Fall 2016

Development of a DSM-Based Model for Managing the Design of Complex Systems Considering the Impact of Technological Obsolescence

Gershom Kwaku Obeng
Old Dominion University, gkobeng88@gmail.com

DEVELOPMENT OF A DSM-BASED MODEL FOR MANAGING THE DESIGN OF COMPLEX SYSTEMS CONSIDERING THE IMPACT OF TECHNOLOGICAL OBSOLESCENCE

by

Gershom Kwaku Obeng
B.S. May 2011, Old Dominion University
M.S. December 2013, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

MECHANICAL ENGINEERING

OLD DOMINION UNIVERSITY
December 2016

ABSTRACT

DEVELOPMENT OF A DSM-BASED MODEL FOR MANAGING THE DESIGN OF COMPLEX SYSTEMS CONSIDERING THE IMPACT OF TECHNOLOGICAL OBSOLESCENCE

Gershom Kwaku Obeng
Old Dominion University, 2016
Director: Dr. Han Bao

Obsolescence has been a constant issue for process planners and designers and if not properly accounted for, obsolescence can become an expensive issue. As systems become more complex, interconnected, and nonhomogeneous, separate studies of single groups of equipment are no longer sufficient in modeling the obsolescence of the systems that they make up. The purpose of this dissertation is to model the likelihood of a process's design becoming obsolescent given the obsolescent behavior of the equipment used to complete the process. The methodology discussed in this work is based on a combination of using a systems engineering tool called the DSM (Design Structure Matrix) and a technological forecasting tool known as the growth curve to simulate the duration and cost of completing a process, and to determine if the process's design will keep its utility as it ages.

This thesis is dedicated to my father Ebenezer, my mother Ama and the rest of my family and friends.

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Han Bao for allowing me to work with him on his research and for the useful advise on being successful academically and professionally. I would also like to thank Dr. Tony Dean, Dr. Jennifer Micheali and the rest of ASNE (American Society Of Naval Engineers) for the invites to conferences, facility tours and helping me expand my network. Many thanks to Old Dominion University and its NSBE (National Society Of Black Engineers) chapter for the professional development and providing many good friends.

NOTATIONS

i	a row number in a DSM or table unless otherwise specified
j	a column number in a DSM or table unless otherwise specified
n	number of tasks needed to complete a process
$v_{(i,j)}$	the value of element (i, j) of a DSM
$mutmaster$	mute (mutation) master
PBM	Percent of Backtracking Move
LM	Loop Master
m	number of rows in the final loop master
$DSMTS$	Design Structure Matrix for Task Sorting
a^p	adjacency matrix raised to power p
RC	Rework Coefficient of task(s)
$DSMRStart$	Design Structure Matrix for Rework Starts
$DSMRate$	Design Structure Matrix for Rates
$DSMMAT$	Design Structure Matrix for MATurities
$DSMR(t)$	Design Structure Matrix for Rework at year t
T	Duration of Tasks Vector
C	Cost of Tasks Vector
$Redo$	Redo Matrix
T_{total}	total duration of a process
T_0	duration of zero order rework
T_1	duration of first order rework
T_2	duration of second order rework
C_{total}	total cost of a process
C_0	cost of zero order rework
C_1	cost of first order rework
C_2	cost of second order rework

FOV	First Order Vector
$T_{embedded}$	duration of second order rework of an embedded loop
$C_{embedded}$	cost of second order rework of an embedded loop

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Importance of Technological Forecasting in Process Planning

1.1.1 Background of Study

Systems Engineering has been used many times to ensure that projects have been completed within duration and cost constraints. Locatelli et al. [1][1] suggested that project managers using Systems Engineering should not only look inward at themselves, but also outward to ensure that projects are delivered and perform successfully. Because of changing standards of customers and/or failing equipment, the ability of a manager to meet goals on time and budget becomes increasingly difficult because of equipment or process obsolescence. During the design phase of a process, attempts are made so that the process in question can continuously meet given targets during its useful life and keep its utility. The design phase of a process is the best time to make changes because the process has not been implemented yet, and all changes are on paper and/or digital in nature when it is cheapest.

1.1.2 Purpose of Study

The purpose of this study is to develop a method of modeling obsolescence effects on process duration and cost due to the changes in properties of the interactions between tasks in a process. These interactions are typically represented in a Design Structure Matrix (DSM) in its design phase.

1.1.3 Statement of the Problem

Obsolescence impacts a process's task interfaces through deterioration of performance and quality necessitating rework. Few technological forecasting methods exists that aid in address-

[1]The documentation style in this dissertation is written in format of the American Society of Mechanical Engineers (ASME).

ing obsolescence effects on interface interactions during the design phase of a process. Common methods of forecasting include analogies and extrapolations of curves such as Neilson's law of bandwidth. Analogies and extrapolations are commonly used for examining obsolescence in equipment but not the effects of obsolescence on the behavior of non-homogeneous equipment-equipment and equipment-method task interfaces. The DSM is used to study and model equipment interfaces by studying the tasks to which they are applied, but current DSM based literature does not consider rework whose probability of rework is not constant due to obsolescence. Process design, however, must consider also the requirement that a process's duration and cost will stay below acceptable thresholds as the probability of rework changes due to obsolescence.

1.1.4 Research Hypotheses

As time passes, the quality of interactions between tasks degrades because the equipment used to complete them becomes obsolescent. This will make the probability of rework increase. As the probability of rework increases, the distribution of time and cost will change.

1.2 Dissertation Overview

1.2.1 Preview of Chapters

Chapter 2 contains a literature review. This chapter will present some popular tools in examining obsolescence. Then, an explanation of how to use DSMs will be given along with some applications. At the end of this chapter, a literature review of papers that represent the intersection of obsolescence and DSMs research will be given along with an extended description of the gaps in the literature that this dissertation will attempt to address.

Chapter 3 explains the methodology of modeling obsolescence effects on process duration and cost due to the failing interactions between tasks in a process by using Monte Carlo simulations. Software was written in MatLab (Matrix Laboratory Version R2013b) for automation purposes, and the software's architecture will be explained in this chapter. This chapter will refer to

Appendix A which contains the software's codes.

Chapter 4 will demonstrate the use of the software in a case study. This chapter will show the manual calculation of one data point with the methodology explained in Chapter 3. Discussion of results will be given in this chapter.

Chapter 5 has the conclusion and suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

A literature review will be given in this chapter. The first section will explain the definition of obsolescence and review the use of curve fitting and extrapolation for measuring obsolescence. The second section will explain what a DSM is, how to read one, and some recent applications of the DSM. The third section will discuss the intersections of obsolescence and DSM research. The final section of this chapter will point out the gaps in research that this dissertation will attempt to address in greater detail.

2.1 Review of Obsolescence

2.1.1 Definition of Obsolescence

The definition of obsolescence in this dissertation is the inability of a piece of equipment or a process to perform at a certain standard. For example, a company has a standard of fuel consumption that is 25 miles/gallon buys a truck that meets that standard now; later, the truck ages, and it operates at 20 miles/gallon. In the company's eyes, the truck is now obsolete because it cannot meet the company's standard. Obsolescence can also come from a change in the standard. An example of such a situation is if the truck manages to continually operate at 25 miles/gallon and the company's operations now require a truck that operates at 30 miles/gallon the truck is obsolete in the company's eyes. In both cases, the occurrence of obsolescence is involuntary because all equipment will eventually age when used and, organizations must continuously make improvements to be competitive.

2.1.2 How Obsolescence is Measured

The issue with the use of analogies is that an existing process is needed for comparison. A notable example of a failed application in forecasting using analogies was explained by Martino in [2]. The French engineer deLesseps tried to construct the Panama Canal based on his experience of the Suez Canal 40 years earlier. The Suez Canal was made on flat, desert terrain by moving sand and mud. The Panama Canal needed to be built in a swampy jungle with many mountains. The types of soil that had to be moved for the Panama Canal were rock and clay. The first attempt to construct the Panama Canal failed because the issues of designing the locks needed to over the mountains and the epidemics of malaria and yellow fever could not be solved.

It is common to use the extrapolation of curves to measure obsolescence. An example of a basic extrapolation is for studying a phenomenon called Nielson's Law of internet bandwidth shown in Fig. 2.1. The data points Nielson [3] collected were obtained by measuring the bandwidth given from high end internet providers, and it was determined that the internet providers' connectivity grows by 50% per year. This has become an expectation of customers of internet service.

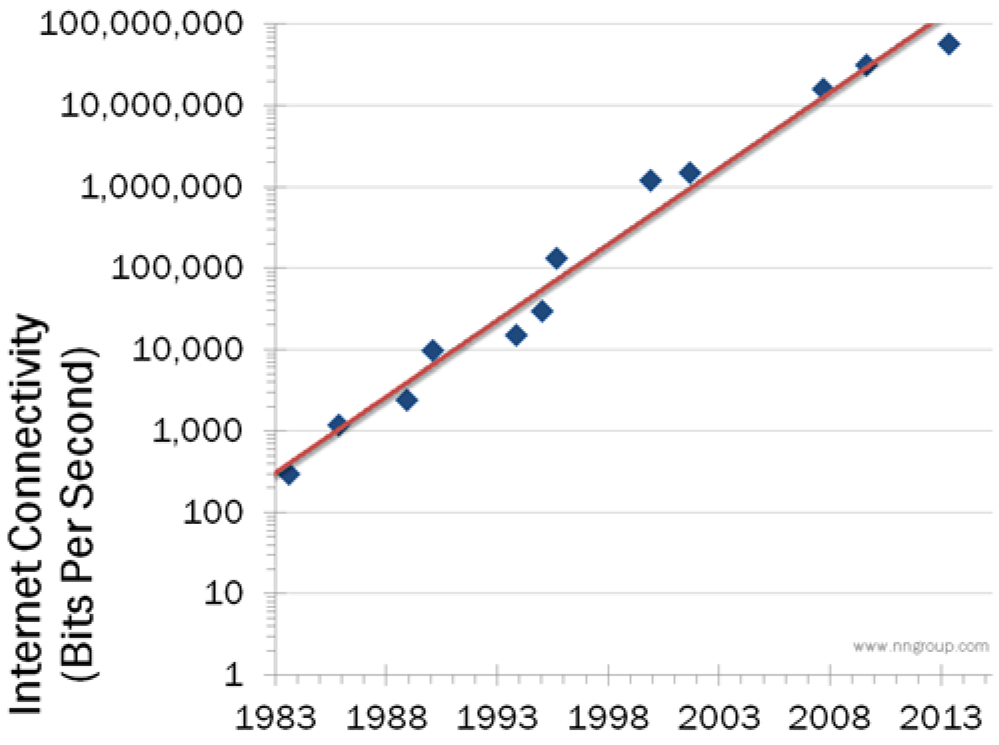


Figure 2.1: Plot Showing Nielson's Law. (Made by Nielson [3])

Biological systems provide a good paradigm for the life cycle of products and man-made systems. The s-shaped growth curve has been used to model market share and failure rates of different products. An example of a growth curve and its derivative is shown in Fig. 2.2 made by Kucharavy and DeGuio [4]. The life cycle of a product is in four stages. The first stage is when the product is first introduced to the market and growth rate of market share and/or failure is increasing slowly. The second stage occurs when growth is increasing at an exponential rate. The third stage starts when the growth rate begins to slow down. The fourth and final stage occurs when the growth rate is level. In economic terms, the product has reached saturation and no more can be sold. In failure terms, all devices in a given batch are "broken."

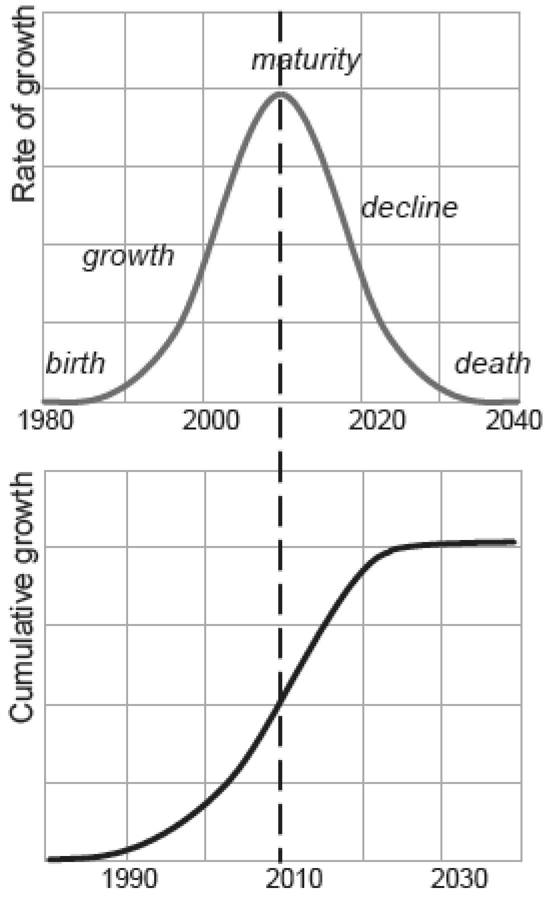


Figure 2.2: An Cumulative S-Curve And Bell Shaped Curve Showing Change In Rate. (Made by Kucharavy and DeGuio [4])

Figure 2.3 shows an example of an application of using a growth curve for technological forecasting by Intepe and Koc [5]. They predicted that the number of patents for 3D televisions will stop at around 500 in the year 2050. Growth curves are good for forecasting because no previous item is needed for comparisons and most of the subjectivity is removed because data is collected by simple counting.

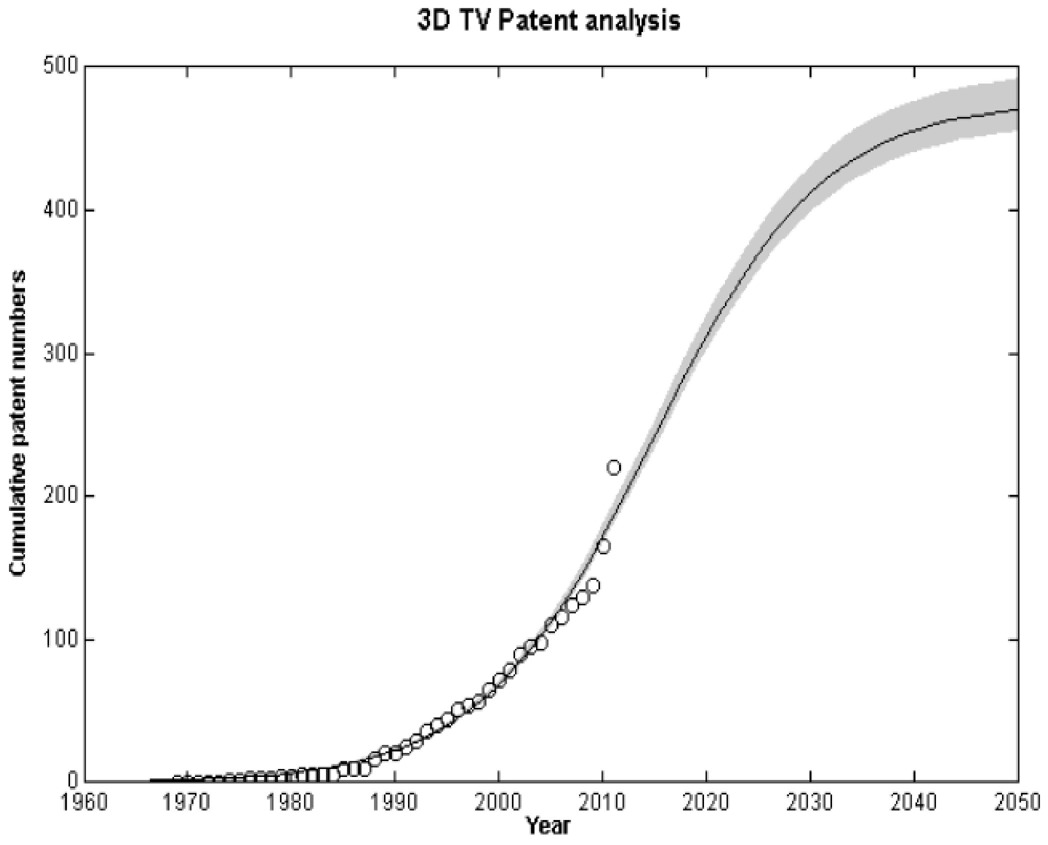


Figure 2.3: Growth Curve Of 3D Television Patents. (Made by Intepe and Koc [5])

Not all growth curve literature measures cumulative growth in terms of market patents. Many authors such as Lam, et al. [6] and Ryu and Byeon [7] used growth curves to track market saturation of devices. Miranda and Lima [8] conducted a study on the market saturation of digital cameras in the United States. The use of growth curves for examining market saturation is as common as counting unit sales of devices. Figure 2.4 made by Miranda and Lima [8] contains a growth curve based on counting sales of digital cameras.

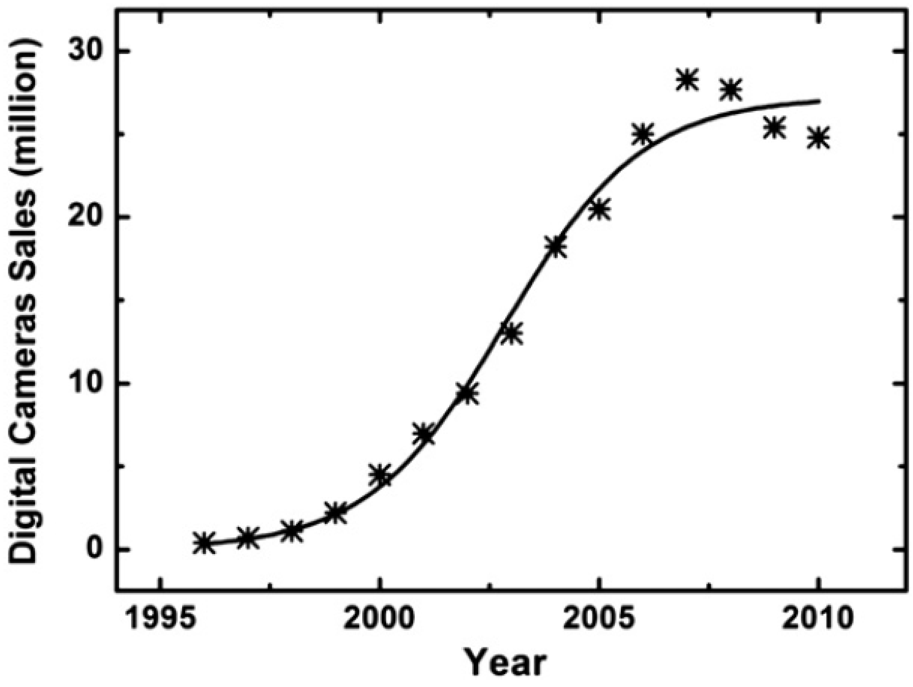


Figure 2.4: Growth Curve Of Digital Camera Sales. (Made by Miranda and Lima [8])

The literature also contains various attempts that use the derivative of the growth curve in Fig. 2.2 to measure the obsolescence of devices. The derivative of the growth curve is shaped as a Gaussian curve and is treated as such. An example of how a Gaussian curve is used to fit units shipped per unit time was made by Solomon, et al. [9] and is shown in Figure 2.5. Such authors use the time when the most units are shipped and/or sold to mark the time of device maturity, μ (mean), and σ (standard deviation) is used to frame the phases of the device's life cycle. Figure 2.6 contains a life cycle curve that was fitted by Karls et al. [10] for a type of device known as a DRAM (Dynamic Random Access Memory). What Karls et al. [10] predicted was that the sale of DRAMs would peak in the first quarter of 1998 with 2500 million units being shipped at that time. The standard deviation is 1.6 years. The DRAMs were expected to become obsolete in early 2002. Bartels et al. [11] used life cycle curves to measure the obsolescence of various devices such as SRAMs (Static Random Access Memory), microprocessors, microcontrollers, logic parts, analog parts, and different types of integrated circuits.

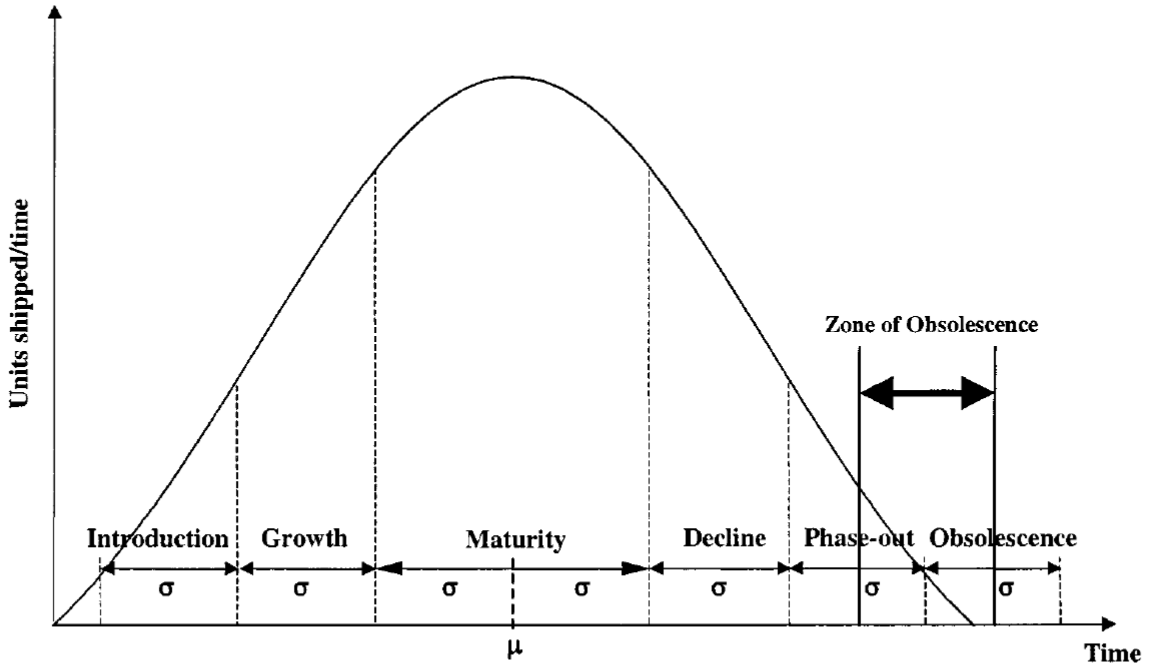


Figure 2.5: Derivative Of A Growth Curve With Life Cycle Phases. (Made by Solomon, et al. [9])

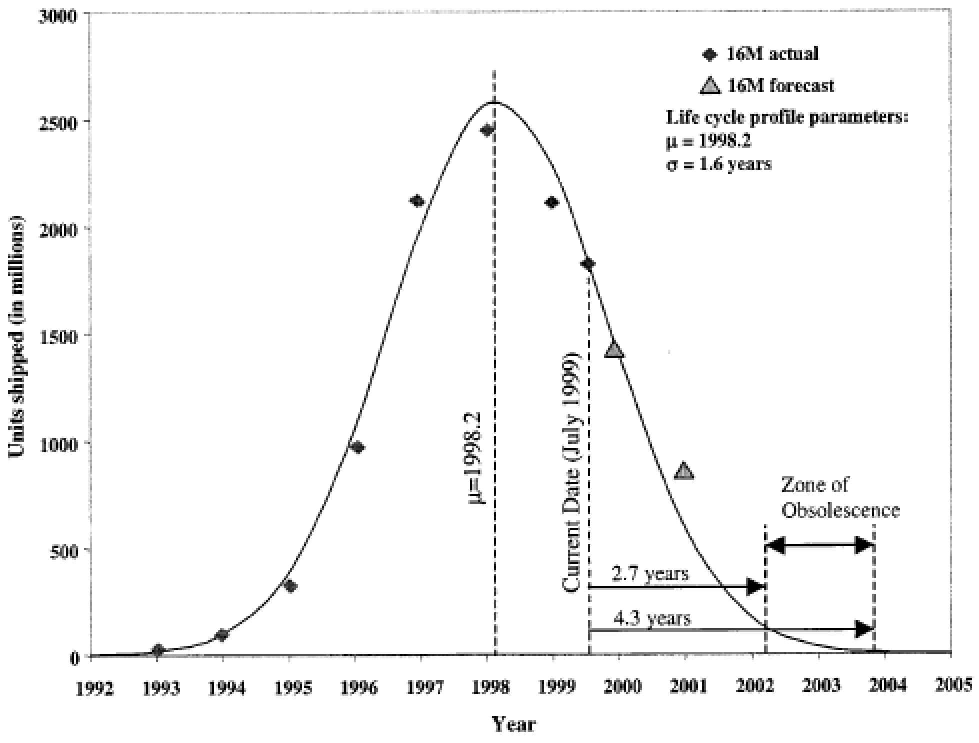


Figure 2.6: Life Cycle Curve For DRAMS. (Made by Karls et al. [10])

Barreca [12] has shown a methodology that involves treating obsolescence as a mode of failure for devices from market share and device sales. Barreca has expressed concern about not including the effects of the obsolescence of equipment while creating maintenance plans. Barreca maintained that if equipment obsolescence is considered while making a maintenance plan then the probability of failure for this equipment would be higher than the probability of failure due to the physical deterioration alone. Barreca explained that it is important to combine failure by obsolescence with failure by physical means to make the failure analyses of devices more accurate. Figure 2.7 made by Barreca [12] shows an illustration of such a situation. When the total probability of failure is lower at a given time than expected, the cost associated with these failures can catch equipment operators off guard, and budget overruns can occur. Obeng [13] has written software to implement Barreca's methodology of combining the failure curves and used it to estimate the total probability of failure of computer hard drives with Bao [24] and Obeng [25] for various types of cars.

