  $(0, t]$  will have left the system, that is

$$t' = \sup \{D_{K(n)} : n = 1, 2, \dots, A(t)\}.$$

Then

$$\sum_{n=1}^{A(\tau)} \{D_{K(n)} - A_n\} \leq \sum_{n=1}^{D(t')} \{D_n - A_{K^{-1}(n)}\}. \quad (23)$$

Note that by assumption  $T_n$  is uniformly finite for all  $n$ , therefore  $t' - t$  is uniformly finite with probability one and in particular  $t' \rightarrow \infty$  as  $t \rightarrow \infty$ . Therefore dividing inequality (23) by  $t$  and taking the limit as  $t \rightarrow \infty$  Buzacott and Shanthikumar [7] show that

$$\lambda \bar{T} \leq \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{n=1}^{D(t')} \{D_n - A_{K^{-1}(n)}\} = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{n=1}^{D(t)} \{D_n - A_{K^{-1}(n)}\}. \quad (24)$$

Now from (21) and the first inequality in (22) and (16) it follows that  $\bar{N} \geq \lambda \bar{T}$ . This combined with the previous observation  $\bar{N} \leq \lambda \bar{T}$  implies that  $\bar{N} = \lambda \bar{T}$ .

Note that other than uniform finiteness of the time spent by jobs in the system and the existence of the appropriate limits  $\bar{N}$  and  $\bar{T}$  no additional assumptions are needed to draw this conclusion. Particularly observe that no assumptions regarding the availability of servers or variation in the arrival rate is made. Therefore, even if the servers are interrupted or the arrival rate is variable the preceding formula derivation remains valid.

## Appendix III – Counting The Number of States

When the state space of the model is expanded to account for the Erlang phases in arrival and processing time distributions calculating the number of states is not trivial. For a system with maximum of  $w^{\max}$  WIP,  $k^{\max}$  number of tools,  $P^{\max}$  phases of processing time distribution and  $A^{\max}$  phases of inter-arrival distribution the enumeration of all the states is calculated  $(P^{\max} + 2)^{k^{\max}} \times w^{\max} \times A^{\max}$ . Note that each tool can be in one of the three general states of "failed", "idle", or "busy". If the tool is busy processing a lot then it can be in any of the  $P^{\max}$  phases of process. Therefore the total number of states for each tool is  $(P^{\max} + 2)$ .

However out of the  $(P^{\max} + 2)^{k^{\max}} \times w^{\max} \times A^{\max}$  possible states there are many that are infeasible. For example when there is no WIP in the system none of the tools can be in any of its  $P^{\max}$  processing states. Or when the WIP level is higher than the number of active tools none of the tools can be in the idle state. This can be demonstrated as the positions for the permutations of tool states, WIP level and arrival phases as follow.

$$\boxed{k^1} \quad \boxed{k^2} \quad \dots \quad \boxed{k^{\max}} \quad \boxed{w} \quad \boxed{A}$$

To find the number of feasible states for toolsets with Erlang distributions we apply the following method. We divide the feasible states into four categories based on the WIP level,  $w$ . These four categories include  $w = 0$ ,  $0 < w < k^{\max}$ ,  $k^{\max} \leq w < w^{\max}$  and finally  $w = w^{\max}$ . We first calculate the number of feasible states in each category and then sum the value over all categories.

1- When  $w = 0$  then each of the  $k^{\max}$  tools can either be in idle state,  $I$ , or in failed state  $F$ . The WIP level,  $w$ , has only one possibility and the phase of the arrival can be in any of the  $A^{\max}$  phases. Therefore the total number of feasible states for this category is calculated as  $2^{k^{\max}} \times 1 \times A^{\max}$ .

2- When  $0 \leq w < k^{\max}$  then the WIP position can be in  $\max\{(k^{\max} - 1), w\}$  states and the arrival process can be in one of  $A^{\max}$  phases. To find the possible states of the tools however, we need to divide the states into subcategories based on the number of active tools. Let  $k^F$  denote the number of failed tools and  $i$  denote the number of tools that are active. It follows that  $i = k^{\max} - k^F$ .

a) If  $k^F = k^{\max}$  then  $i = 0$  and there is only 1 permutation and that is when all the  $k^{\max}$  tools are failed.

b) When  $i \leq w$  then all the active tools must be in the *busy* state. In this case for  $i = 1, \dots, w$  each of the  $i$  busy tools can be in one of the  $P^{\max}$  phases of the process. Therefore we have  $\sum_{i=1}^w (P^{\max})^i$  different possibilities for the busy tools and the permutation of busy tools among  $k^{\max}$  tools is found by  $\binom{k^{\max}}{i}$ . The total number of feasible states in this subcategory is  $\sum_{i=1}^w (P^{\max})^i \times \binom{k^{\max}}{i}$ .

c) When  $i > w$  then we will have  $w$  busy tools each in one of the  $P^{\max}$  states. The combination of failed tools is  $\binom{k^{\max}}{k^F}$  and  $\binom{i}{w}$  is the number of combinations to position  $w$  busy tools among the  $k^{\max}$  tools. The rest of the tools are either failed or idle. Therefore the total number of states is calculated as  $(P^{\max})^w \binom{k^{\max}}{k^F} \binom{i}{w}$ .