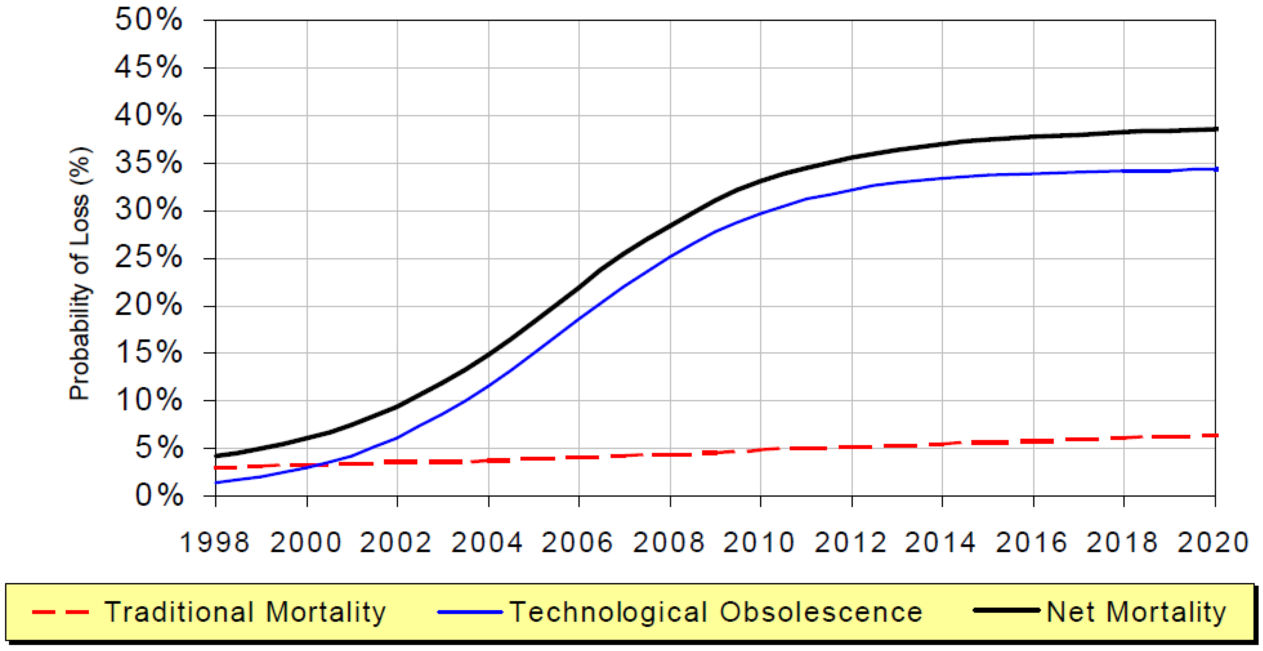


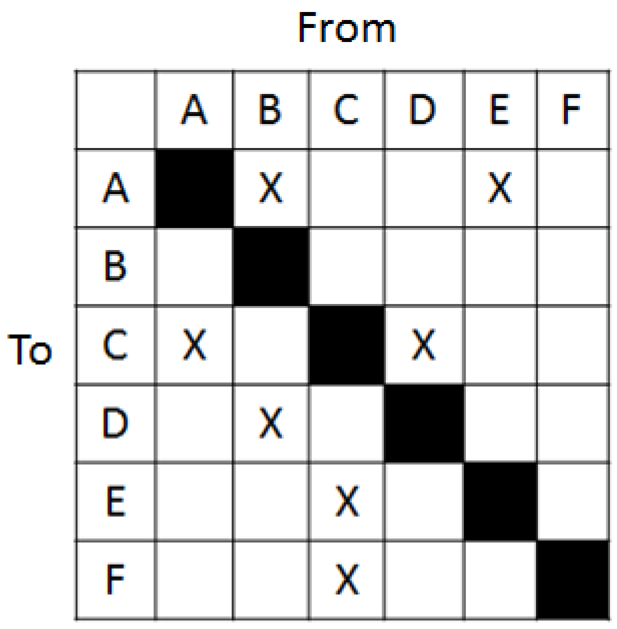
Figure 2.7: Failure Curves Representing Modes Of Failure. (Made by Barreca [12])

2.2 Review of the Design Structure Matrix (DSM)

2.2.1 What is a DSM and How to Read One

The purpose of the DSM is to model a system by showing how components interact with each other. There are four types of DSMs respectively for product architecture, organization architecture, process architecture and multidomain that is a combination of the previous three. The type of DSM that will be used for this research is the process based DSM. The process based DSM is used to show how tasks interact. The DSM is an n x n matrix where n is the number of tasks needed to complete a process. Interactions that exist between tasks are usually shown with a "one" or "X." Non-binary DSMs use numbers to show the weight of interactions.

Figure 2.8 shows an example of a simple, binary DSM. To determine the order of completing the tasks in the process, read the headings on the rows or columns. The notation used in this research is called IR/FAD (Input shown in Rows/ Feedback is Above Diagonal). Interactions below the diagonal are feedforward moves, and interactions above the diagonal are feedback moves. Reading the rows shows which tasks need input from which others and reading the columns shows which tasks give output to which others. Reading across the first row in Figure 2.8 shows that Task A needs inputs from Task B and Task E. Reading down the first column in Figure 2.8 shows that Task A gives output to Task C. Figure 2.9 shows the DSM in Figure 2.8 after being optimized. Figure 2.10 shows the resulting digraph (directed graph). The digraph is a chart that graphically shows how tasks in a project interact. The digraph in Figure 2.10 is read from left to right. The optimal order of doing the tasks are B, D, E, A, C, and F is completed last. Note that some tasks can be performed concurrently. For example, Tasks B and E can be performed at the same time. The process in this case has one incident of feedback (rework) if completed in the optimal order. Tasks E, A, and C form a loop (or circuit). A loop is a group of tasks that repeat because of the existence of feedback moves.

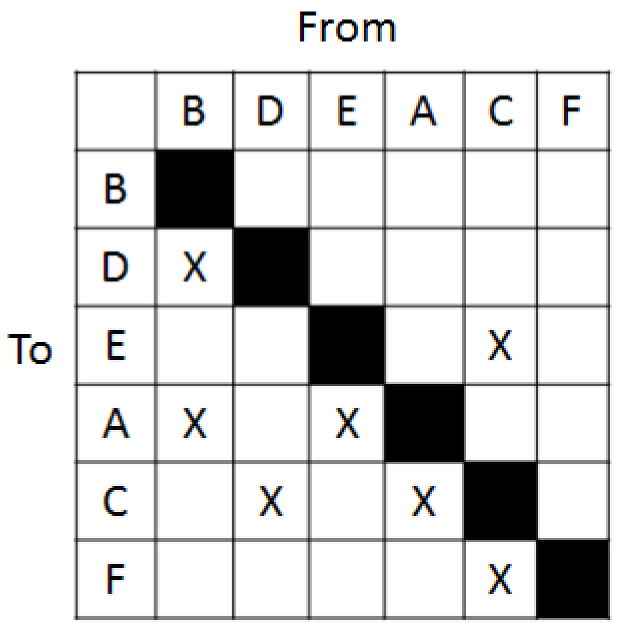


From

To

	A	B	C	D	E	F
A	■	X			X	
B		■				
C	X		■	X		
D		X		■		
E			X		■	
F			X			■

Figure 2.8: DSM Example.



From

To	B	D	E	A	C	F
B	■					
D	X	■				
E			■		X	
A	X		X	■		
C		X		X	■	
F					X	■

Figure 2.9: DSM From Figure 2.8 After Optimization.

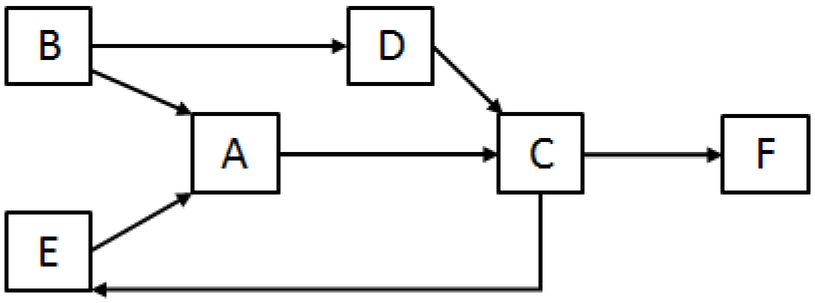


Figure 2.10: Digraph Of DSM From Figure 2.8.

Figure 2.11 was made by Browning [14] and shows the types of interactions that exist between the tasks in a project and how they are represented in a DSM. The DSM is better at modeling interactions than methods such as the critical path method (CPM) and Gantt charts because the latter two cannot properly represent feedback.

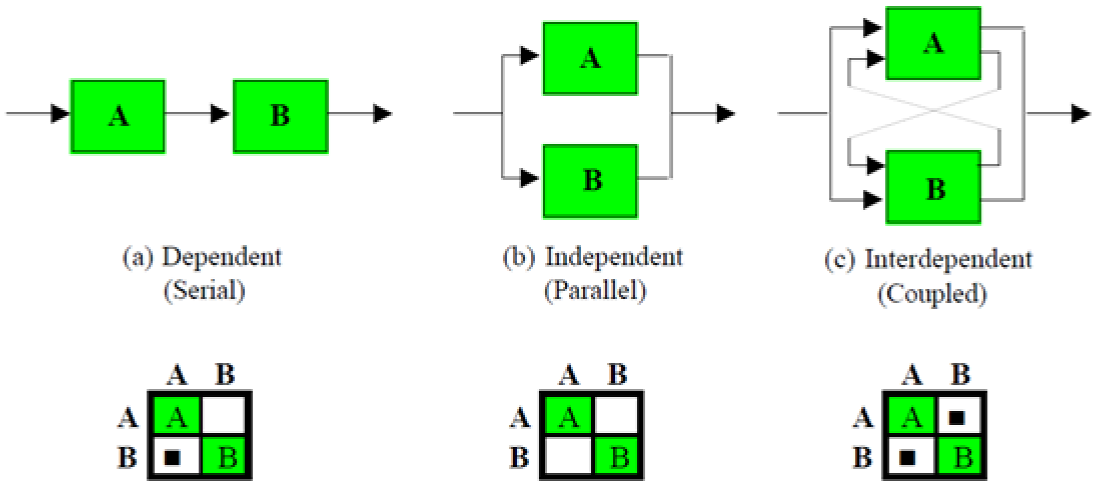
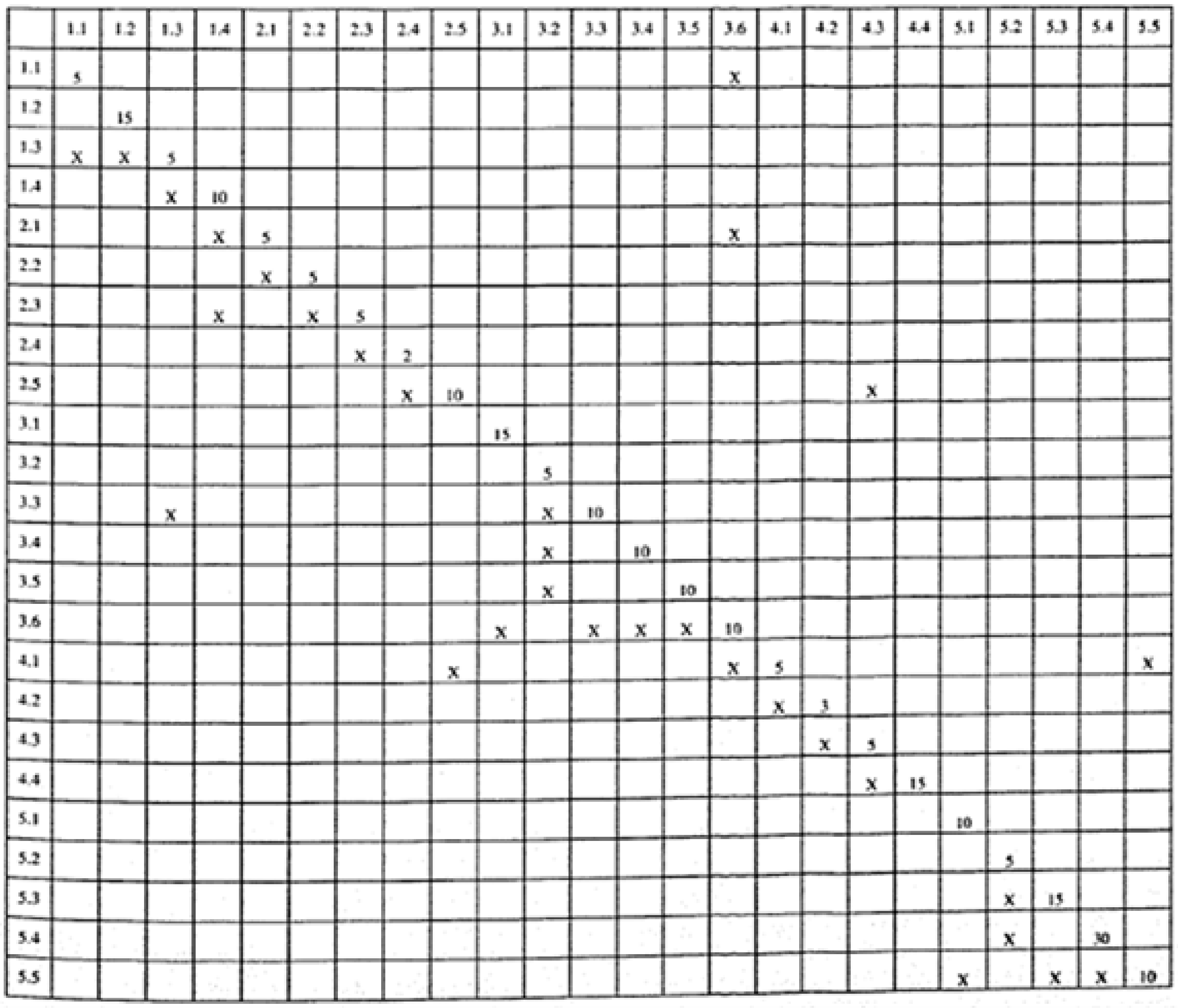


Figure 2.11: Types Of Tasks Interactions. (Made by Browning [14])

2.2.2 Examples of DSM Applications

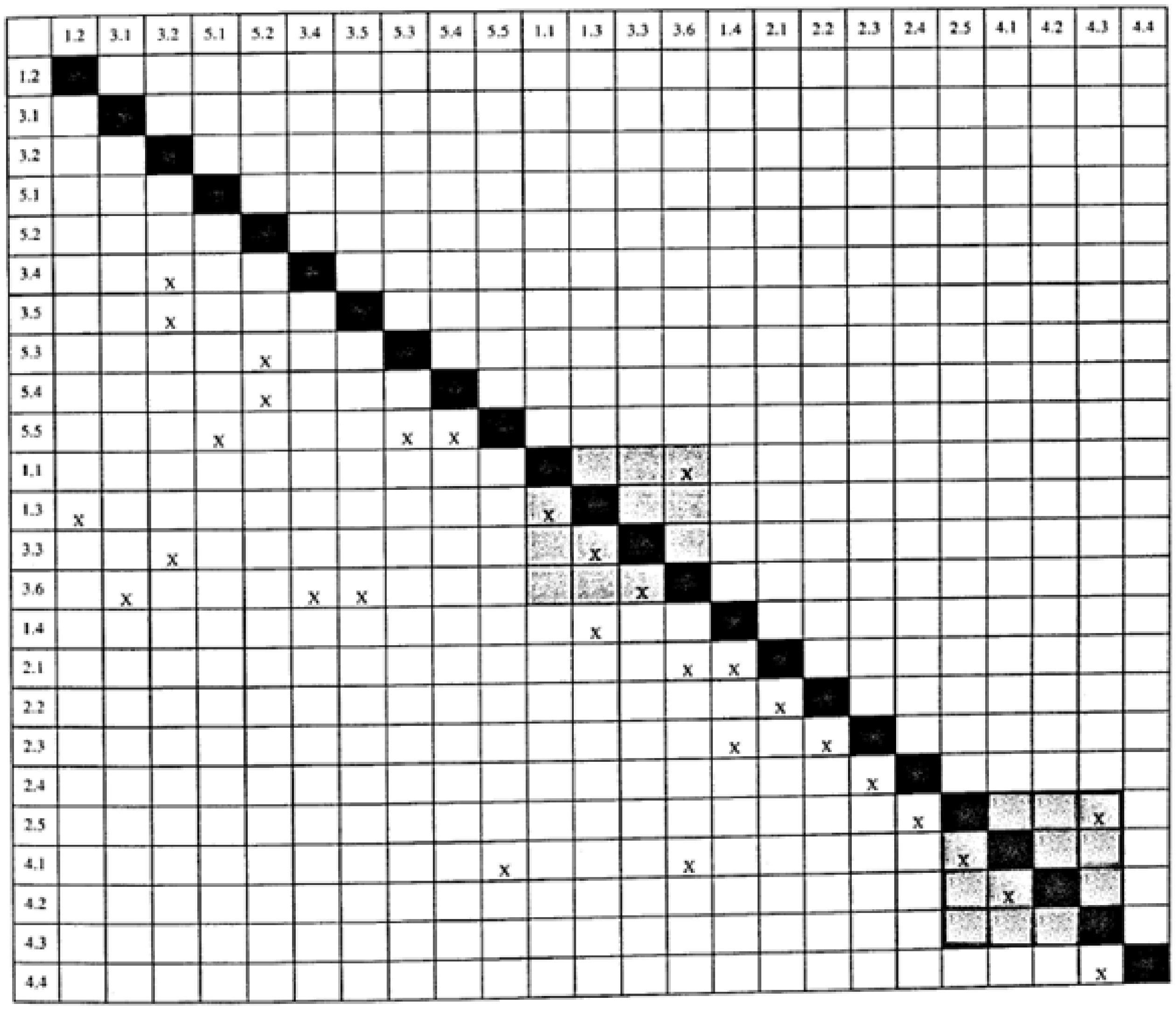
Figures 2.12 and 2.13 contain an application of binary DSMs by Gunawan [15] for the development of an oil field. Figure 2.12 shows the unsolved DSM, and Fig. 2.13 shows the solved DSM. The highlighted blocks in Fig. 2.13 show which tasks form loops. In this case, Tasks 1.1, 1.3, 3.3, and 3.6 are part of a loop, and Tasks 2.5, 4.1, 4.2, and 4.3 are part of another loop. The first five tasks can be completed in parallel.



	1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4	2.5	3.1	3.2	3.3	3.4	3.5	3.6	4.1	4.2	4.3	4.4	5.1	5.2	5.3	5.4	5.5
1.1	5														X									
1.2		15																						
1.3	X	X	5																					
1.4			X	10																				
2.1				X	5										X									
2.2					X	5																		
2.3				X		X	5																	
2.4							X	2																
2.5								X	10									X						
3.1										15														
3.2											5													
3.3			X								X	10												
3.4											X		10											
3.5											X			10										
3.6										X		X	X	X	10									
4.1									X						X	5								X
4.2																X	3							
4.3																	X	5						
4.4																		X	15					
5.1																				10				
5.2																					5			
5.3																					X	15		
5.4																					X		30	
5.5																				X		X	X	10

Note: The numbers on the diagonals indicate days needed to complete its corresponding task.

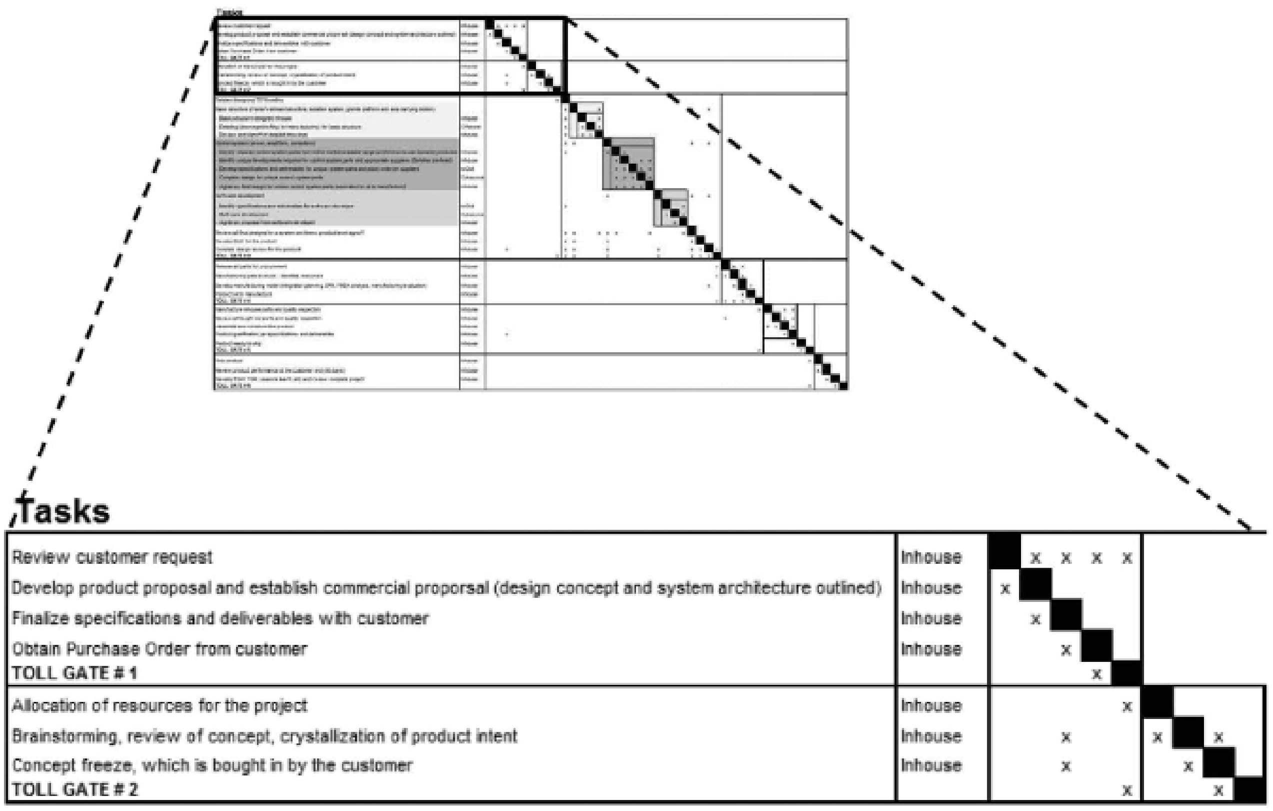
Figure 2.12: Staring DSM For Oil Field Development Project. (Made by Gunawan [15])



	1.2	3.1	3.2	5.1	5.2	3.4	3.5	5.3	5.4	5.5	1.1	1.3	3.3	3.6	1.4	2.1	2.2	2.3	2.4	2.5	4.1	4.2	4.3	4.4
1.2	■																							
3.1		■																						
3.2			■																					
5.1				■																				
5.2					■																			
3.4			X			■																		
3.5			X				■																	
5.3					X			■																
5.4					X				■															
5.5				X				X	X	■														
1.1											■	▨	▨	▨ X										
1.3	X										▨ X	■	▨	▨										
3.3			X								▨	▨ X	■	▨										
3.6		X				X	X				▨	▨	▨ X	■										
1.4												X			■									
2.1														X	X	■								
2.2																X	■							
2.3															X		X	■						
2.4																		X	■					
2.5																			X	■	▨	▨	▨ X	
4.1										X				X						▨ X	■	▨	▨	
4.2																				▨	▨ X	■	▨	
4.3																				▨	▨	▨	■	
4.4																							X	■

Figure 2.13: Solved DSM For Oil Field Development Project With Highlighted Blocks. (Made by Gunawan [15])

Tripathy and Eppinger [16] used a binary DSM, shown in Fig. 2.14, to help Dover Motion develop a process for creating air bearing technology. A “toll gate” was placed at the end of each block. The quality control of the tasks occurs at those points and not after the completion of each individual task.



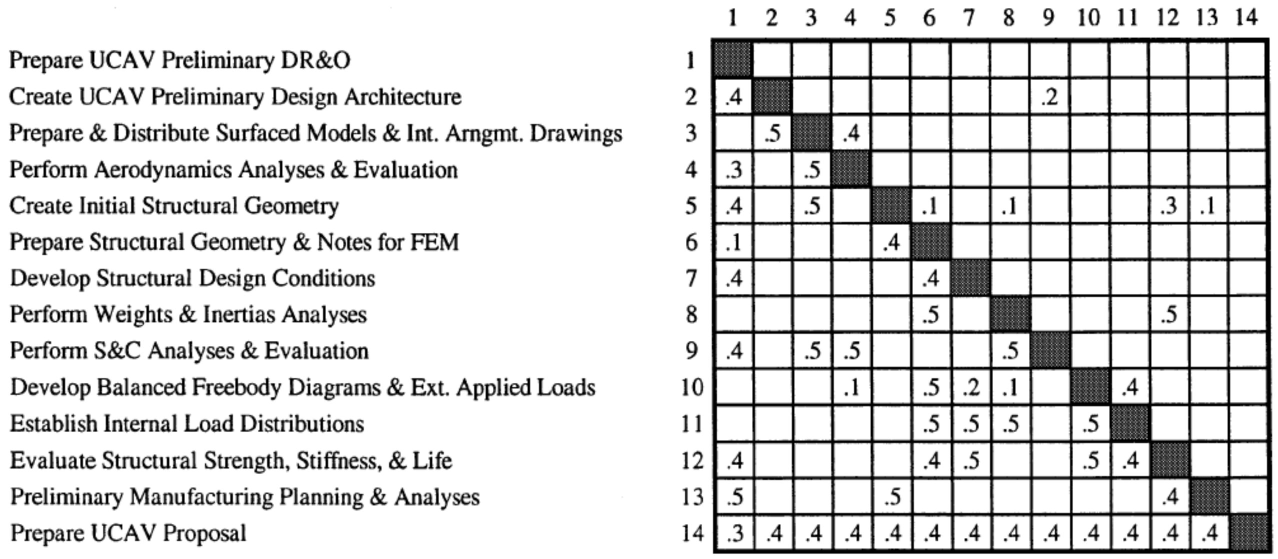
Tasks

Review customer request	Inhouse	■	x	x	x	x				
Develop product proposal and establish commercial proporsal (design concept and system architecture outlined)	Inhouse	x	■							
Finalize specifications and deliverables with customer	Inhouse		x	■						
Obtain Purchase Order from customer	Inhouse			x	■					
TOLL GATE # 1					x	■				
Allocation of resources for the project	Inhouse					x	■			
Brainstorming, review of concept, crystallization of product intent	Inhouse			x			x	■	x	
Concept freeze, which is bought in by the customer	Inhouse			x				x	■	
TOLL GATE # 2						x			x	■

Figure 2.14: Solved DSM From Dover Motion. (Made by Tripathy and Eppinger [16])

Non-binary DSMs use numbers to show the weight of interactions. Abdelsalam and Bao [17] used DSM weights that indicate the strength of the task interaction with one being extremely weak to nine being extremely strong. Browning and Eppinger [18] used non-binary DSMs that included the probability and impact of rework to make estimates of duration, cost, and risk of projects. An example is shown in Fig. 2.15 for the development of UCAVs (Unmanned Combat

Aerial Vehicles). If i is a DSM row and j is a DSM column, then element (i,j) is read as "the probability that task i has to be repeated because of an error in task j." An example from Fig. 2.15 is element (2,1) shows that Task 2 (Creating UCAV Preliminary Design Architecture) has a 40% chance of being reworked because of an error in Task 1(Preparing UCAV Preliminary DR&O). With this DSM, Browning and Eppinger made a joint PDF (Probability Density Function) of process duration and cost.



		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Prepare UCAV Preliminary DR&O	1	■													
Create UCAV Preliminary Design Architecture	2	.4	■							.2					
Prepare & Distribute Surfaced Models & Int. Arngmt. Drawings	3		.5	■	.4										
Perform Aerodynamics Analyses & Evaluation	4	.3		.5	■										
Create Initial Structural Geometry	5	.4		.5		■	.1		.1				.3	.1	
Prepare Structural Geometry & Notes for FEM	6	.1				.4	■								
Develop Structural Design Conditions	7	.4					.4	■							
Perform Weights & Inertias Analyses	8						.5		■				.5		
Perform S&C Analyses & Evaluation	9	.4		.5	.5				.5	■					
Develop Balanced Freebody Diagrams & Ext. Applied Loads	10				.1		.5	.2	.1		■	.4			
Establish Internal Load Distributions	11						.5	.5	.5		.5	■			
Evaluate Structural Strength, Stiffness, & Life	12	.4					.4	.5			.5	.4	■		
Preliminary Manufacturing Planning & Analyses	13	.5				.5							.4	■	
Prepare UCAV Proposal	14	.3	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	■

Figure 2.15: Browning And Eppinger's DSM Of Rework Probabilities. (Made by Browning and Eppinger [18])

2.3 Intersections of Obsolescence and DSM Research

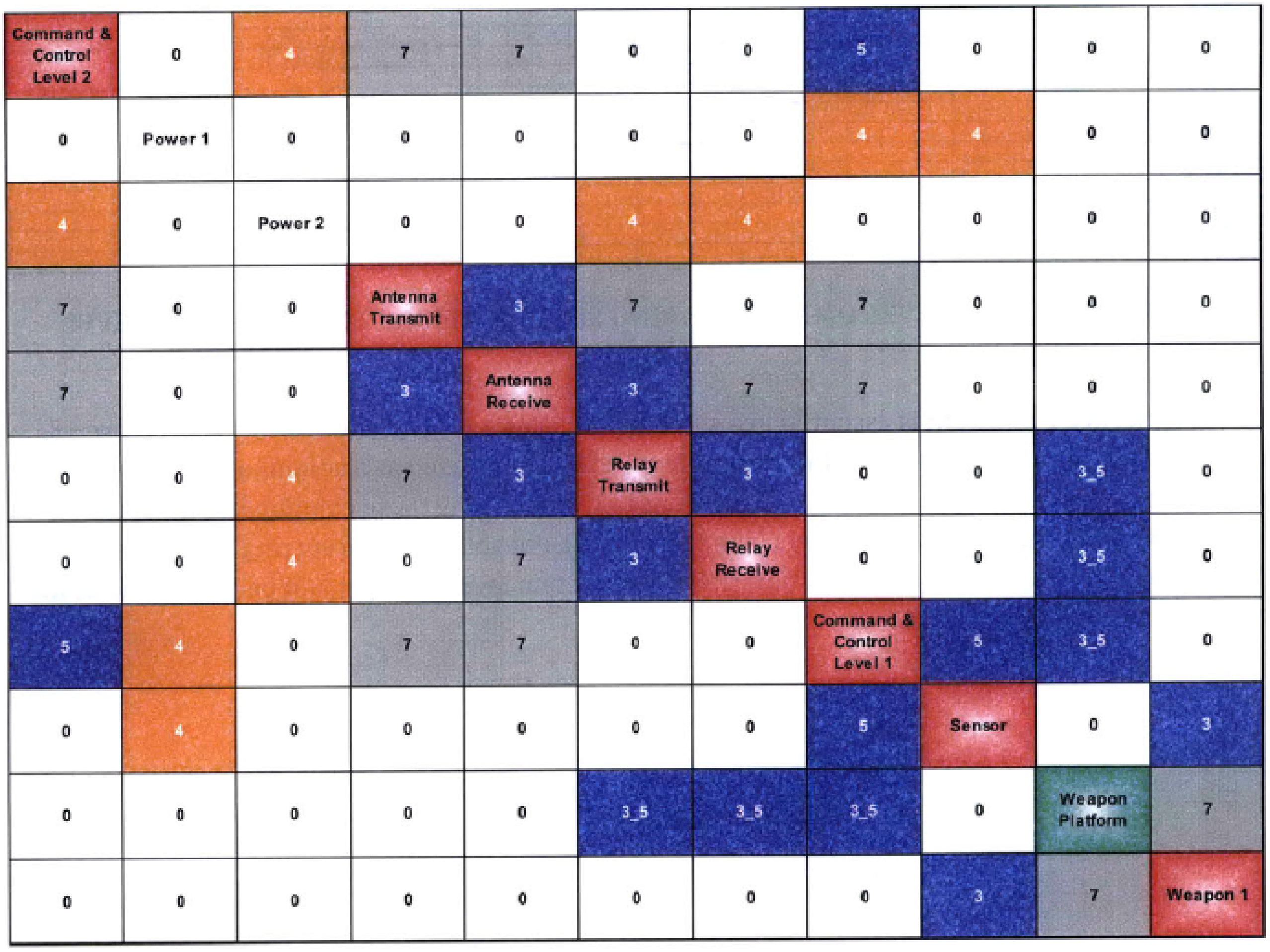
2.3.1 The Need To Consider System Interfaces

Khadke and Gershenson [19] discussed the importance of considering the role of technology evolution in the design of products and platforms that they make up to avoid frequent redesigns and premature obsolescence. Devereaux [20] worried about complex defense systems whose components have useful lives that are extremely short compared to the design life of the system that they are used for. Both Khadke and Gershenson [19] and Devereaux [20] have suggested using the DSM for system mapping.

2.3.2 Metrics For Describing Interfaces

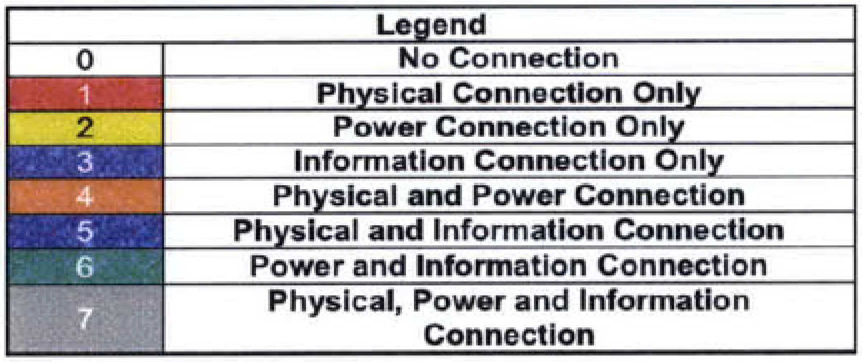
Khadke and Gershenson [19] decided to use three metrics to measure technology change. The first metric is called performance level, and it describes how well a given technology can perform based on its function. The second metric is called principle of operation and describes how the function is fulfilled. The third metric is technology architecture which is a basic description. Technology architecture is described in terms of its spatial, material, energy, and information needs. Next, two DSMs (one for internal dependency and another for external dependency)are filled out based on how likely of the technologies' metrics are to change based on numbers such as 5 (strong), 3 (medium), and 1 (weak) for weight, used to identify technologies in a platform that will cause problems later.

Devereaux [20] suggested using numbers and colors to fill DSMs. An example is shown in Figure 2.16 and the legend is shown in Fig. 2.17. Figure 2.18 shows the legend for the diagonal elements. The higher the element number, the more complex the interactions between subsystems are, and the more likely these elements become obsolete.



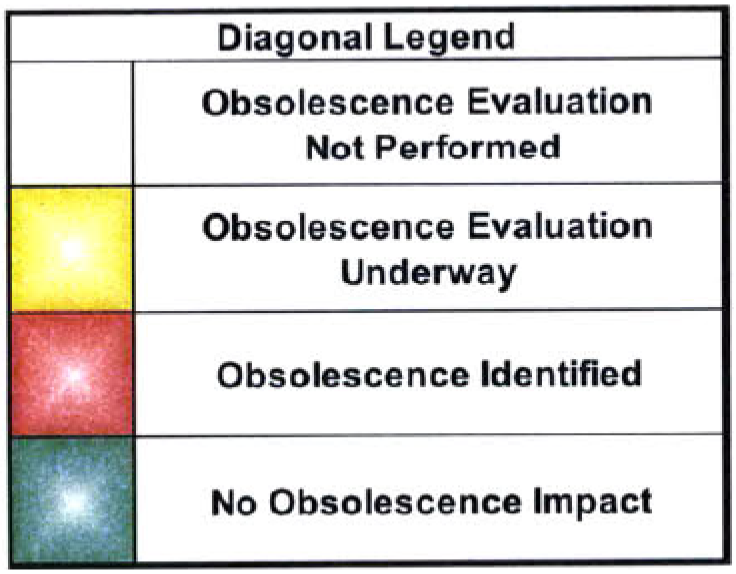
Command & Control Level 2	0	4	7	7	0	0	5	0	0	0
0	Power 1	0	0	0	0	0	4	4	0	0
4	0	Power 2	0	0	4	4	0	0	0	0
7	0	0	Antenna Transmit	3	7	0	7	0	0	0
7	0	0	3	Antenna Receive	3	7	7	0	0	0
0	0	4	7	3	Relay Transmit	3	0	0	3_5	0
0	0	4	0	7	3	Relay Receive	0	0	3_5	0
5	4	0	7	7	0	0	Command & Control Level 1	5	3_5	0
0	4	0	0	0	0	0	5	Sensor	0	3
0	0	0	0	0	3_5	3_5	3_5	0	Weapon Platform	7
0	0	0	0	0	0	0	0	3	7	Weapon 1

Figure 2.16: A DSM Considering Subsystem Strength. (Made by Devereaux [20])



Legend	
0	No Connection
1	Physical Connection Only
2	Power Connection Only
3	Information Connection Only
4	Physical and Power Connection
5	Physical and Information Connection
6	Power and Information Connection
7	Physical, Power and Information Connection

Figure 2.17: Legend For The DSM In Fig. 2.16. (Made by Devereaux [20])



Diagonal Legend	
⬜	Obsolescence Evaluation Not Performed
🟨	Obsolescence Evaluation Underway
🟥	Obsolescence Identified
🟦	No Obsolescence Impact

Figure 2.18: Diagonal Legend For The DSM In Fig. 2.16. (Made by Devereaux [20])

2.4 Research Gaps This Dissertation Will Address

2.4.1 Static Time Domain of Interface Study

There are many papers that assume that the nature of system interfaces stays the same as time passes. Browning and Eppinger [18] assumed that their DSM for probability of rework will stay constant as time passes. Khadke and Gershenson [19] and Devereaux [20] assumed that the obsolescence between system components is also constant. Nomaguchi et al. [21] used the growth curve and fuzzy numbers for making DSMs that are a function of time, but the growth curves were used to measure technical performance and task integrity. Also, the methodology of Nomaguchi et al. [21] only tracks process performance and not how a future process may appear to replace the current and obsolescent one.

2.4.2 Subjectivity of Interface Study

An issue with the papers in the previous section is that they both use DSMs that were assembled with expert opinion, but not all experts are the same. The second paper does not consider how the connection is made. An example of what the second paper misses is that there is more than one type of power connections and not all have equal influences or impacts.

The goal of this dissertation is to use the DSM for to keep track of interactions. The interactions will be made more objective by using growth curves whose parameters were obtained by examining the markets from which the equipment came that was used for completing the process and failure studies. The use of growth curves will make the DSMs not static in nature. The next chapter will explain the methodology used in this dissertation to create a non-static DSMs that will be used in modeling how obsolescence affects the design of a process's design as it ages.

CHAPTER 3

METHODOLOGY AND PROGRAM ARCHITECTURE

This chapter will discuss the methodology of using DSMs to determine how obsolescence affects the design of a process as time passes. A flowchart of the methodology is shown in Figure 3.1. Software was written for automation and is shown in Appendix A which will be referenced multiple times. The chapter is divided into four sections. The first section will discuss how the optimizer (computer codes shown in Appendices A.3 and A.4) works. Because the focus of this dissertation is on simulating the effects of obsolescence and not DSM optimization, the first section will be extremely brief. The second section will discuss what a loop master is, how to read one, and how to make one (codes shown in Appendix A.6). This input for the simulations is so important that it was given its own section. The next section will discuss the rest of the inputs needed and how to conduct the simulations themselves (using code shown in Appendix A.7). The last section will discuss how to determine if a process will become obsolescent during its life.

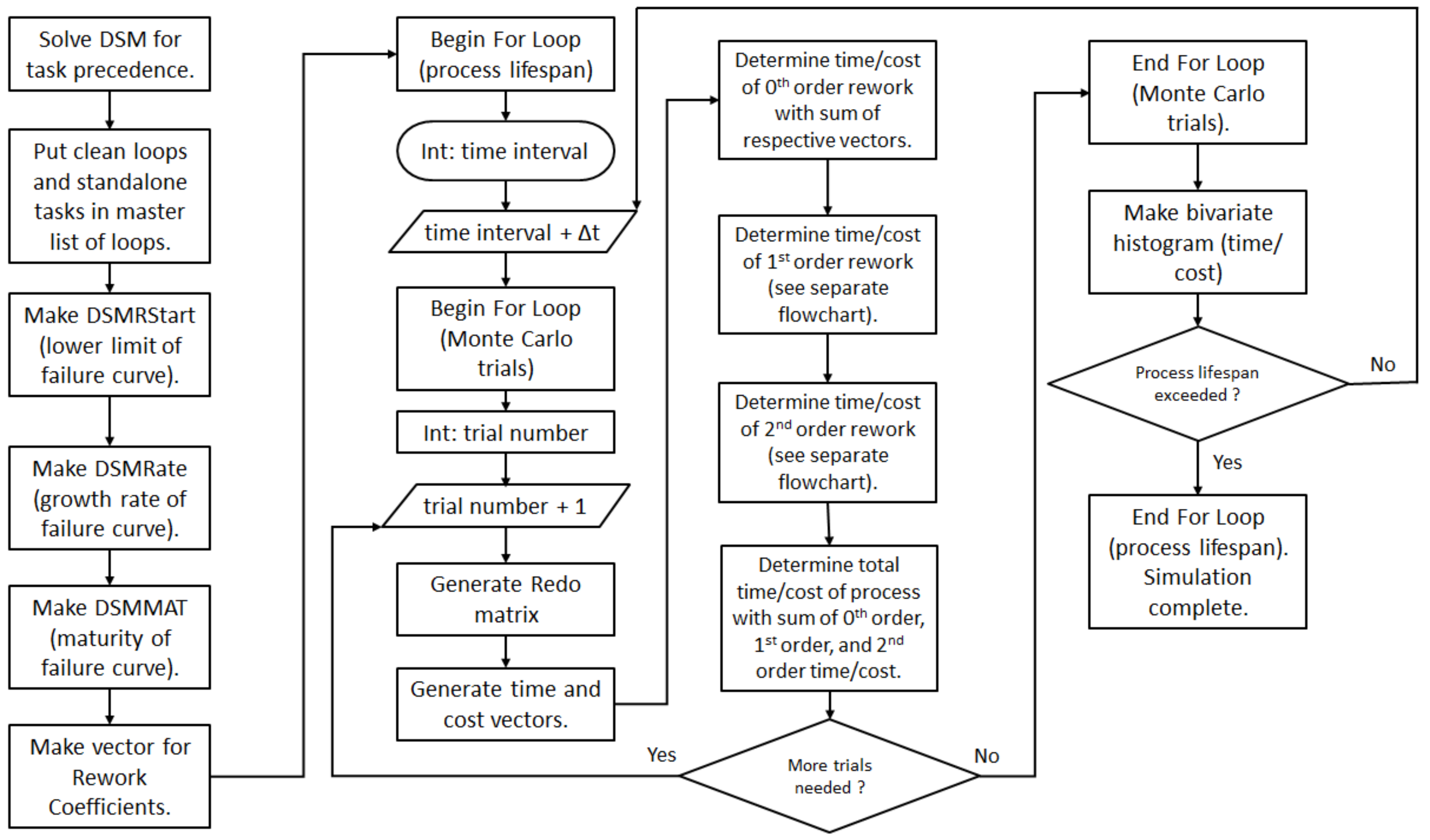

Solve DSM for task precedence.
Put clean loops and standalone tasks in master list of loops.
Make DSMRStart (lower limit of failure curve).
Make DSMRate (growth rate of failure curve).
Make DSMMAT (maturity of failure curve).
Make vector for Rework Coefficients.
Begin For Loop (process lifespan)
Int: time interval
time interval + Δt
Begin For Loop (Monte Carlo trials)
Int: trial number
trial number + 1
Generate Redo matrix
Generate time and cost vectors.
Determine time/cost of 0th order rework with sum of respective vectors.
Determine time/cost of 1st order rework (see separate flowchart).
Determine time/cost of 2nd order rework (see separate flowchart).
Determine total time/cost of process with sum of 0th order, 1st order, and 2nd order time/cost.
More trials needed ?
Yes
No
End For Loop (Monte Carlo trials).
Make bivariate histogram (time/ cost)
Process lifespan exceeded ?
No
Yes
End For Loop (process lifespan). Simulation complete.