3- When  $k^{\max} \leq w < w^{\max}$  then the number of active tools is smaller than the WIP level and therefore we cannot have any idle tool. Each tool is either failed or in one of the  $P^{\max}$  phases of processing. The possible number of WIP levels in this case is  $w^{\max} - k^{\max}$  and at each WIP level the phases of arrival can be in one of the  $A^{\max}$  phases. This gives a total number of feasible states as  $(P^{\max} + 1)^{k^{\max}} \times (w^{\max} - k^{\max}) \times A^{\max}$ .

4- When  $w = w^{\max}$  then the only difference with category 3 is that when the WIP level reaches its limit the arrival rate is cut off and different combinations for  $A^{\max}$  should not be considered. Therefore the total number of feasible states is calculated as  $(P^{\max} + 1)^{k^{\max}}$ .

Therefore the total number of feasible states is calculated as

$$2^{k^{\max}} \times 1 \times A^{\max} + \quad (25)$$

$$\begin{aligned} & \sum_{w=1}^{k^{\max}-1} \left[ 1 + \sum_{i=1}^w (P^{\max})^i \times \binom{k^{\max}}{i} \right. \\ & + \sum_{i=w+1}^{k^{\max}} (P^{\max})^w \binom{k^{\max}}{k^F} \binom{i}{w} \left. \right] \times \max\{(k^{\max}-1), w\} \times A^{\max} \\ & + (P^{\max} + 1)^{k^{\max}} \times (w^{\max} - k^{\max}) \times A^{\max} + (P^{\max} + 1)^{k^{\max}}. \end{aligned} \quad (26)$$

For example for a toolset with  $k^{\max} = 2$ ,  $w^{\max} = 4$ ,  $A^{\max} = 2$ ,  $P^{\max} = 1$  the calculations for the number of states is as follows.

For  $w = 0$  we have  $2^2 \times 1 \times 2 = 8$  states. For  $0 < w < k^{\max}$  or  $w = 1$  we have the three subcategories.

a) For  $k^F = k^{\max} = 2$  we have 1 combination that is all tools are failed.

b) For  $i \leq w$  it follows that  $i = 1$  and we have  $\sum_{i=1}^1 (1)^1 \times \binom{2}{1} = 2$ .

c) For  $i > w$  we have  $i = 2$  and  $(1)^1 \binom{2}{1} = 2$  feasible states.

Now considering the choices for the WIP level and arrival phase in this category we have a total of  $(1 + 2 + 2) \times 1 \times 2 = 10$ . For  $k^{\max} \leq w < w^{\max}$  or  $w = 2, 3$  we have  $(1 + 1)^2 \times (4 - 2) \times 2 = 16$ , and Finally for  $w = w^{\max}$  we have  $(1 + 1)^2 = 4$  states. The total number of states sums up to  $8 + 10 + 16 + 4 = 38$ . For this small example it is easy to verify the results by enumerating all the possible states and we can show that the results match.

## Appendix IV - Selected Utilization Graphs

In this Appendix we present the utilization graphs of selected instances in Case 1 through Case 6 in Chapter 6. For the instances with the No Rule scenario we also present the results for the classical  $G/G/m$  approximations.

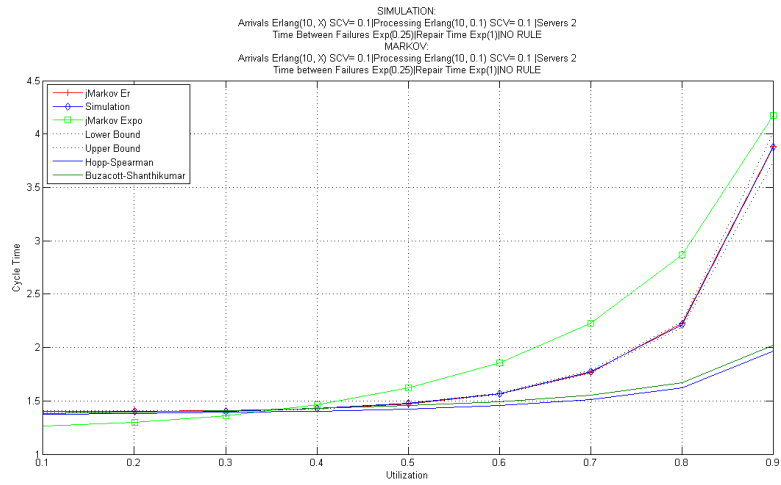


Figure 20: Utilization Graph 1

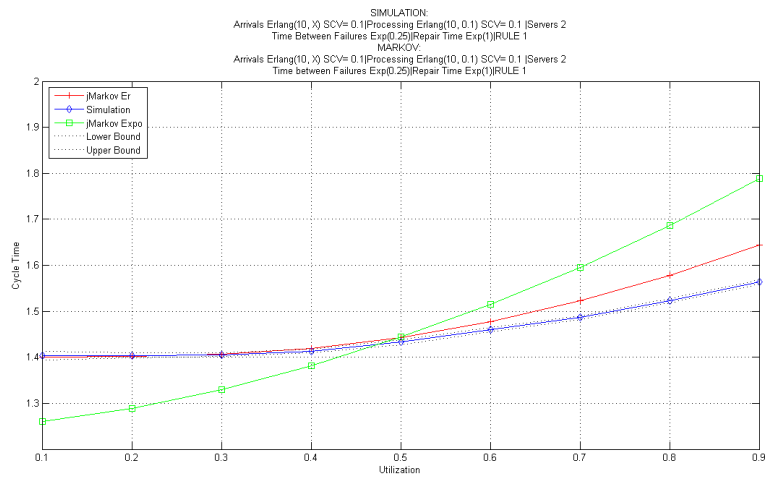


Figure 21: Utilization Graph 2

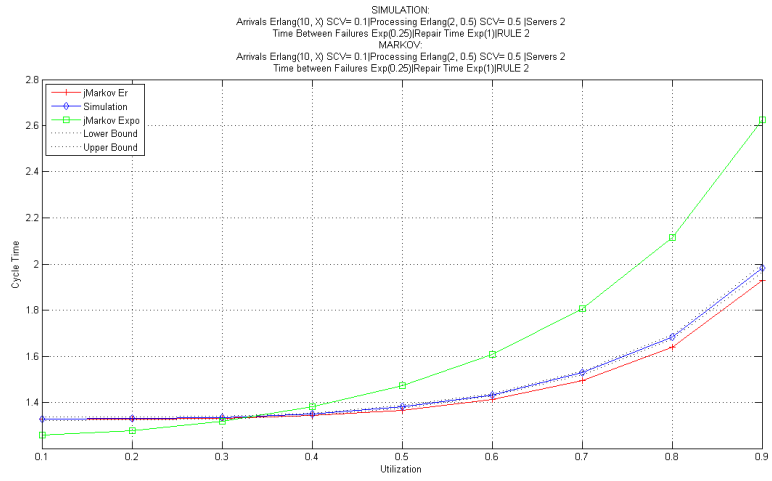


Figure 22: Utilization Graph 3

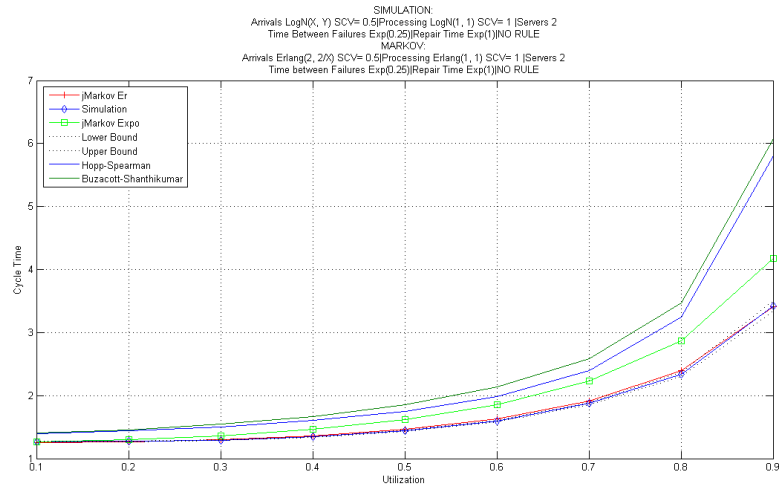


Figure 23: Utilization Graph 4