Figure 3.1: Overall Flowchart Of Methodology.

3.1 DSM Optimization

3.1.1 Method of Optimization

The optimizer given in this dissertation uses a search method called differential evaluation. This method was preferred because it is simple to implement and does not need to use any differential calculus involved in methods such as the steepest decent method, linear programming, etc. Differential evaluation is a direct search method that uses multiple iterations for optimization. The purpose of the optimizer is to find an order of tasks that allows the process to be completed as smoothly as possible. To do this, the optimizer puts the interactions as close to the underside of the diagonal as possible. This will result in interactions being placed into sequential blocks of iteration like the solved binary DSM in Fig.3.2 by Gebala and Eppinger [22]. The first block is made of Tasks I, A, and J. The second block is made of Tasks F, K, and C. The third block is made of Tasks H, E, and D. The fourth block is made of Tasks L, G, and B. To complete the process, the tasks in the first block are repeated until their outputs are satisfactory. The tasks in the second block are worked on, then the third block, and then the final fourth block.

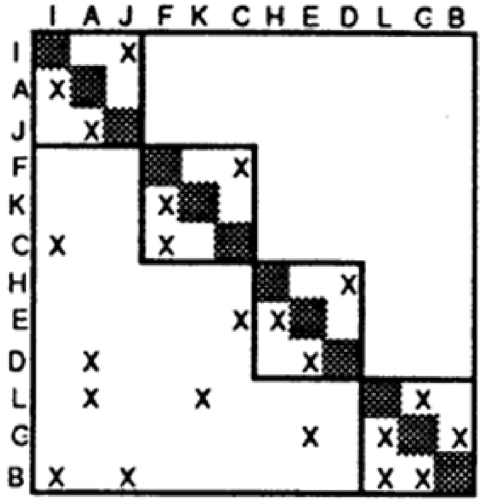


Figure 3.2: Example Of A Solved DSM With Blocks Of Iteration. (Made by Gebala and Eppinger [22])

An iteration of differential evaluation usually comes in four steps: 1) population initialization, 2) mutation, 3) crossover, 4) fitness evaluation. More details of how the optimizer will do these steps will be shown later in this section.

3.1.2 Extraction of Data From Excel

The data is processed in MatLab, but the program inputs are placed in Microsoft Excel. The file names for the inputs and outputs of the optimizer are located as specified in lines 22 and 23. The optimizer extracts the given Design Structure Matrix for Task Sorting (DSMTS) here.

3.1.3 Population Initialization

Equation (3.1) contains the objective function that the optimizer uses (see Appendix A.3) to reduce feedback. Each element in DSM is given a score, and the scores are added to determine the total score of the DSMTS.

$$Score_{total} = |\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i) * v_{(i,j)}| \tag{3.1}$$

where

i is the row,

j is the column,

n in the number of tasks in the process,

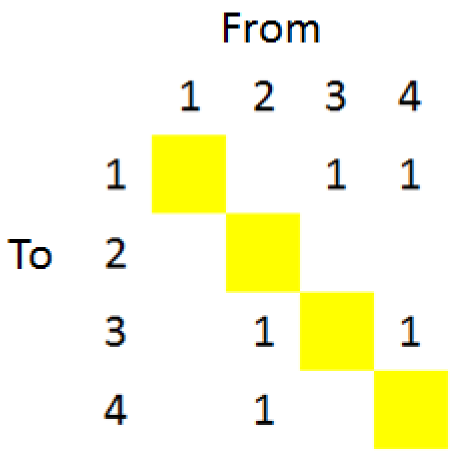
$v(i,j)$ is the value of element (i,j).

The first step that the optimizer does is to generate a set of candidate solutions (individuals) called a population. In differential evaluation, a chromosome is an arrangement of variables (alleles) that describes each individual. The optimizer in this dissertation creates individuals whose number of alleles is equal to twice the square of the number of tasks that make up the process. An example is that, if four tasks are needed to be completed, then the optimizer will make individuals with chromosomes that contain 32 alleles. 16 alleles control the i's of the element's values and 16 alleles control the j's of the element's values. Because the DSM heading for the rows must match the columns, the alleles are made such that no element locations are repeated. Table 3.1 shows how the chromosomes for the unsolved DSM in Fig. 3.3 are represented. Table 3.2 shows how the chromosomes for the solved DSM in Fig. 3.4 are represented.

Table 3.1: Table Explaining The Allele Values For The DSM In Fig. 3.3.

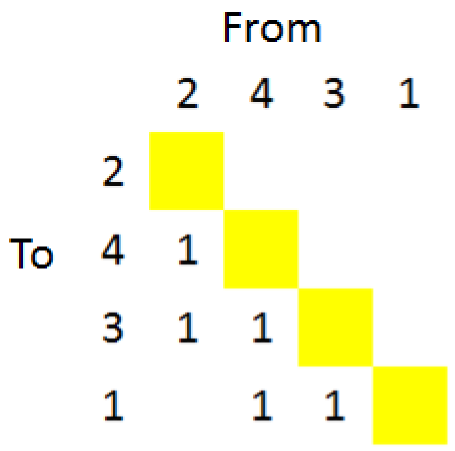
To Task	From Task	i	j	Value	Score
1	1	1	1	0	0
1	2	1	2	0	0
1	3	1	3	1	2
1	4	1	4	1	3
2	1	2	1	0	0
2	2	2	2	0	0
2	3	2	3	0	0
2	4	2	4	0	0
3	1	3	1	0	0
3	2	3	2	1	-1
3	3	3	3	0	0
3	4	3	4	1	1
4	1	4	1	0	0
4	2	4	2	1	-2
4	3	4	3	0	0
4	4	4	4	0	0
				Total Score	3

Note: The lower the score, the better.



		From			
		1	2	3	4
	1	🟨		1	1
To	2		🟨		
	3		1	🟨	1
	4		1		🟨

Figure 3.3: Resulting DSM From Allele Values In Table 3.1.



		From			
		2	4	3	1
	2	🟨			
To	4	1	🟨		
	3	1	1	🟨	
	1		1	1	🟨

Figure 3.4: Resulting DSM From Allele Values In Table 3.2.

Table 3.2: Table Explaining The Allele Values For The DSM In Fig. 3.4.

To Task	From Task	i	j	Value	Score
1	1	4	4	0	0
1	2	4	1	0	0
1	3	4	3	1	-1
1	4	4	2	1	-2
2	1	1	4	0	0
2	2	1	1	0	0
2	3	1	3	0	0
2	4	1	2	0	0
3	1	3	4	0	0
3	2	3	1	1	-2
3	3	3	3	0	0
3	4	3	2	1	-1
4	1	2	4	0	0
4	2	2	1	1	-1
4	3	2	3	0	0
4	4	2	2	0	0
				Total Score	-7

3.1.4 Mutation and Crossover

Mutation and crossover are steps that differential evaluation takes to make populations more diverse. Diverse populations are more likely to contain better solutions than those which are not diverse. Mutation is randomly rearranging the alleles in a single individual's chromosome. The optimizer in this dissertation (see lines 95 - 102 in Appendix A.4) does mutation by first generating a permutation of numbers. The vector (called "mutmaster") represents the order that in which the tasks are completed. Every n alleles in an individual's chromosome that controls DSM rows is made equal to the mutmaster component i. Every n alleles in an individual's chromosome that controls DSM columns are made equal to the mutmaster. The mutmaster from Table 3.1 is [1 2 3 4]. The mutmaster from Table 3.2 is [4 1 3 2].

Crossover is switching alleles from one individual to another. The optimizer in the dissertation does not do crossover because it will result in chromosomes becoming jumbled and after sorting them out, the new chromosomes will resemble another chromosome.

3.1.5 Fitness Evaluation

The next step in an iteration of differential evaluation is called fitness evaluation. In this step, the objective function of each individual is evaluated (determining score) and compared to the others. The more desirable and individual's trait is, the more "fit" it is. An example is that in an minimization problem, individuals with lower scores have higher fitness than individuals with higher scores.

Figure 3.5 shows how the optimizer given in this dissertation works in finding the individual(s) with the most fitness (see lines 109 to 176 in Appendix A.4). Equation (3.1) is used for looking for individuals whose interactions are as close to the diagonal as possible. A problem with using Eq. (3.1) alone is that it allows for the opportunity for interactions placed above the diagonal to be stuck there. To prevent this, another metric called the PBM (Percent of Backtracking Moves) is used. The PBM of a DSM can be determined by dividing the number of interactions above the diagonal by the total number of interactions. The optimizer will first separate the individuals from the population with the lowest scores called the elite (the second and third individuals). Then the members of the elite with the lowest PBMs are determined to have the most fitness (third individual).

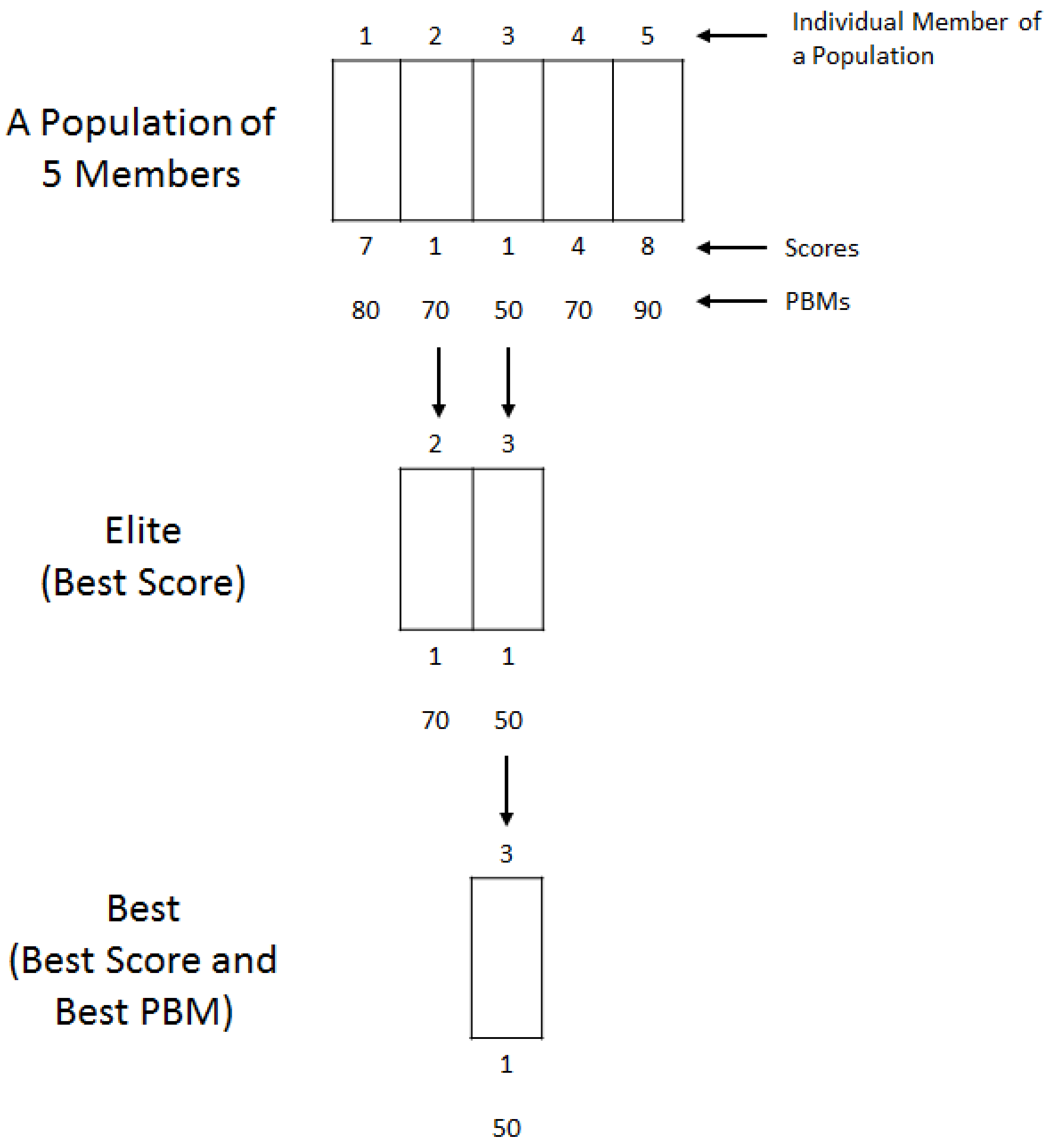

1 2 3 4 5
Individual Member of a Population
A Population of 5 Members
7 1 1 4 8
Scores
80 70 50 70 90
PBMs
2 3
Elite (Best Score)
1 1
70 50
3
Best (Best Score and Best PBM)
1
50

Figure 3.5: Example Of Optimizer's Determination Of Best Individuals.

The best of the population in the current iteration (generation) of the optimizer is compared to the best of the last generation. If the best of the current generation is more fit (having lower score and PBM) than the last generation, then the current generation is kept and is compared to the next generation. If the best of the last generation is more fit than the best in the current generation, then the last generation is kept and is compared to the next generation.

3.1.6 Special Case for Presolved DSMs

If one wishes to use the software to implement the methodology, but already has an optimized DSMTS, the code in Appendix A.5 can be used instead of using the optimizer. This code will take that DSM and determine the mutmaster which will be used by software. The DSMTS will be used to assemble the loop master.

3.2 Loop Master Characterization

3.2.1 Loop Master Notation

The loop master (LM) is a table that contains a list of loops and shows which tasks make them up. The purpose of the loop master is to help process designers determine which loops will become problems. The final loop master is m x n where m is the number of rows and n is the number of tasks in the process. The number of rows in the final loop master is one plus the number of "clean" loops. A clean loop is a loop that is not nested in another loop. If a loop is nested in another loop, it is called an "embedded" loop. There are three classes of loops in the loop master. The first class is for clean loops that contain three or more tasks. The second class is for clean loops that contain only two tasks (often called "coupled pairs"). The third class is for "standalone" tasks. Standalone tasks are tasks that not part of any loop.

Table 3.3 contains an example of a loop master that was made from the digraph in Fig.3.6. The loop (made up of Tasks 1, 2, and 3) in the first row is assigned a Class 1 loop because there are three more tasks and it is not embedded. Tasks 1 and 2 form a coupled pair, but they are not given a row in the loop master in Table 3.3. If the coupled pair was not nested in the loop made of Tasks 1, 2 and 3 , then it would have been given its own row and called a Class 2 loop. Task 4 is not part of any loop and therefore is a standalone task. It is placed in the row of the loop master for standalone tasks. The loop master is binary in nature. If an element (i,j) in the master list of loops row has a one, it means that loop i has task j.

Table 3.3: Loop Master For Digraph In Fig. 3.6.

Task ID		1	2	3	4
Class	1	1	1	1	0
	3	0	0	0	1

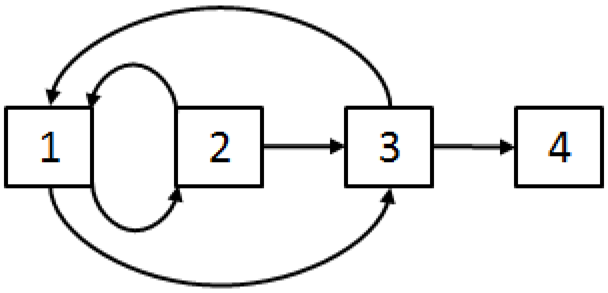


Figure 3.6: Digraph For Loop Master In Table 3.3.

Because the methodology presented in this dissertation assumes that the loops are completed sequentially, there is no need for the rows in the loop master to be in any order. The duration and cost of process as a whole will be the same regardless of the order of rework calculation of the loops. The software shown in Appendix A.6 places the row for standalone tasks at the bottom row of the loop master so that it can be easily found.

3.2.2 Finding Coupled Pairs

To ensure that all the loops in a process are found, the software in Appendix A.6 will use multiple methods to find loops. One method the software (see lines 20 to 57) uses to look for loops is to examine the DSMTS for pairs of coupled tasks. Tasks i and j make a coupled pair if and only if $DSMTS_{(i,j)}$ and $DSMTS_{(j,i)}$ are both not equal to zero.

3.2.3 Powers of Adjacency Matrix

Another method the software uses to look for loops is the powers of adjacency matrix method (see lines 58 through 96 in Appendix A.6). An adjacency matrix is made from the binary form of the DSMTS. Raising the adjacency matrix to the p^{th} power will show which tasks are p steps from each other. At first, the loop master's number of rows will be the same as the number of tasks. Each row of the loop master is coincident with the power of the adjacency matrix. For the software to know which tasks are in a loop, the software looks at each power of the adjacency matrix. If any diagonal elements in the adjacency matrix have any ones, then its is possible that those tasks representing those diagonals are in a loop. The matching row in the loop master is filled out accordingly. To keep each power of the adjacency matrix binary, the software follows the method by Gunawan [15]. What the software does is that after calculating each power, all non-zero elements are given a value of one. The software in this dissertation will calculate adjacency matrices to the n^{th} power because any higher powers are redundant. An example of this redundancy is that tasks in a five task loop are not just five steps apart, but 10, 15, 20, etc.

An example of using a power of an adjacency matrix is shown in Fig. 3.7. The adjacency matrix shown is of a second power. Ones in the diagonals show that Tasks 1 and 2 are in a loop. In the second row of the loop master, ones are placed in elements $LM_{(1,2)}$ and $LM_{(2,2)}$ to show that Tasks 1 and 2 are two steps apart.

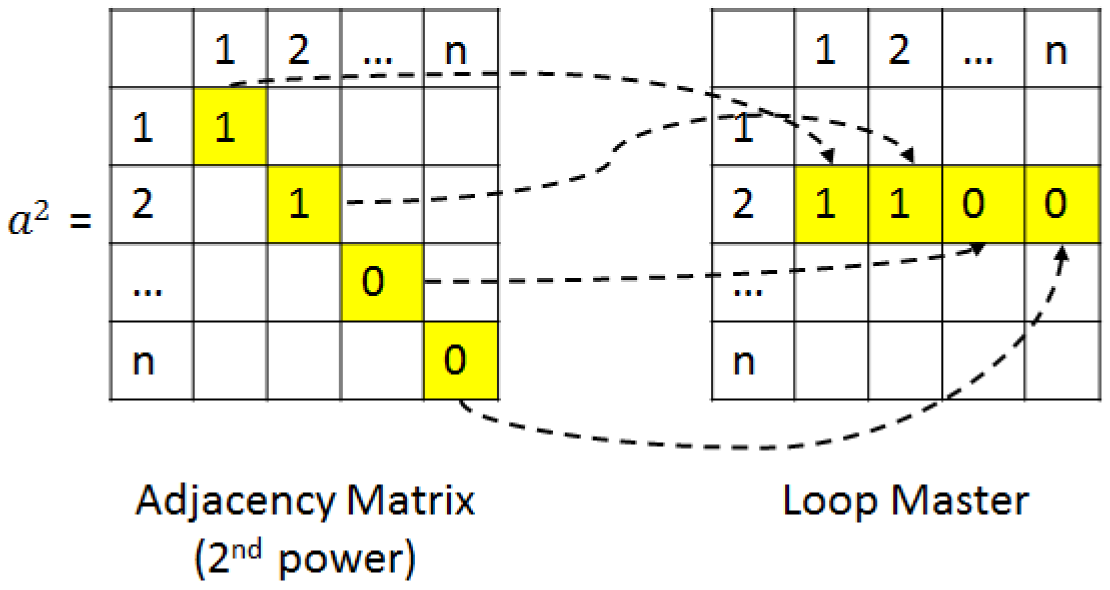


Figure 3.7: An Example Of Using An Adjacency Matrix To Fill A Loop Master.

3.2.4 Finding Standalone Tasks and Removing Connected Loops From Loop Master

At this point, standalone tasks can be found in the columns in the loop master that have a sum of zero (see lines 98 through 106 in Appendix A.6).

The powers of adjacency matrix method is not a perfect method of finding loops. Gunawan [15] and Maurer [23] noticed that if multiple loops exist, they can be entangled with each other. The extra steps in the software were written to remove the connected loops (see lines 108 through 159 of Appendix A.6). If a row in the loop master and the sum of its row are equal, then software will examine above the diagonal in the DSMTS. If element locations made by filled in columns of the loop master show no interaction in the DSMTS, loops of the same length exist and the row in question in the loop master is made all zeros. An example of this type of situation will be shown in the case study in the next chapter.

Another method the software uses for finding connected loops (see lines 98 through 106 of Appendix A.6) is looking for rows in the loop master whose sum is greater than its row number. These rows in the loop master will be made into all zeros. After this step in making the loop master, the coupled pairs are re-added to the loop master and duplicate rows are removed. Rows that have all zeros are also removed. Here, row number no longer matters.

3.2.5 Removing Embedded Loops From Loop Master

The software (see lines 190 through 220 in Appendix A.6) determines if a loop is embedded in another loop by adding their two loop master rows. If any components of the resulting vector is two or more, then the loop whose row sum is smaller loop is the embedded loop and the loop whose row sum is larger is the clean loop. All embedded loops are removed from the loop master.

3.2.6 Final Assembly and Exportation of Loop Master

Now the loop master row for standalone tasks is added to the loop master and the classes are assigned. The final loop master is exported to Excel for the software user to examine (see lines 222 though 278 in Appendix A.6).

3.3 Generation of Process Joint Duration/Cost Distribution Plots

3.3.1 Other Required Inputs and Preparation

There are more required inputs than the DSMTS and the loop master to make the distribution plots. Other inputs for the simulations are the duration and cost of each task in the form of a distribution. The software in this dissertation assumes that duration and cost of each task follow a triangular distribution. An example of a triangular distribution is shown in Fig. 3.8.