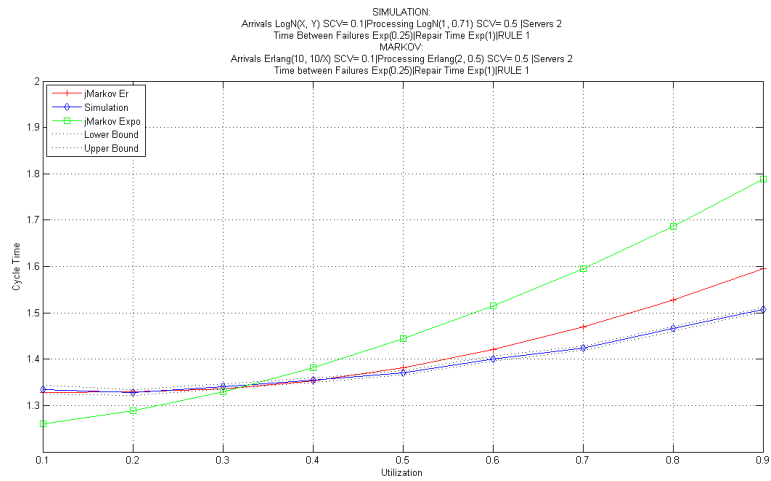


Figure 24: Utilization Graph 5

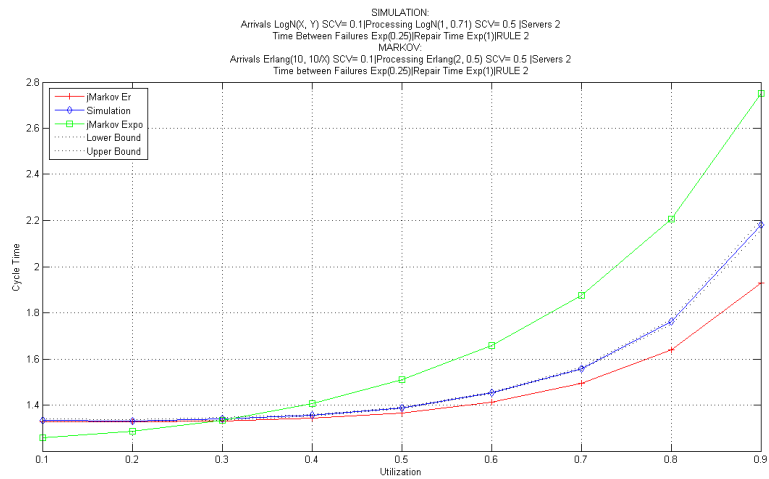


Figure 25: Utilization Graph 6

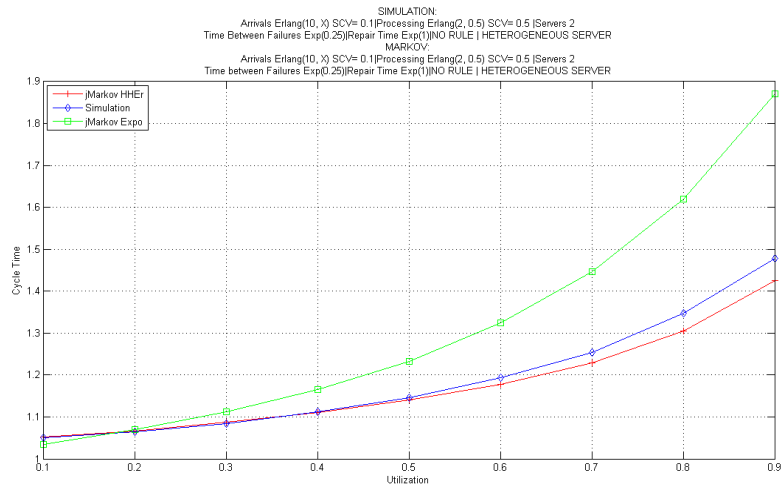


Figure 26: Utilization Graph 7

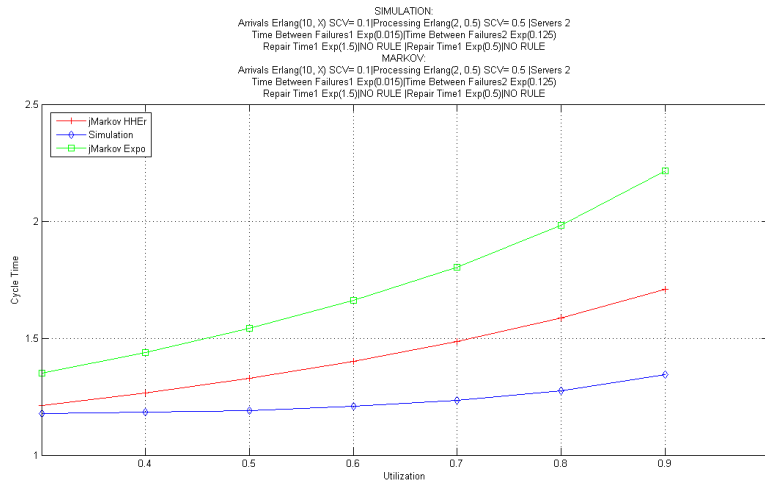


Figure 27: Utilization Graph 8