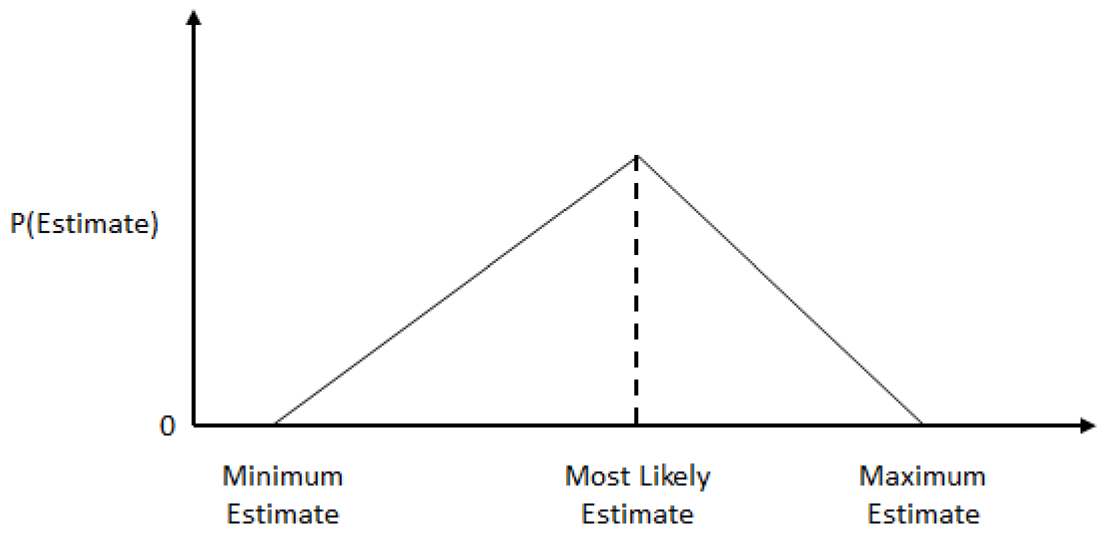


Figure 3.8: Triangular Distribution Example.

The next needed input is the Rework Coefficient (RC) of each task. An assumption of this methodology is that if a task has to be reworked then only parts of the task have to be redone. The RCs of the tasks are in terms of percentages. An example is, if a task needed an hour and 1000 USD (United States Dollars) to complete and the RC is 50, then half an hour (30 minutes) and 500 USD is spent each time the task has to be repeated. The RCs must be arranged into a vector (called RC) in which each component represents the rework to complete each task.

The growth curves that represent the probability of failure of the dominant devices used to complete each task are the next needed inputs. Figure 3.9 contains two examples of such failure curves. The failure curves are assumed to have the combined probability of failure from device mortality and device obsolescence and are also s-shaped. The parameters of the curves that are needed are the starting probabilities of the failure curves, the growth rate (controls slope of curve's center) and time of maturity (β). The maturity times are in reference to when the process is first used. The blue curve in Fig. 3.9 represents a technology that matured before the start of the process's life; therefore, β_1 would be a negative number. The red curve in Fig. 3.9 represents a technology that matured after the start of the process's life; therefore, β_2 would be a positive number.

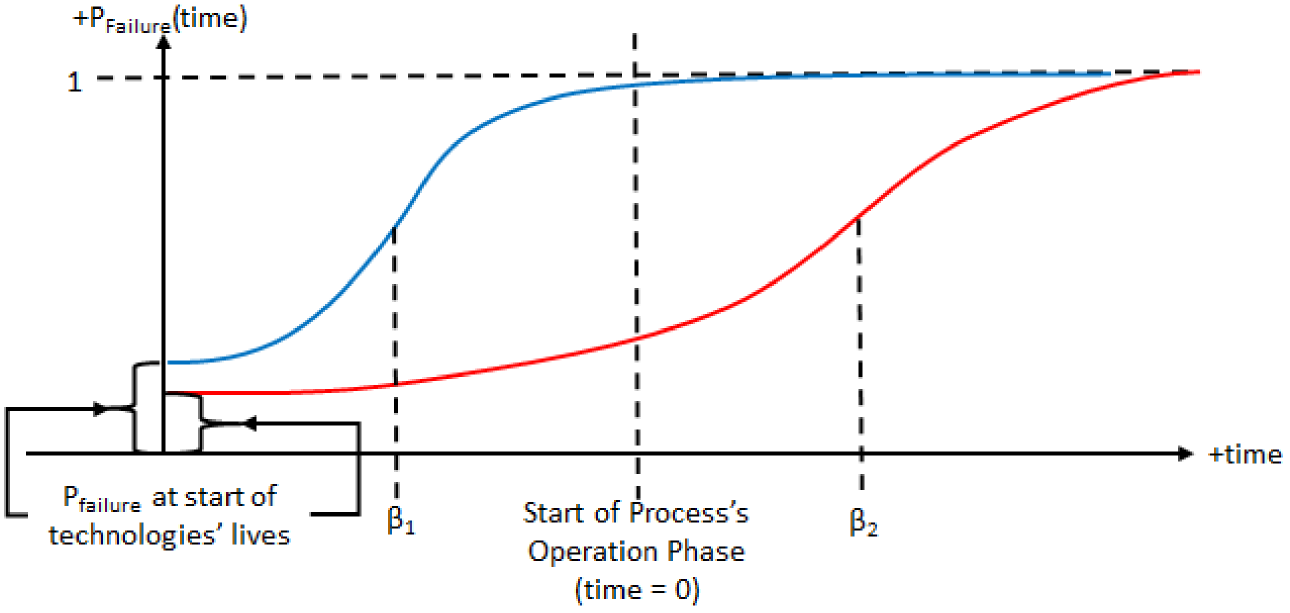


Figure 3.9: Examples Of Failure Curves.

The curve parameters are arranged into three DSMs. The first DSM is called the DSMRStart (Design Structure Matrix for Rework Starts). This DSM contains the starting probabilities of failure. An element in the DSMRStart, is read as "Task i depends on Task j and the connection's starting probability of failure is $DSMRStart_{(i,j)}$." The elements in the DSMRStart should be adjusted because the technology needed to complete a task might not interface the same with all its successors. The second DSM is called the DSMRate (Design Structure Matrix for Rates) and it is made of the growth rates. Reading across a task's row shows the rate of failure of technologies used to complete its predecessors. An element in the DSMRate is read as "Task i depends on Task j whose rate of failure is $DSMRate_{(i,j)}$." The third DSM is called the DSMMAT (Design Structure Matrix for MATurities) and it is made of the maturities. Reading across a task's row shows when the failure curves of the technologies used to complete its predecessors mature relative to the start of the use of the process. An element in the DSMMAT, is read as "Task i depends on Task j whose technology matures $DSMMAT_{(i,j)}$ before the start of the process."

Figure 3.10 shows how the software make the PDFs. The process resembles three nested loops. The process's lifespan is divided into intervals and each interval's PDF is made with a number of Monte Carlo trials. The software user needs to define the number of time intervals of interest and the number of Monte Carlo trials (see lines 3 through 11 in Appendix A.7).

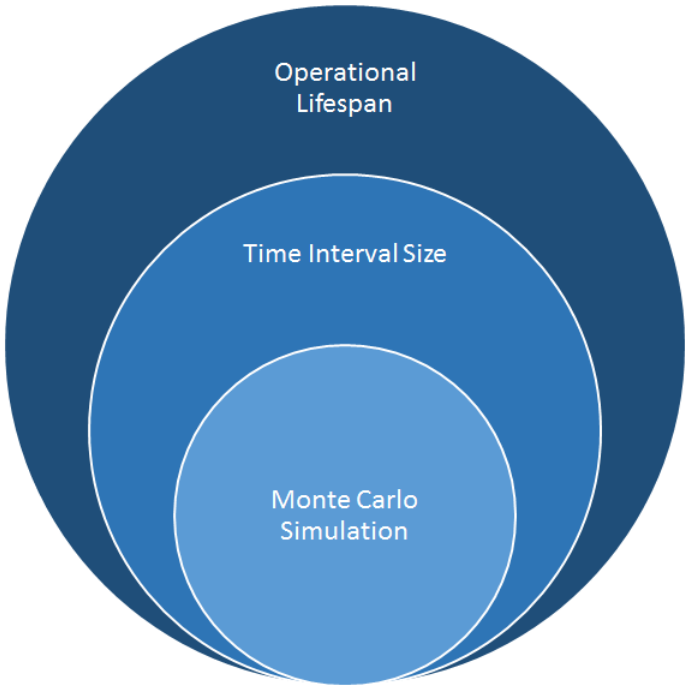


Figure 3.10: Illustration Of Loops In Methodology.

During each time interval, a $DSMR(t)$ (Design Structure Matrix for Rework at year t) is evaluated. It is read as "$DSMR(t)_{(i,j)}$ is the probability that Task i has to reworked because of a error in Task j." The assumption of rework caused by device failures is fundamental in this methodology. Equation (3.2) is used to fill in the $DSMR(t)$. It is more accurate to use this for determining the probability of rework than the DSMRStart because the failure curves of the technologies used to complete the tasks might not start at the same time (see lines 161 through 180 in Appendix A.7).

$$DSMR(t)_{(i,j)} = \frac{1 - DSMRStart_{(i,j)}}{1 + e^{-DSMRate_{(i,j)}(t - DSMMAT_{(i,j)})}} + DSMRStart_{(i,j)} \tag{3.2}$$

During each Monte Carlo trial, a vector made of the durations, T, and cost, C, is randomly generated from the given distributions of tasks time and cost in which each component represents task order of completion (see lines 190 through 209 in Appendix A.7). Also a matrix that is n x n called the Redo matrix is made. The Redo matrix is filled with randomly generated numbers from zero to one. In each Monte Carlo trial, the total duration and cost is split into three types of rework (0^{th} order, 1^{st} order, and 2^{nd} order) which will be explained in more detail later. The total duration and cost needed to complete the process is calculated with Eqs. (3.3) and (3.4).

$$T_{total} = T_0 + T_1 + T_2 \tag{3.3}$$

where

T_{total} is total duration of a process,

T_0 is duration of zero order rework,

T_1 is duration of first order rework (considered wasted time),

T_2 is duration of second order rework (considered wasted time).

$$C_{total} = C_0 + C_1 + C_2 \tag{3.4}$$

where

C_{total} is total cost of a process,

C_0 is cost of zero order rework,

C_1 is cost of first order rework (considered wasted money),

C_2 is cost of second order rework (considered wasted money).

3.3.2 Definition of 0th Order Rework

Zeroth order rework of a task is its first ever attempt to complete it. To determine the time and cost spent doing first order rework, simply add the components of the time and cost vectors as shown in Eqs. (3.5) and (3.6) respectively.

$$T_0 = \sum_{i=1}^{n} T(i) \tag{3.5}$$

where $T(i)$ is a component in the time vector.

$$C_0 = \sum_{i=1}^{n} C(i) \tag{3.6}$$

where $C(i)$ is a component in the cost vector.

3.3.3 Definition of 1st Order Rework

Figure 3.11 shows an incident of first order rework that can occur while completing the process in Fig. 2.10. What occurs in incidents of first order rework is that after a task is completed, its outputs are inspected and seen as not meeting specifications; then the task in question is repeated such that its outputs are to specifications. With regards to the situation in Fig. 3.11, the following events occurs.

1. Task A is completed.
2. The input specifications of Task C are compared to the outputs of Task A.
3. The outputs of Task A are determined to be not to specifications.
4. Task A is repeated.

5. Task C is completed.

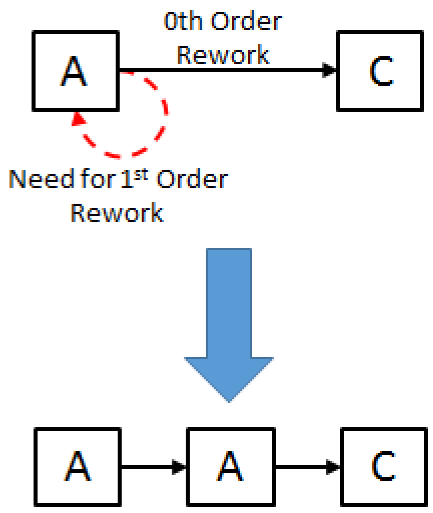


Figure 3.11: An Illustration Of A Situation Involving First Order Rework.

Figure 3.12 contains the flowchart that the software uses to calculate first order rework. To determine if any incidents of first order rework have occurred, the lower halves (where row number is greater than column number) of the $DSMR(t)$ for a given time and the Redo matrix are compared to each other. If an element in the Redo matrix, $Redo_{(r,c)}$ is lower than its corresponding element in the DSMR, $DSMR(t)_{(r,c)}$, an incidence of first order rework has occurred and the task associated with the element's column has to be repeated. To keep track of the number of times a task is repeated because of first order rework, a vector called the FOV (First Order Vector) is used. The vector component represents the task order and the component's value represents the number of times the task is repeated because of first order rework. An example of the FOV's notation is when $FOV(2)$ has a value of five means the second task in the process was repeated five times because of first order rework. To fill in the FOV, just take each component, $FOV(c)$ and determine the number of elements in the matching column, c in the Redo matrix is lower than the same column in the $DSMR(t)$ that are below the diagonal (see lines 224 through 232 in Appendix A.7). Equations (3.7) and (3.8) are used by the software to determine the time and cost spent on first order rework.

$$T_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} FOV(i)T(j)RC(k) \tag{3.7}$$

$$C_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} FOV(i)C(j)RC(k) \tag{3.8}$$

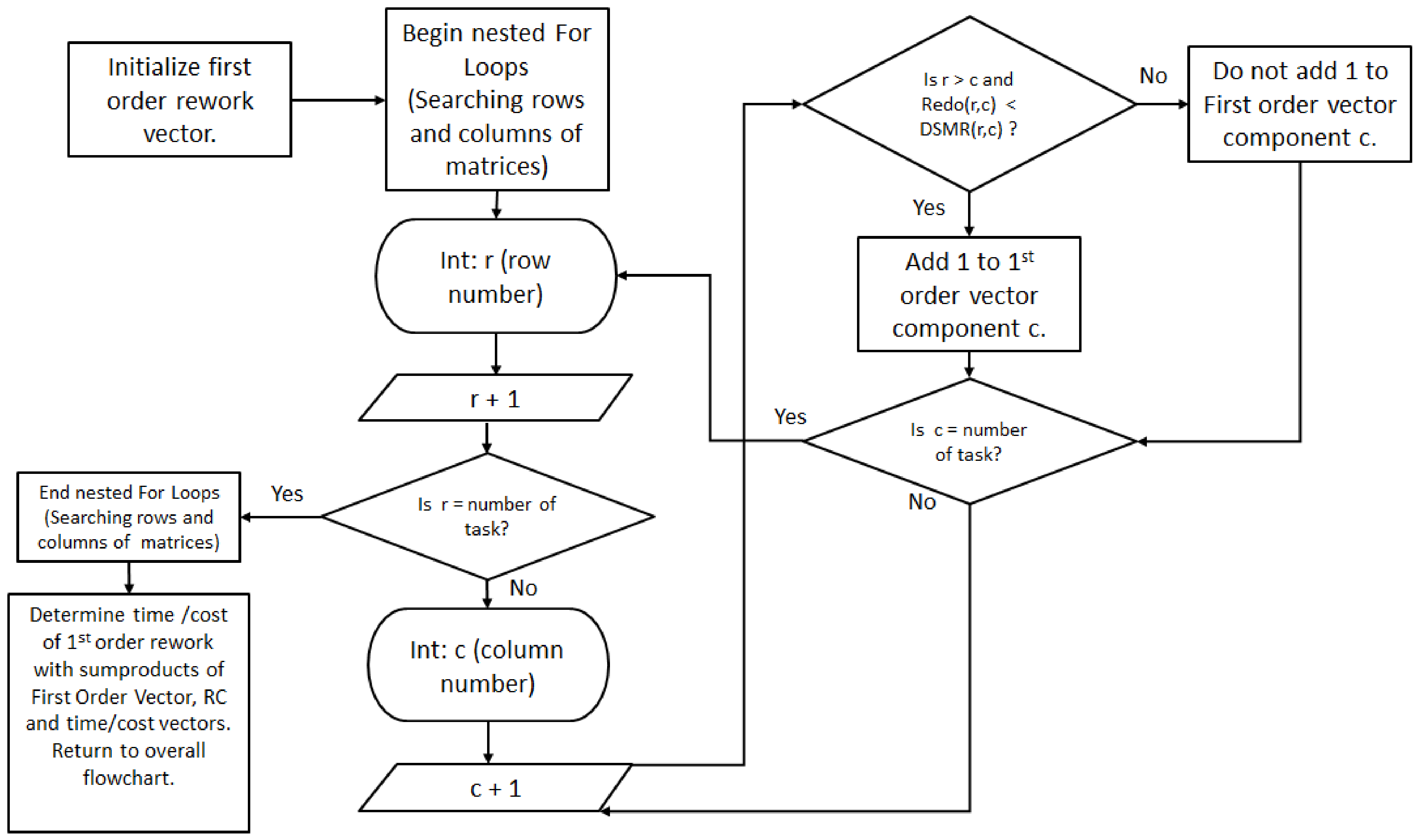

Initialize first order rework vector.
Begin nested For Loops (Searching rows and columns of matrices)
Int: r (row number)
r + 1
Is r = number of task?
Yes
End nested For Loops (Searching rows and columns of matrices)
Determine time /cost of 1st order rework with sumproducts of First Order Vector, RC and time/cost vectors. Return to overall flowchart.
No
Int: c (column number)
c + 1
Is r > c and Redo(r,c) < DSMR(r,c) ?
No
Do not add 1 to First order vector component c.
Yes
Add 1 to 1st order vector component c.
Is c = number of task?
Yes
No

Figure 3.12: Flowchart Of First Order Rework Calculation.

3.3.4 Definition of 2nd Order Rework

Figure 3.13 shows an incidence of second order rework that can occur while completing the process in Fig. 2.10. Second order rework is the worst kind of rework because it involves repeating entire sets of tasks rather than single tasks like in first order rework. What occurs in incidences of second order rework is that the outputs of the last task in a loop are not to specification, but the first task needs input from the last task and all the tasks in the loop have to be repeated so that the last task's output is to specifications. With regards to the situation in Fig. 3.13, the following events occurs.

1. Task E is completed.
2. Task A is completed.
3. Task C is completed.
4. The input specifications of Task F are compared to the outputs of Task C.
5. The outputs of Task C are determined to be not to specifications.
6. Those using the process notice that Task E need inputs from Task C for Task E to be completed correctly.
7. Task E is repeated.
8. Task A is repeated.
9. Task C is repeated.

An assumption of this methodology is that it is possible that PBMs (interactions above a diagonal in a DSMTS) in a process are not always needed but occurs with a certain probability. It is possible that the tasks in a loop can be completed correctly without any rework needed.

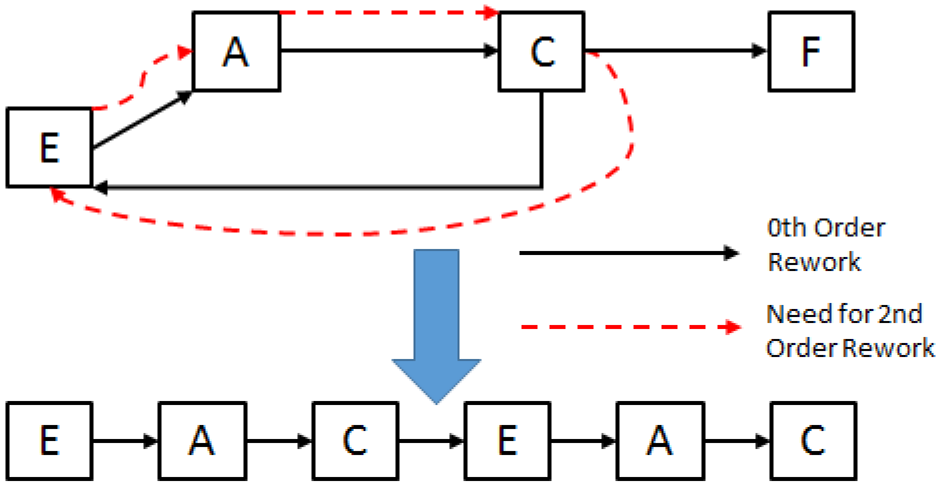


Figure 3.13: An Illustration Of A Situation Involving Second Order Rework.

Figure 3.14 contains the flowchart that the software uses to calculate second order rework. Eqs. (3.9) and (3.10) are used by the software to determine the time and cost spent on second order rework (see lines 237 through 248 in Appendix A.7). To determine if any incidences of second order rework has occurred, the upper halves (where row number is less than column number) of the $DSMR(t)$ for a given time and the Redo matrix are compared to each other. If an element in the Redo matrix, $Redo_{(r,c)}$ is lower than its corresponding element in the DSMR, $DSMR(t)_{(r,c)}$, then an incidence of second order rework has occurred and the task associated with the element's column has to be repeated. This is repeated for each row in the loop master other than the one for standalone tasks, hence the extra summation outside the brackets. While calculating the time and money spent on second order rework, only parts of the loop master rows, time vectors, cost vectors, and RC are needed. This makes the indexes of the sums go from r to c. An example of this is if the loop in Fig. 2.10 has to be reworked, then the index for l is from one to one (the loop master would have two rows, but only one is needed) and the other indexes would be from three to five. The terms with the *embedded* subscript ($T_{embedded}$ and $C_{embedded}$) represent the second order rework of the embedded loops of each loop master row. The index for those summations is for the number of embedded loops each clean loop contains, b.

$$T_2 = \sum_{l=1}^{m-1} [\sum_{i=r}^{c} \sum_{j=r}^{c} \sum_{k=r}^{c} LM(l,i)T(j)RC(k) + \sum_{a=1}^{b} T_{embedded}] \tag{3.9}$$

$$C_2 = \sum_{l=1}^{m-1} [\sum_{i=r}^{c} \sum_{j=r}^{c} \sum_{k=r}^{c} LM(l,i)C(j)RC(k) + \sum_{a=1}^{b} C_{embedded}] \tag{3.10}$$

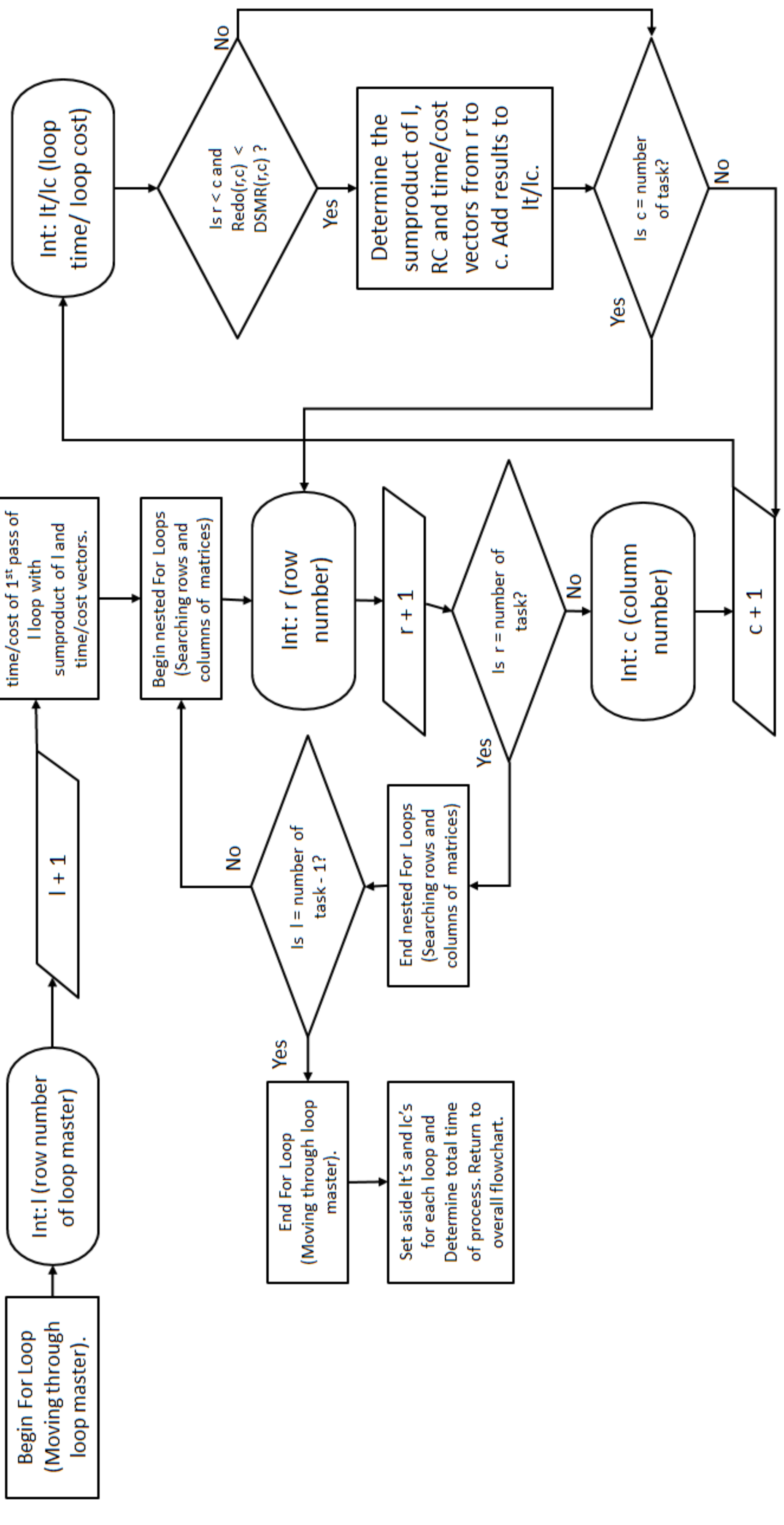

Begin For Loop (Moving through loop master).
Int: l (row number of loop master)
l + 1
Determine the time/cost of 1st pass of l loop with sumproduct of l and time/cost vectors.
Begin nested For Loops (Searching rows and columns of matrices)
Int: r (row number)
r + 1
Is r = number of task?
No
Yes
Int: c (column number)
c + 1
Int: lt/lc (loop time/ loop cost)
Is r < c and Redo(r,c) < DSMR(r,c) ?
Yes
No
Determine the sumproduct of l, RC and time/cost vectors from r to c. Add results to lt/lc.
Is c = number of task?
Yes
No
End nested For Loops (Searching rows and columns of matrices)
Is l = number of task - 1?
No
Yes
End For Loop (Moving through loop master).
Set aside lt's and lc's for each loop and Determine total time of process. Return to overall flowchart.

Figure 3.14: Flowchart Of Second Order Rework Calculation.

3.4 Checking If Process Design Becomes Obsolescent During Its Life

As time passes in the simulation, the distribution of the process's duration and cost will move and change shape. If an unwanted majority of the PDF moves outside predetermined duration and/or cost limits, then the process design will become obsolescent during its life and the process should be redesigned.

In the next chapter, a case study will be provided. The case study will first discuss the need for preparations to conduct the simulations and then how to do the simulations themselves.

CHAPTER 4

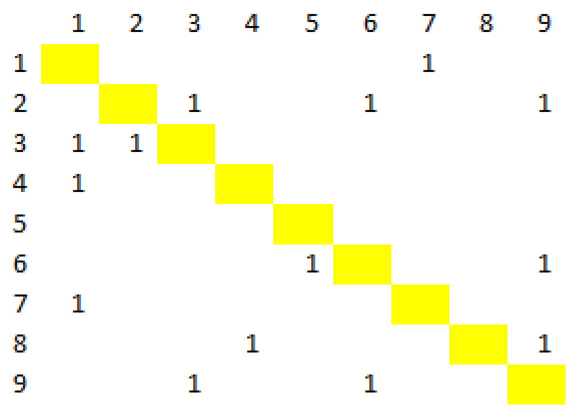
CASE STUDY AND PROGRAM DEMONSTRATION

For a better understanding of how to implement the methodology to a process and determine whether or not the process's design will become obsolescent during its operational life, a case study will be used. The case study was first started by Rogers [26] where only an optimization considering task sorting was discussed. Later, Obeng [27] used the DSM used by Rogers's case study, assigned properties to the task, and implemented the methodology discussed in the previous chapter with success. The purpose of the case study in this dissertation is to provide a more precise discussion of the case study Obeng [27] made and add some extensions. To be able to implement the methodology, some prior preparation is needed. The first section (Section 4.1)will explain how the task order was optimized with the software presented in the appendix. The second section (Section 4.2) will discuss how the loop master was assembled for the case study. In the third section (Section 4.3), a discussion of how the methodology was implemented to create a single data point for a simulation. This section will show the task information needed to conduct the simulations and the Appendix will show how these are made into inputs for the software made to automate the simulations. The fourth section (Section 4.4)will also show what happens if obsolescence is included and neglected. The fifth section (Section 4.5) will contain a sensitivity analysis. The last section (Section 4.6) will discuss the results of the case study.

4.1 DSM Optimization For Case Study

4.1.1 DSMTS Before Optimization

Figure 4.1 was made by Rogers [26] and contains the DSMTS before it was optimized. Figure A.1 shows how the DSM in Fig. 4.1 is entered into Microsoft Excel so that the software can conduct the optimization, assemble the loop master, and do the simulations.



	1	2	3	4	5	6	7	8	9
1	🟨						1		
2		🟨	1			1			1
3	1	1	🟨						
4	1			🟨					
5					🟨				
6					1	🟨			1
7	1						🟨		
8				1				🟨	1
9			1			1			🟨

Figure 4.1: Unsolved DSMTS For Case Study. (Made by Rogers [26])

4.1.2 DSMTS After Optimization

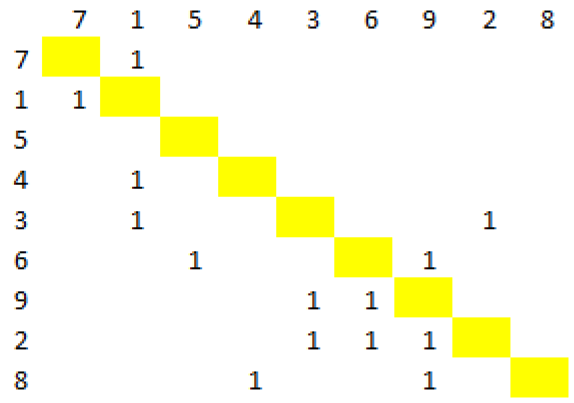
Figure 4.2 was made by Obeng [27] and contains the DSMTS after it was optimized and the task order was found acceptable. Because the given optimizer does not always converge to solution given the number of generations allowed to calculate and/or population size, the optimized DSMTS has to be examined and found acceptable before anything can be done. The case study used 100 generations and a population size of 100 individuals The following criteria can be used in determining if the process order is acceptable.

1. Most starting tasks (tasks that need no inputs) are within the first 25% of the sequence. For example, if a process has 100 tasks, then this task order is not rejected if most starting tasks are within the first 25 slots.

2. Most ending tasks (tasks that give no outputs) are within the last 25% of the sequence. For example, if a process has 100 tasks, then this task order is not rejected if most ending tasks are within the last 25 slots.

The tasks are in an optimized order if both of these criteria are satisfied. Increasing the population size and number of generations can make finding solutions easier if acceptable DSMTS cannot be found. After optimizing the DSMTS for the case study, the score went down from

48 to 30 and the PBM went down from 42.9% to 21.4%. The software outputs many metrics about the process to the Excel file that has the simulation inputs.



	7	1	5	4	3	6	9	2	8
7	🟨	1							
1	1	🟨							
5			🟨						
4		1		🟨					
3		1			🟨			1	
6			1			🟨	1		
9					1	1	🟨		
2					1	1	1	🟨	
8				1			1		🟨

Figure 4.2: Solved DSMTS For Case Study. (Made by Obeng [27])

4.2 Creation of Loop Master For Case Study

4.2.1 Looking For Coupled Pairs For Case Study

The process in the case study has three coupled pairs. They are made of Task 7 and 1 (because $DSMTS_{(1,2)}$ and $DSMTS_{(2,1)}$ both nonzero), Tasks 3 and 2 (because $DSMTS_{(5,8)}$ and $DSMTS_{(8,5)}$ both nonzero), and Tasks 6 and 9 (because $DSMTS_{(6,7)}$ and $DSMTS_{(7,6)}$ both nonzero).

4.2.2 Filling Loop Master Powers of Adjacency Matrix For Case Study

Table 4.1 contains the loop master for this case study after using the powers of adjacency matrix method. As shown, the loop master has not yet been organized. Each row represents a power of the adjacency matrix and the elements represent the power's diagonal. For example, the second power of the adjacency matrix have ones in the diagonals for Tasks 7, 1, 3, 9, 6, and

2 implying that they are two steps from each other.

Table 4.1: Loop Master Filled Out After Using Powers Of Adjacency Matrix Method In Case Study.

Power	7	1	5	4	3	6	9	2	8
1	0	0	0	0	0	0	0	0	0
2	1	1	0	0	1	1	1	1	0
3	0	0	0	0	1	0	1	1	0
4	1	1	0	0	1	1	1	1	0
5	0	0	0	0	1	1	1	1	0
6	1	1	0	0	1	1	1	1	0
7	0	0	0	0	1	1	1	1	0
8	1	1	0	0	1	1	1	1	0
9	0	0	0	0	1	1	1	1	0

4.2.3 Finding Standalone Tasks and Correcting Loop Master For Case Study

Based on the sums of the columns in the loop master in Table 4.1 Tasks 5, 4, and 8 are standalone tasks because they are not part of any loop.

Because the powers of the adjacency matrix method are not perfect, some extra steps have to be taken to ensure that the final edition of the loop master is correct. The first step is to look for rows whose sum is the same as its power. In this case, it is the sixth row. Here, DSMTS locations are made with the column numbers of first and last non zero columns. By looking at $DSMR_{(1,7)}$ and noticing that it is filled with a zero, the sixth row has connected loops and not one loop that has six tasks. That corresponding row and its duplicates in higher powers (rows four and eight) are removed by filling it with zeros.

The second step taken in correcting the loop master is removing rows whose sum is higher than the row's power. Here, the second row is removed for that reason. It has the coupled pairs entangled. After this point, row number does not represent power of the adjacency matrix. Here, extra rows representing coupled pairs are added to ensure that they are not skipped before determining which loops are embedded. Table 4.2 contains the loop master for the case study after the connected loops are removed and the coupled pairs are added.

Table 4.2: Case Study Loop Master Filled Out After Connected And Duplicate Loops Are Removed.

Row	7	1	5	4	3	6	9	2	8
1	0	0	0	0	1	0	1	1	0
2	0	0	0	0	1	1	1	1	0
3	1	1	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	1	0
5	0	0	0	0	0	1	1	0	0

4.2.4 Finding Embedded Loops For Case Study

To determine which loops are embedded, each row in the loop master is added to every other. If the resulting vector contains any twos, then one of an embedded loop has been found. The loop with few tasks is the embedded loop. Such a situation in the case study is shown in Table 4.3. The first and second rows of the loop master were added. By the twos in the last row in Table 4.3 representing Tasks 3, 9, and 2, those tasks are embedded in a loop that is made of Tasks 3, 6, 9, and 2. The embedded loop is removed to prevent double counting when calculating second order rework.

Table 4.3: A Found Embedded Loop For Case Study.

Row	7	1	5	4	3	6	9	2	8
1	0	0	0	0	1	0	1	1	0
2	0	0	0	0	1	1	1	1	0
Sum	0	0	0	0	2	1	2	2	0

Table 4.4 shows two loops that are not embedded. The second and third rows were added together. These loops were found to be not connected because the resulting vector does not have any twos.

Table 4.4: Two Loops That Are Not Embedded In Case Study.

Row	7	1	5	4	3	6	9	2	8
2	0	0	0	0	1	1	1	1	0
3	1	1	0	0	0	0	0	0	0
Sum	1	1	0	0	1	1	1	1	0

4.2.5 Final Assembly of Loop Master For Case Study

After taking the loop master from Table 4.4, adding a row for the standalone tasks, and assigning classes to the loops, the final loop master for the case study by Obeng [27] is obtained and shown in Table 4.5. Figure 4.3 contains the digraph for this case study with the clean loops shown.

Table 4.5: Completed Loop Master For Case Study. (Made by Obeng [27])

Task ID		7	1	5	4	3	6	9	2	8
	1	0	0	0	0	1	1	1	1	0
Class	2	1	1	0	0	0	0	0	0	0
	3	0	0	1	1	0	0	0	0	1

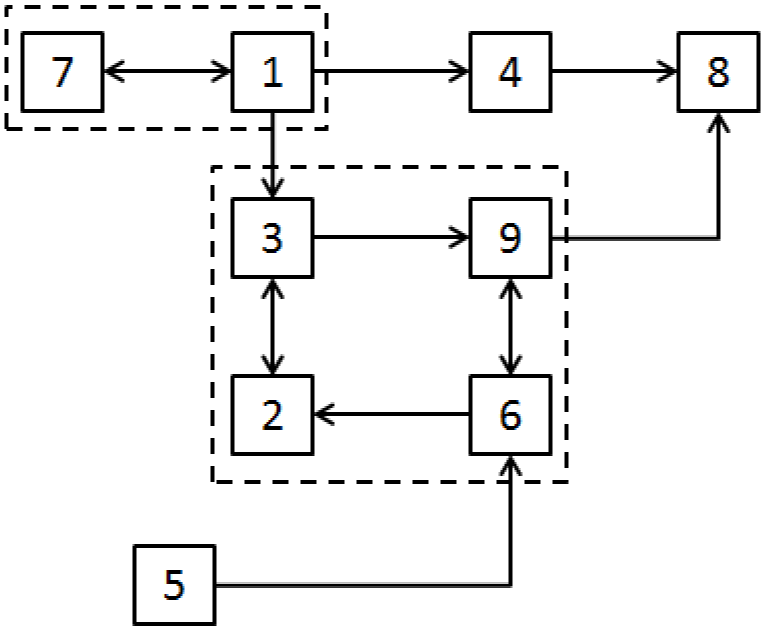


Figure 4.3: Digraph For Case Study With Clean Loops Shown.

4.3 Example of Data Set Creation For Case Study

4.3.1 Task Information and Preparation For Case Study

Tables 4.6 and 4.7 were made by Obeng [27] and contain the duration and cost of each of the tasks respectively. The joint PDFs for the case study will have the total process duration in years.

Table 4.6: Duration Of Individual Tasks In Years. (Made by Obeng [27])

Task	Min(Years)	Expected (Years)	Max (Years)
1	0.074763	0.07668	0.078597
2	0.11502	0.15336	0.17253
3	0.13419	0.1917	0.28755
4	0.01917	0.05751	0.061344
5	0.1917	0.23004	0.30672
6	0.09585	0.11502	0.13419
7	0.017253	0.01917	0.028755
8	0.13419	0.17253	0.193617
9	0.01917	0.03834	0.07668

Table 4.7: Cost Of Individual Tasks In 1000's OF USD. (Made by Obeng [27])

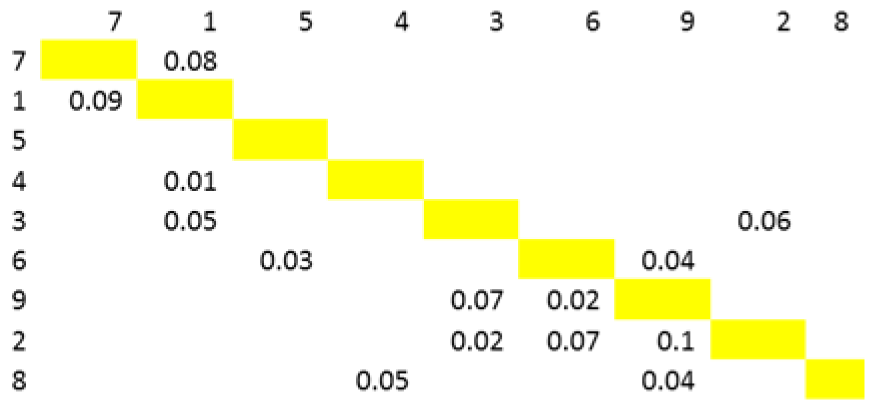
Task	Min	Expected	Max
1	30	40	50
2	50	100	101
3	10	20	22
4	114	150	177
5	25	30	36
6	40	55	90
7	9	10	12
8	25	35	40
9	15	19	20

Table 4.8 was made by Obeng [27] and contains the rework coefficients, growth rate of failure curves and maturity time of tasks in the case study. These parameters are assumed as such in our case study.

Table 4.8: RC'S, Technology Rates, And Maturities Of Tasks For Case Study. (Made by Obeng [27])

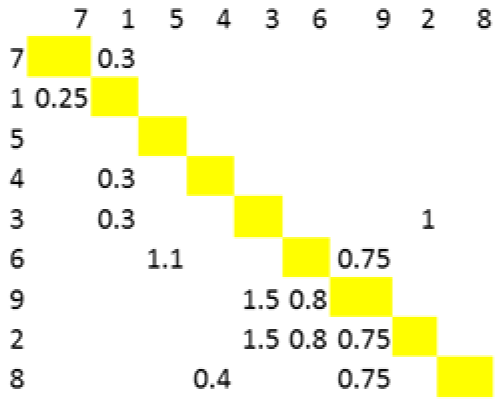
Task	RC	Rate	Maturity (Years)
1	20	0.3	10
2	50	1	18
3	80	1.5	15
4	100	0.4	-10
5	10	1.1	5
6	25	0.8	6
7	78	0.25	-2
8	6	1.8	50
9	10	0.75	1

Figures 4.4, 4.5, 4.6 were also made by Obeng [27] contains the DSMRtart, DSMRate, and DSMMAT respectively for the case study.



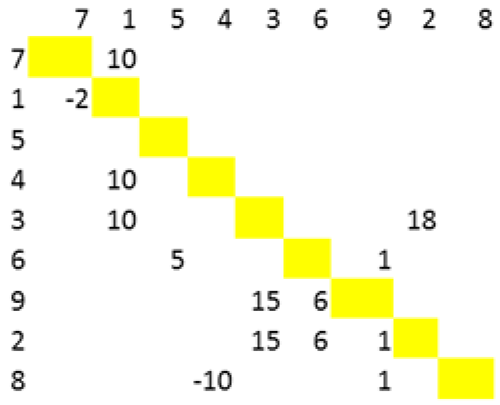
	7	1	5	4	3	6	9	2	8
7	🟨	0.08							
1	0.09	🟨							
5			🟨						
4		0.01		🟨					
3		0.05			🟨			0.06	
6			0.03			🟨	0.04		
9					0.07	0.02	🟨		
2					0.02	0.07	0.1	🟨	
8				0.05			0.04		🟨

Figure 4.4: DSMRStart For Case Study. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	0.3							
1	0.25	🟨							
5			🟨						
4		0.3		🟨					
3		0.3			🟨			1	
6			1.1			🟨	0.75		
9					1.5	0.8	🟨		
2					1.5	0.8	0.75	🟨	
8				0.4			0.75		🟨

Figure 4.5: DSMRate For Case Study. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	10							
1	-2	🟨							
5			🟨						
4		10		🟨					
3		10			🟨			18	
6			5			🟨	1		
9					15	6	🟨		
2					15	6	1	🟨	
8				-10			1		🟨

Figure 4.6: DSMMAT For Case Study. (Made by Obeng [27])

Table 4.9 was made by Obeng [27] and contains the time and cost vectors that will be used for this case study. When determining the time and cost of the process, the components of these vectors and the RC shown in Table 4.10 must be arranged to match the order of tasks in the optimized DSMTS.

Table 4.9: Time And Cost Vectors With Labels For Case Study. (Made by Obeng [27])

Task Order	1	2	3	4	5	6	7	8	9
Task Name	7	1	5	4	3	6	9	2	8
Time Vector Component, $T(j)$	0.02	0.077	0.23	0.03	0.25	0.11	0.05	0.115	0.07
Cost Vector Component, $C(j)$	11	33	27	140	21	55	18	75	30

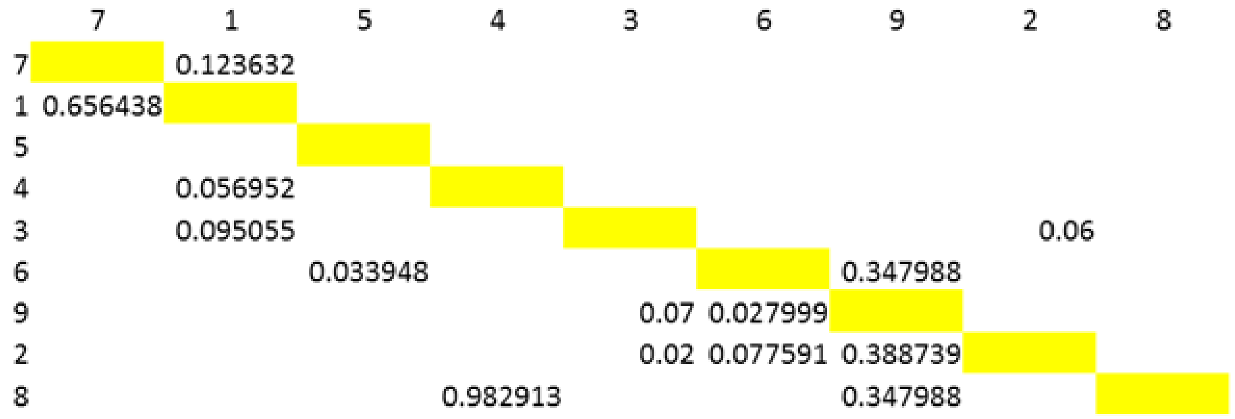
Table 4.10: Rework Coefficient Vector With Labels For Case Study. (Made by Obeng [27])

Task Order	1	2	3	4	5	6	7	8	9
Task Name	7	1	5	4	3	6	9	2	8
Rework Coefficients Vector Component, $RC(k)$ (%)	78	20	10	100	80	25	10	50	6

The data set will be calculated in this case study at the start of the process's life (at year zero). The DSMR(0) (Design Structure Matrix for Rework at year zero) has to be determined. It is more accurate to use this to determine the probability of rework than the DSMRStart because all the technologies used to complete the tasks might not start at the same time. To determine DSMR(0), Eq. (3.2) is evaluated at zero for all non-zero elements and those that are not on the diagonal. Equations (4.1) and (4.2) show a sample calculation for the probability of rework of Task 1 because of mistakes in Task 7 at start of operational life. Figure 4.7 made by Obeng [27] has the entire DSMR(0) that will be used for the case study.

$$DSMR(0)_{(2,1)} = \frac{1 - DSMRStart_{(2,1)}}{1 + e^{-DSMRate_{(2,1)}(0 - DSMMAT_{(2,1)})}} + DSMRStart_{(2,1)} \tag{4.1}$$

$$0.656 = \frac{1 - 0.09}{1 + e^{-0.25(0 - -2)}} + 0.09 \tag{4.2}$$



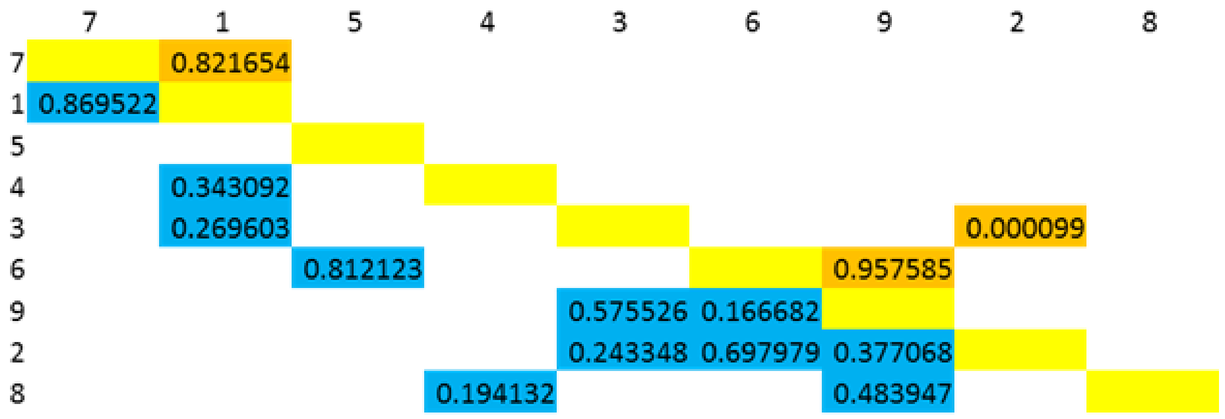
	7	1	5	4	3	6	9	2	8
7	🟨	0.123632							
1	0.656438	🟨							
5			🟨						
4		0.056952		🟨					
3		0.095055			🟨			0.06	
6			0.033948			🟨	0.347988		
9					0.07	0.027999	🟨		
2					0.02	0.077591	0.388739	🟨	
8				0.982913			0.347988		🟨

Figure 4.7: Entire DSMR(0) For Case Study. (Made by Obeng [27])

Figure 4.8 by Obeng [27] shows the Redo matrix for this case study. The elements were generated at random. The feedforward interactions to compare are highlighted in blue and the feedback interactions in orange.

4.3.2 Calculation of 0th Order Rework For Case Study

Equations (4.3), (4.4), and (4.5) show the calculations to determine the duration of zeroth order rework for the case study. The index was set from one to nine because the process has nine tasks. To determine the cost of first order rework, replace the time vector with the cost vector to $410k for the cost of zeroth order rework.



	7	1	5	4	3	6	9	2	8
7		0.821654							
1	0.869522								
5									
4		0.343092							
3		0.269603						0.000099	
6			0.812123				0.957585		
9					0.575526	0.166682			
2					0.243348	0.697979	0.377068		
8				0.194132			0.483947		

Figure 4.8: Redo Matrix For Case Study. (Made by Obeng [27])

$$T_0 = \sum_{i=1}^{9} T(i) \tag{4.3}$$

$$T_0 = 0.02 + 0.077 + 0.23 + 0.03 + 0.25 + 0.11 + 0.05 + 0.115 + 0.07 \tag{4.4}$$

$$T_0 = 0.952\ years \tag{4.5}$$

4.3.3 Calculation of 1st Order Rework For Case Study

To be able to determine the time and money wasted on first order rework, the First Order Vector (FOV) has to be created. To determine which tasks are repeated due to first order rework, the lower half of DSMR and the lower half of the Redo matrix are compared and if any elements of the Redo matrix are lower than the DSMR, then the tasks associated with those elements' columns have to be reworked. Comparing the bottom rows of the two matrices in Figs. 4.7 and 4.8 respectively shows that Tasks 4 and 9 have to be reworked once. The FOV is shown for the case study in Table 4.11 made by Obeng [27]. No other tasks have to be reworked in terms of the first order.

Table 4.12 by Obeng [27] shows the calculation of time wasted due to first order rework that

Table 4.11: First Order Vector For Case Study. (Made by Obeng [27])

Task Order	1	2	3	4	5	6	7	8	9
Task Name	7	1	5	4	3	6	9	2	8
1st Order Rework Vector Component	0	0	0	1	0	0	1	0	0

uses Eq. (3.7). It is a triple sumproduct calculation that involves the FOV, RC, and time vectors. The sum is in years. Replacing the time vector with the cost vector will show that the money wasted by doing first order rework is \$142k.

Table 4.12: Table Showing Calculation Of Time Wasted By First Order Rework. (Made by Obeng [27])

Task Order	1	2	3	4	5	6	7	8	9	
Task Name	7	1	5	4	3	6	9	2	8	Sum
Vector Component Multiplication	0	0	0	1(1)(0.03)	0	0	1(0.1)(0.05)	0	0	0.035

4.3.4 Calculation of 2nd Order Rework And Total Work For Case Study

To determine which tasks are repeated due to second order rework, the upper half of DSMR and the upper half of the Redo matrix are compared and if any elements of the Redo matrix are lower than the DSMR, then the tasks associated with those elements' columns have to be reworked. It is not the same as for first order rework. The loop master is used to calculate the time and money wasted due to second order rework instead of the FOV. When an incidence of second order rework is found during the comparison, the sumproducts of the loop master row that contains the tasks are involved, the RC, and the time or cost vector. The index is from the row number the incident to the column number of the incident's element as shown in Eq.

(3.9).

An example of an incident of second order rework that can be found by seeing that $Redo_{(5,8)}$ is less than $DSMR(0)_{(5,8)}$. This means that the fifth through eighth tasks in the sequence (Tasks 3, 6, 9, and 2) have to be repeated. Table 4.13 shows the calculation of second order rework. The sum is in years. The terms related to the embedded loops is equal to zero because $Redo_{(6,7)}$ is greater than $DSMR(0)_{(6,7)}$. Replace the T with C and cost of repeating this loop will be \$69.85k.

Table 4.13: Table Showing Calculation Of Time Wasted By Second Order Rework. (Made by Obeng [27])

Task Order Index	5	6	7	8	
Task Name	3	6	9	2	Sum
Vector Component Multiplication	1(0.8)(0.25)	1(0.25)(0.11)	1(0.1)(0.05)	1(0.5)(0.115)	0.29

The total duration of the process for the data point is calculated by adding the duration of each order of rework as shown in Eqs. (4.6) and (4.7). The total cost of the process for this data point is \$621.85k.

$$T_{total} = 0.952 + 0.035 + 0.29 \tag{4.6}$$

$$T_{total} = 1.27 \; years \tag{4.7}$$

4.4 Generation of Process Joint Duration/Cost Distribution Plots For Case Study

4.4.1 Process Joint Duration/Cost Distribution Plots Including Obsolescence

For this case study, the bounds of the process's total duration and cost for this case study are assumed to be 1.7 years and $750k respectively. The process life is 20 years. 10,000 Monte Carlo simulations were conducted to generate each PDF in the case study. Figures 4.9 through 4.17 by Obeng [27] show the DSMR's and resulting PDFs for duration and cost of completing the process. The interval of interest is five years. As the process's life progressed, the duration and cost increased because the probability of rework increased. The process's design would be considered obsolete at the end of life (see Fig. 4.17) because an unwanted majority of the distribution went above the acceptable thresholds. The least likely data points are blue and the most likely data points are dark red.

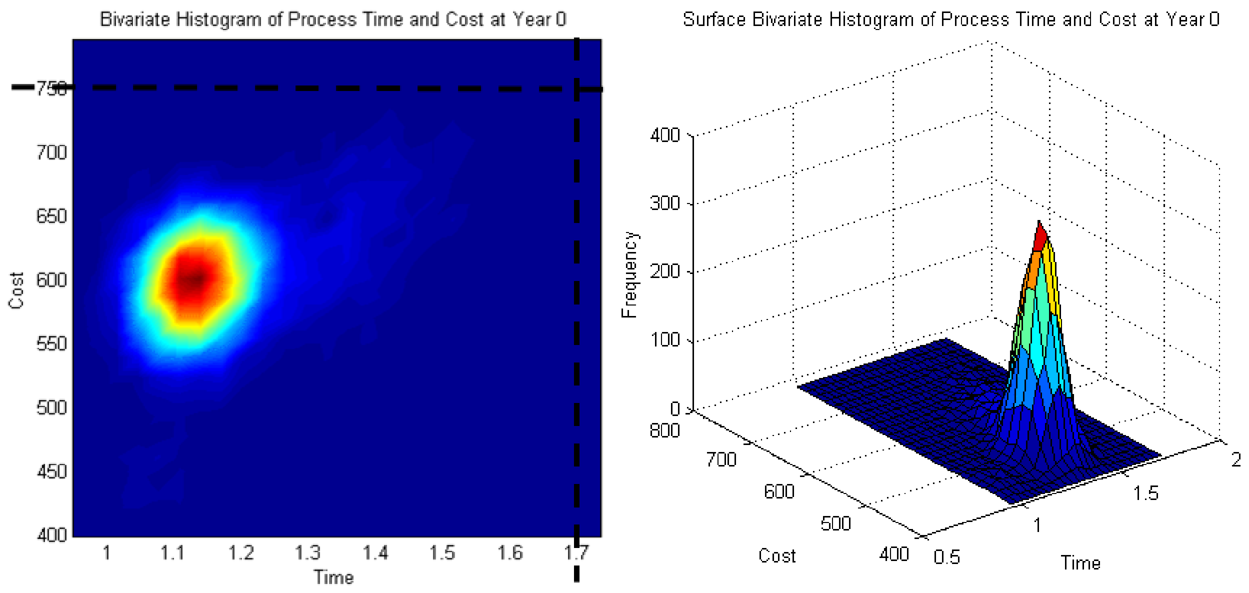
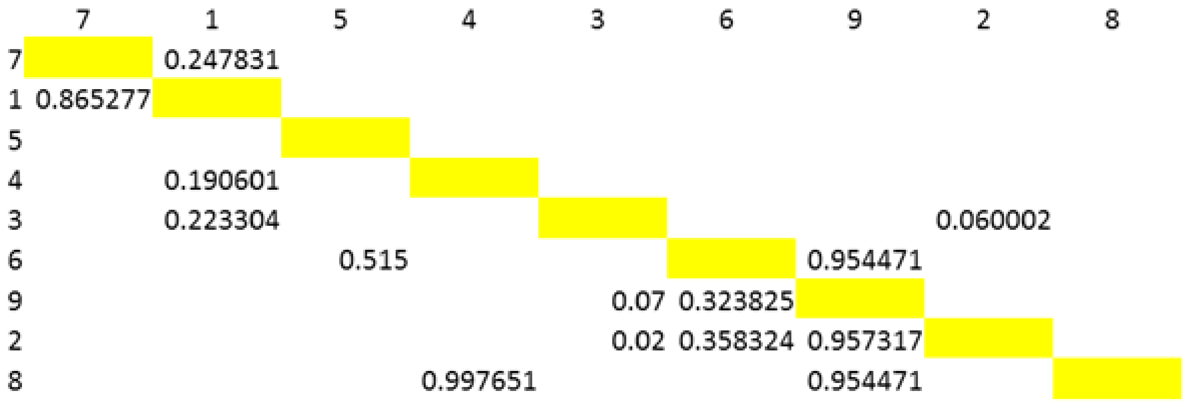


Figure 4.9: Bivariate Histogram With Constraints At Year 0 Of Case Study. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	0.247831							
1	0.865277	🟨							
5			🟨						
4		0.190601		🟨					
3		0.223304			🟨			0.060002	
6			0.515			🟨	0.954471		
9					0.07	0.323825	🟨		
2					0.02	0.358324	0.957317	🟨	
8				0.997651			0.954471		🟨

Figure 4.10: DSMR(5) Of Case Study. (Made by Obeng [27])

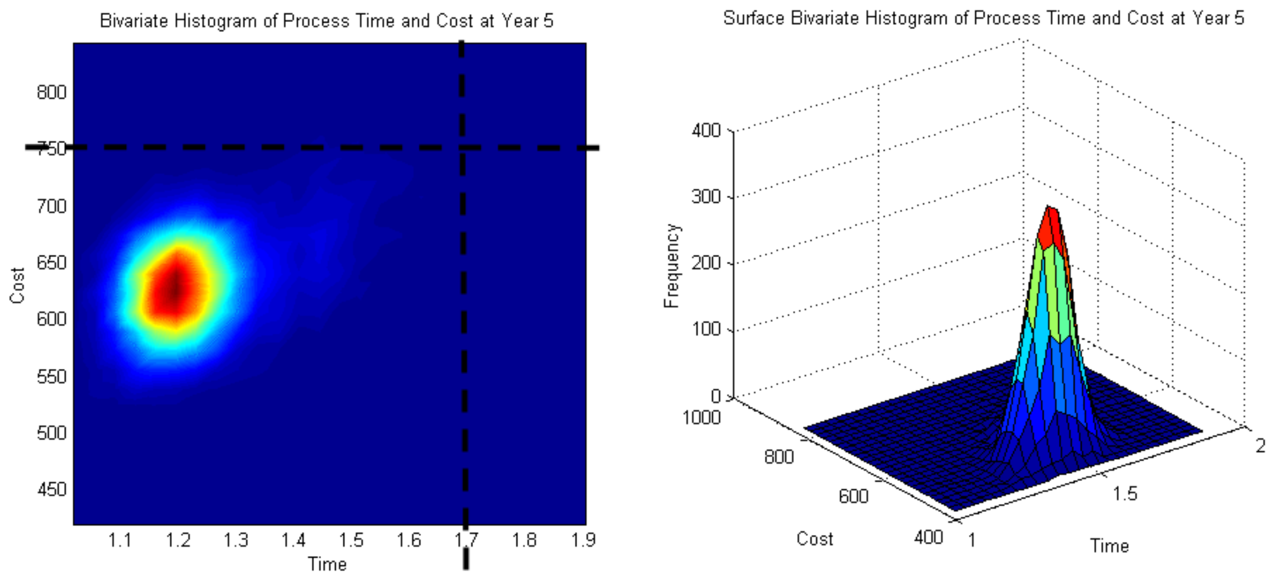
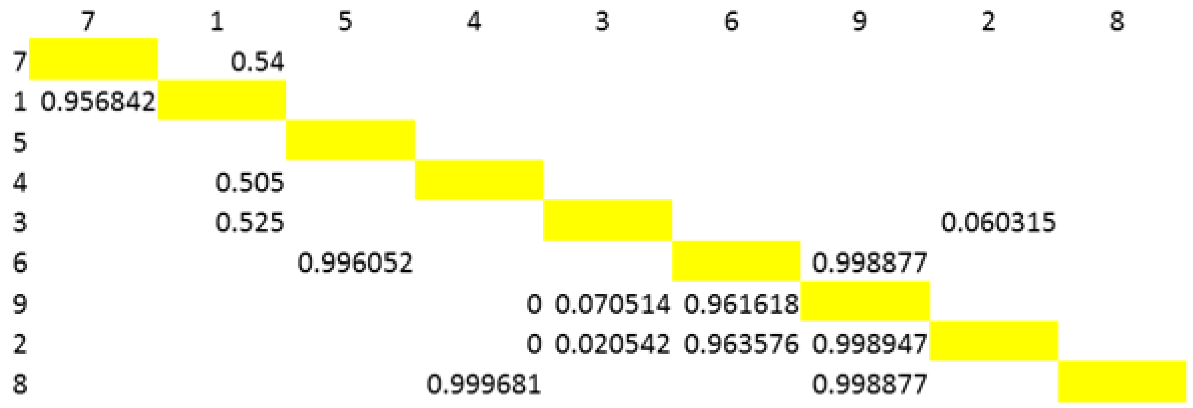


Figure 4.11: Bivariate Histogram With Constraints At Year 5 Of Case Study. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	0.54							
1	0.956842	🟨							
5			🟨						
4		0.505		🟨					
3		0.525			🟨			0.060315	
6			0.996052			🟨	0.998877		
9				0	0.070514	0.961618	🟨		
2				0	0.020542	0.963576	0.998947	🟨	
8				0.999681			0.998877		🟨

Figure 4.12: DSMR(10) Of Case Study. (Made by Obeng [27])

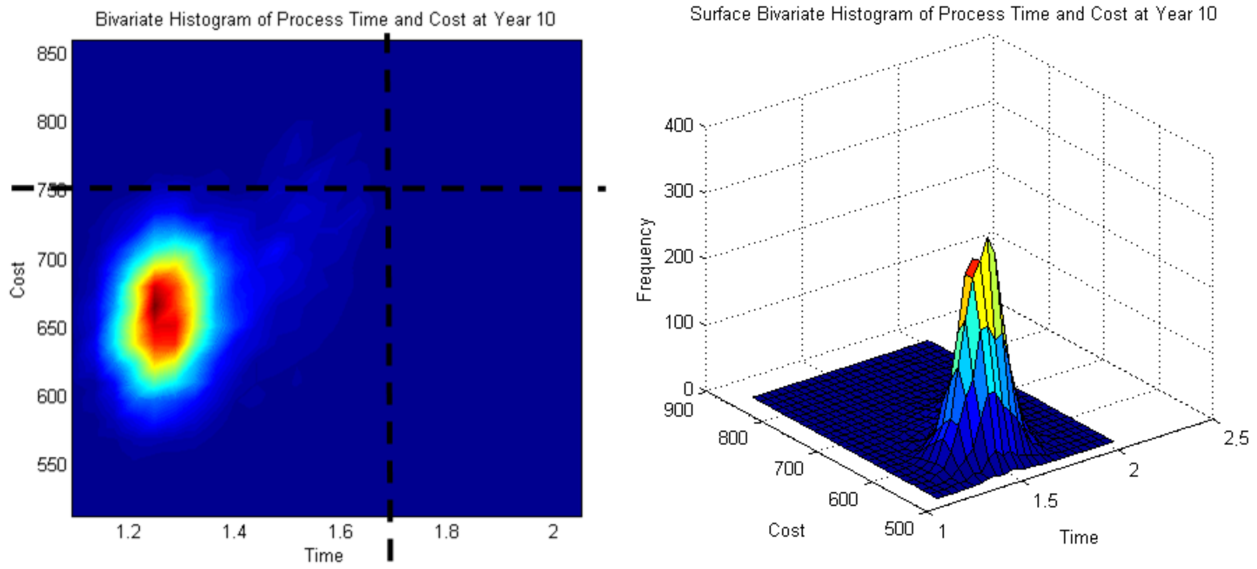
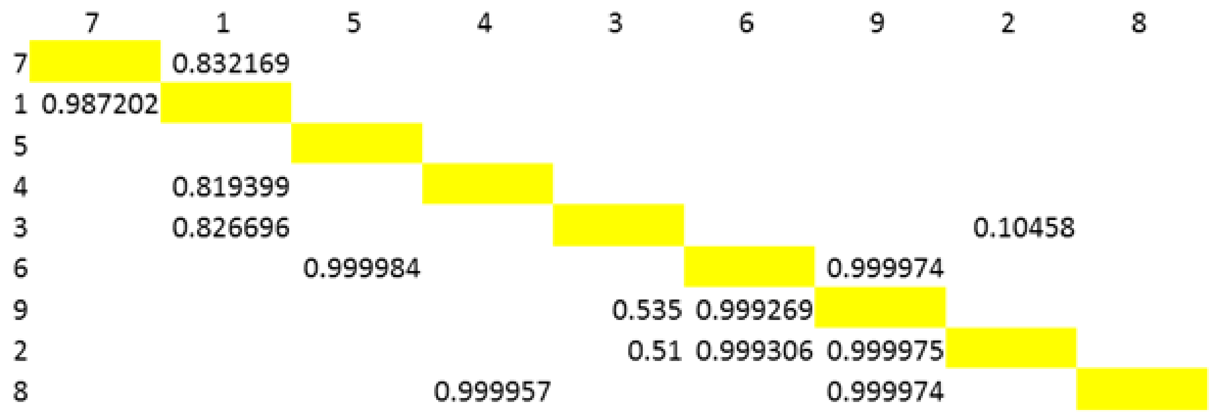


Figure 4.13: Bivariate Histogram With Constraints At Year 10 Of Case Study. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	0.832169							
1	0.987202	🟨							
5			🟨						
4		0.819399		🟨					
3		0.826696			🟨			0.10458	
6			0.999984			🟨	0.999974		
9					0.535	0.999269	🟨		
2					0.51	0.999306	0.999975	🟨	
8				0.999957			0.999974		🟨

Figure 4.14: DSMR(15) Of Case Study. (Made by Obeng [27])

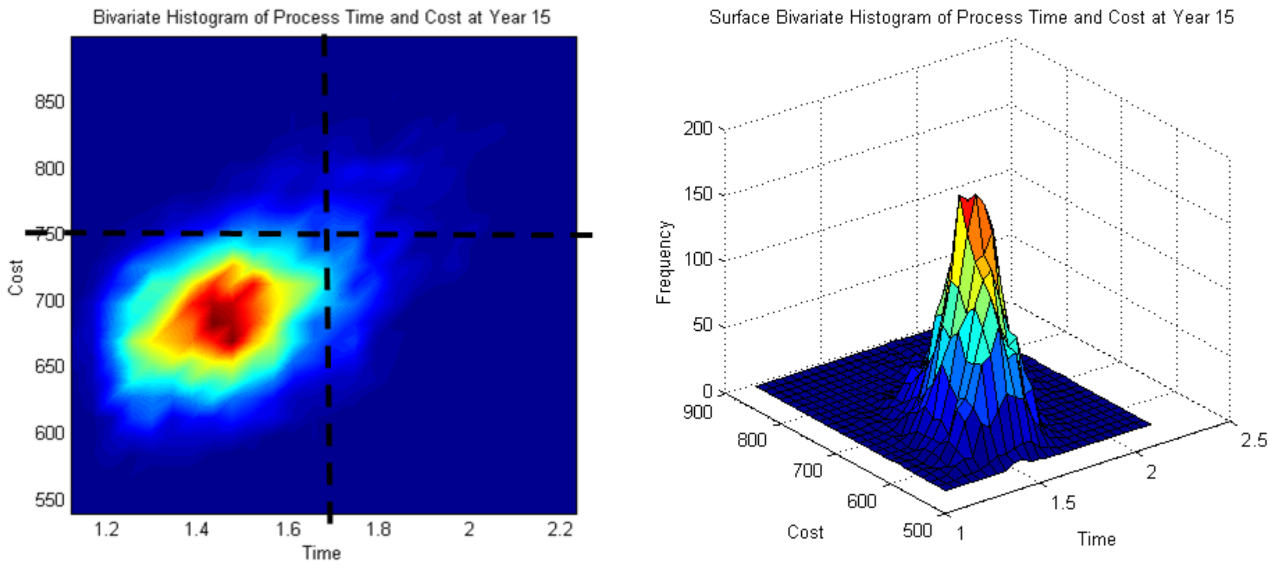
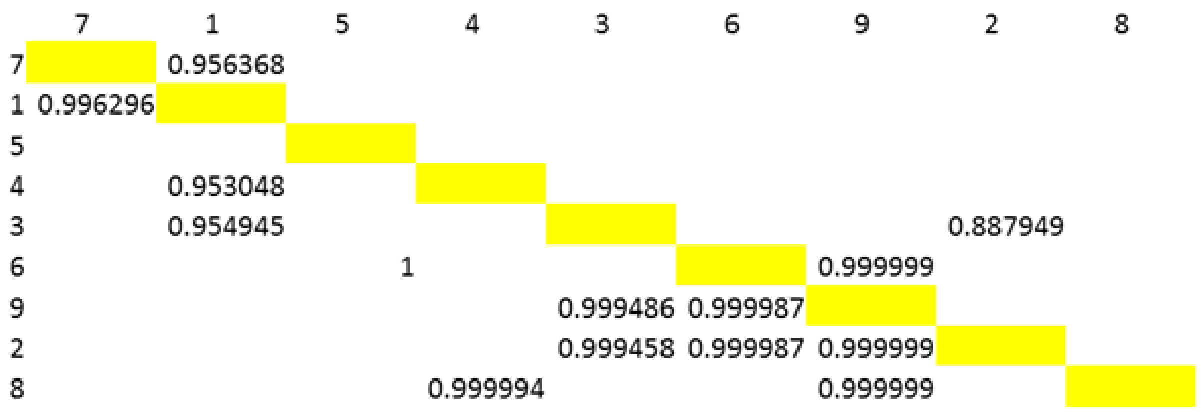


Figure 4.15: Bivariate Histogram With Constraints At Year 15 Of Case Study. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	0.956368							
1	0.996296	🟨							
5			🟨						
4		0.953048		🟨					
3		0.954945			🟨			0.887949	
6			1			🟨	0.999999		
9					0.999486	0.999987	🟨		
2					0.999458	0.999987	0.999999	🟨	
8				0.999994			0.999999		🟨

Figure 4.16: DSMR(20) Of Case Study. (Made by Obeng [27])

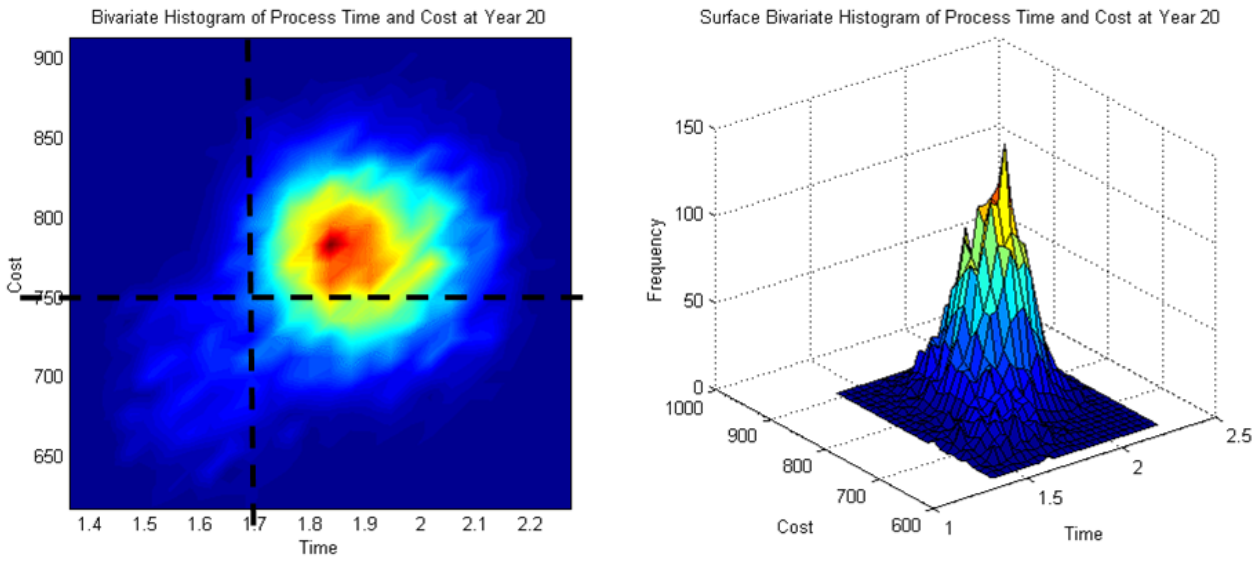
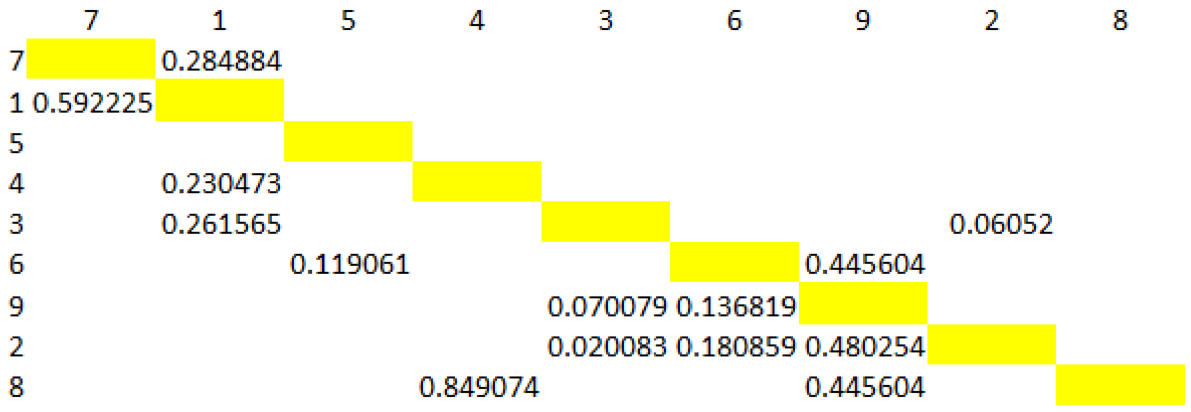


Figure 4.17: Bivariate Histogram With Constraints At Year 20 Of Case Study. (Made by Obeng [27])

4.4.2 Process Joint Duration/Cost Distribution Plots Not Considering Obsolescence

To simulate what the process's duration and cost PDFs would look life if device obsolescence was neglected and considering only device mortality, shifts in failure curves were used. To create the failure curve shifts, the technology rates from Table 4.8 were divided by three and the maturities were increased by 25%. Figures 4.18 to 4.27 by Obeng [27] show the DSMRs for each time interval and their respective duration and cost PDFs.



	7	1	5	4	3	6	9	2	8
7		0.284884							
1	0.592225								
5									
4		0.230473							
3		0.261565						0.06052	
6			0.119061				0.445604		
9					0.070079	0.136819			
2					0.020083	0.180859	0.480254		
8				0.849074			0.445604		

Figure 4.18: DSMR(0) Of Case Study Not Considering Obsolescence. (Made by Obeng [27])

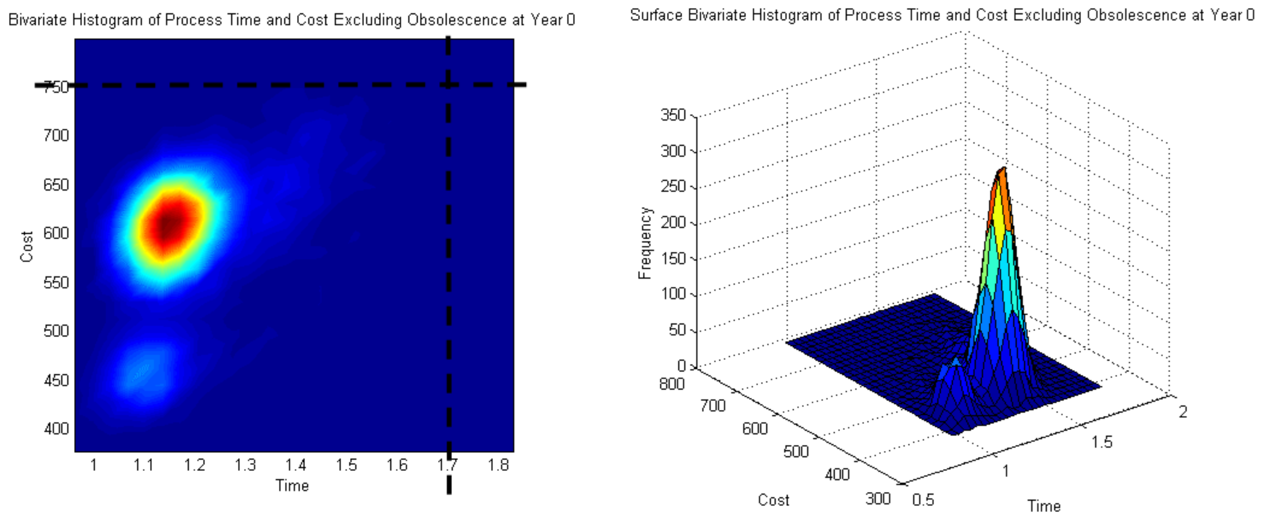
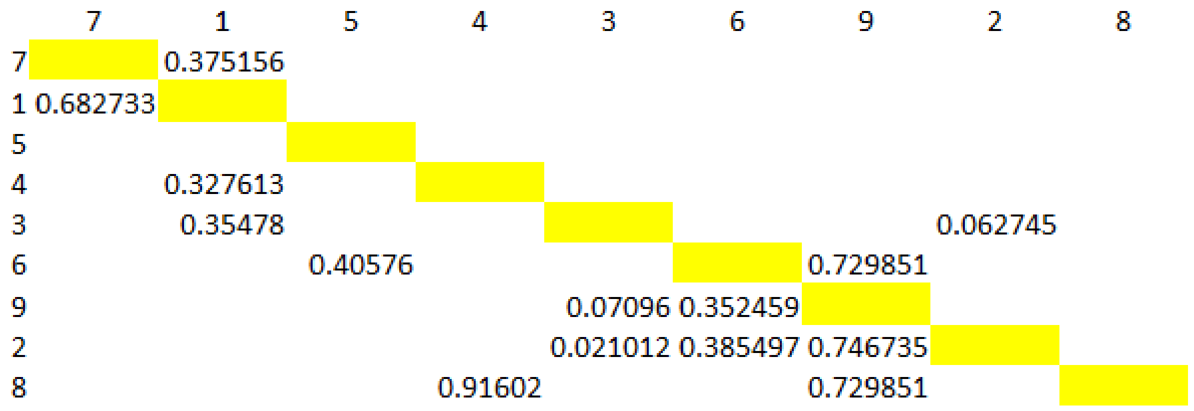


Figure 4.19: Bivariate Histogram With Constraints At Year 0 Of Case Study Not Considering Obsolescence. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7		0.375156							
1	0.682733								
5									
4		0.327613							
3		0.35478						0.062745	
6			0.40576				0.729851		
9					0.07096	0.352459			
2					0.021012	0.385497	0.746735		
8				0.91602			0.729851		

Figure 4.20: DSMR(5) Of Case Study Not Considering Obsolescence. (Made by Obeng [27])

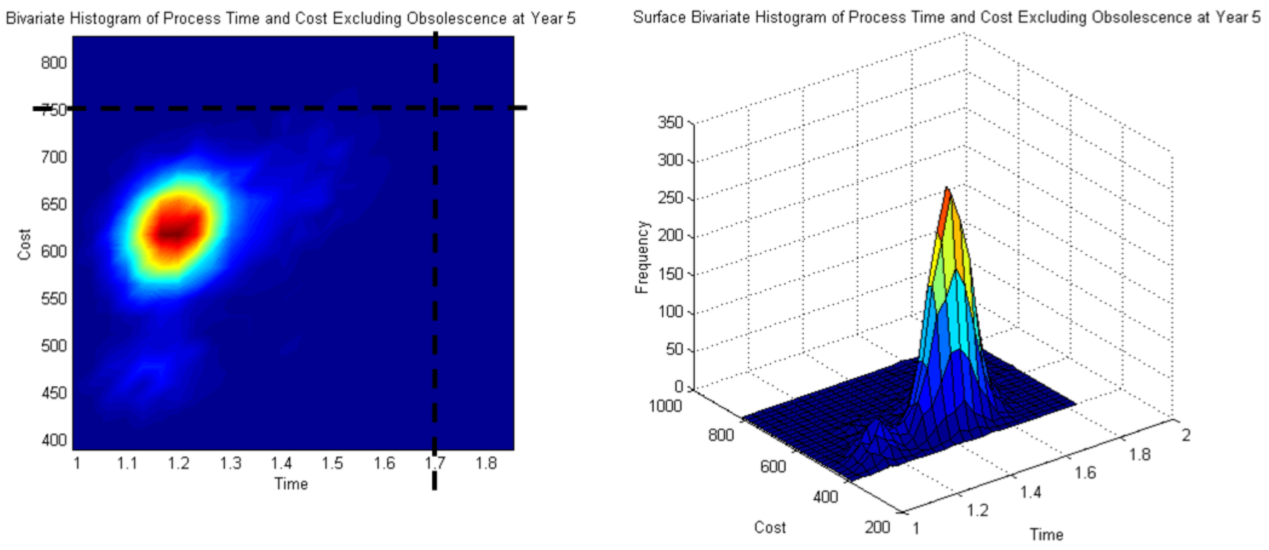
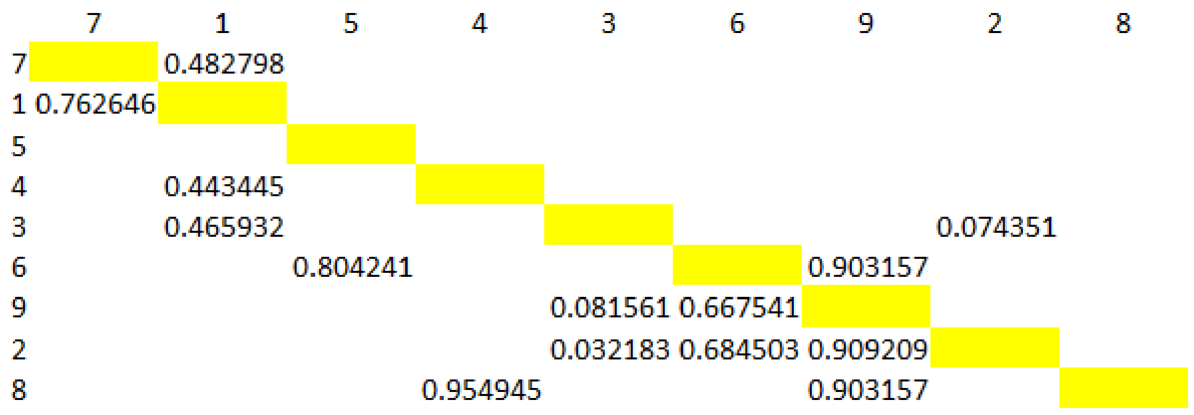


Figure 4.21: Bivariate Histogram With Constraints At Year 5 Of Case Study Not Considering Obsolescence. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7		0.482798							
1	0.762646								
5									
4		0.443445							
3		0.465932						0.074351	
6			0.804241				0.903157		
9					0.081561	0.667541			
2					0.032183	0.684503	0.909209		
8				0.954945			0.903157		

Figure 4.22: DSMR(10) Of Case Study Not Considering Obsolescence. (Made by Obeng [27])

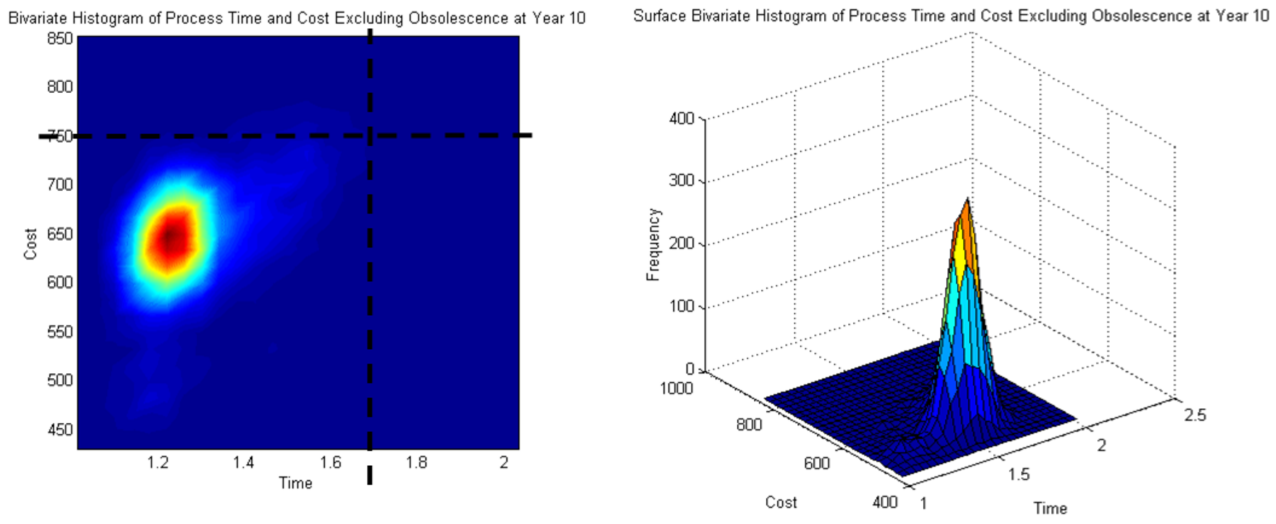
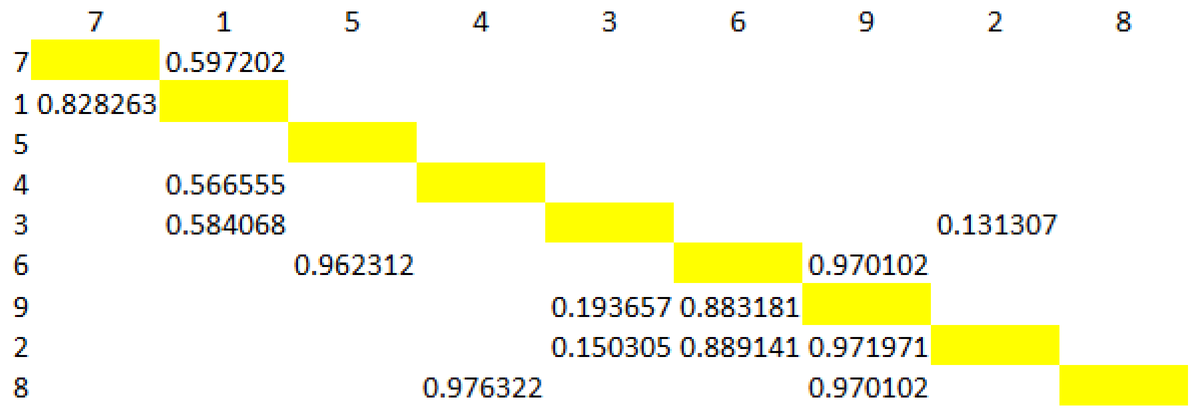


Figure 4.23: Bivariate Histogram With Constraints At Year 10 Of Case Study Not Considering Obsolescence. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	0.597202							
1	0.828263	🟨							
5			🟨						
4		0.566555		🟨					
3		0.584068			🟨			0.131307	
6			0.962312			🟨	0.970102		
9					0.193657	0.883181	🟨		
2					0.150305	0.889141	0.971971	🟨	
8				0.976322			0.970102		🟨

Figure 4.24: DSMR(15) Of Case Study Not Considering Obsolescence. (Made by Obeng [27])

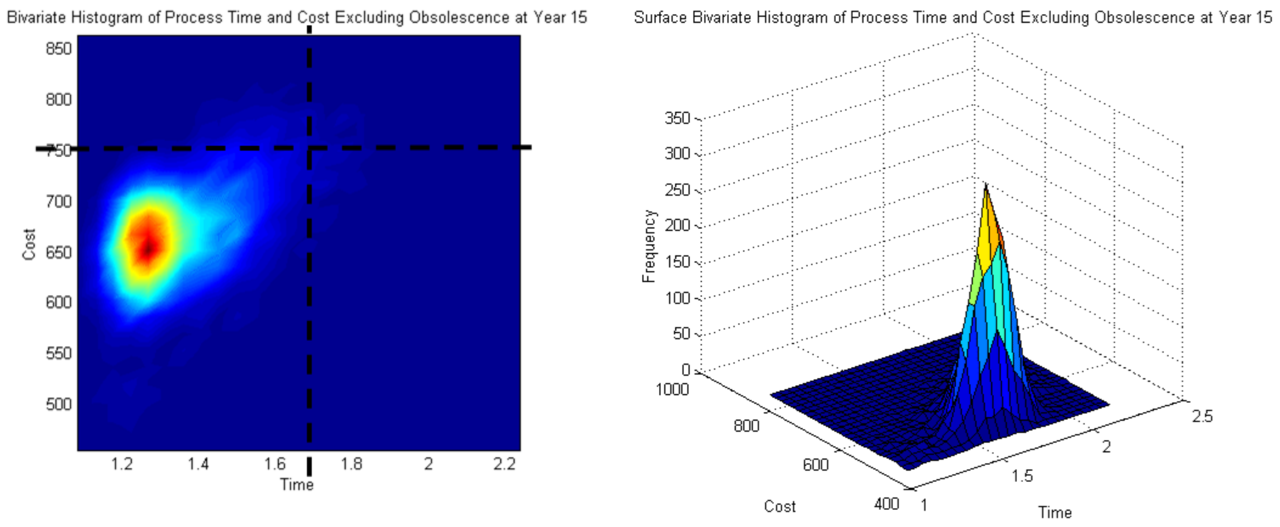
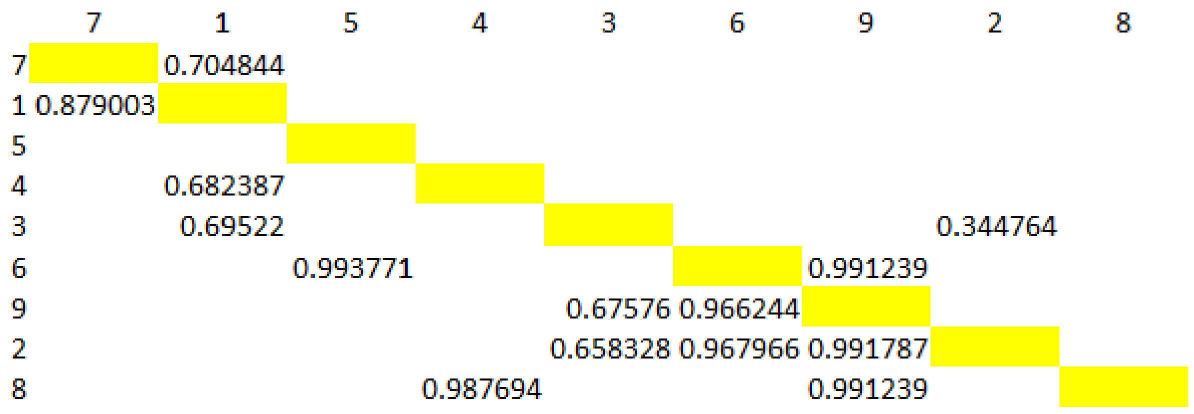


Figure 4.25: Bivariate Histogram With Constraints At Year 15 Of Case Study Not Considering Obsolescence. (Made by Obeng [27])



	7	1	5	4	3	6	9	2	8
7	🟨	0.704844							
1	0.879003	🟨							
5			🟨						
4		0.682387		🟨					
3		0.69522			🟨			0.344764	
6			0.993771			🟨	0.991239		
9					0.67576	0.966244	🟨		
2					0.658328	0.967966	0.991787	🟨	
8				0.987694			0.991239		🟨

Figure 4.26: DSMR(20) Of Case Study Not Considering Obsolescence. (Made by Obeng [27])

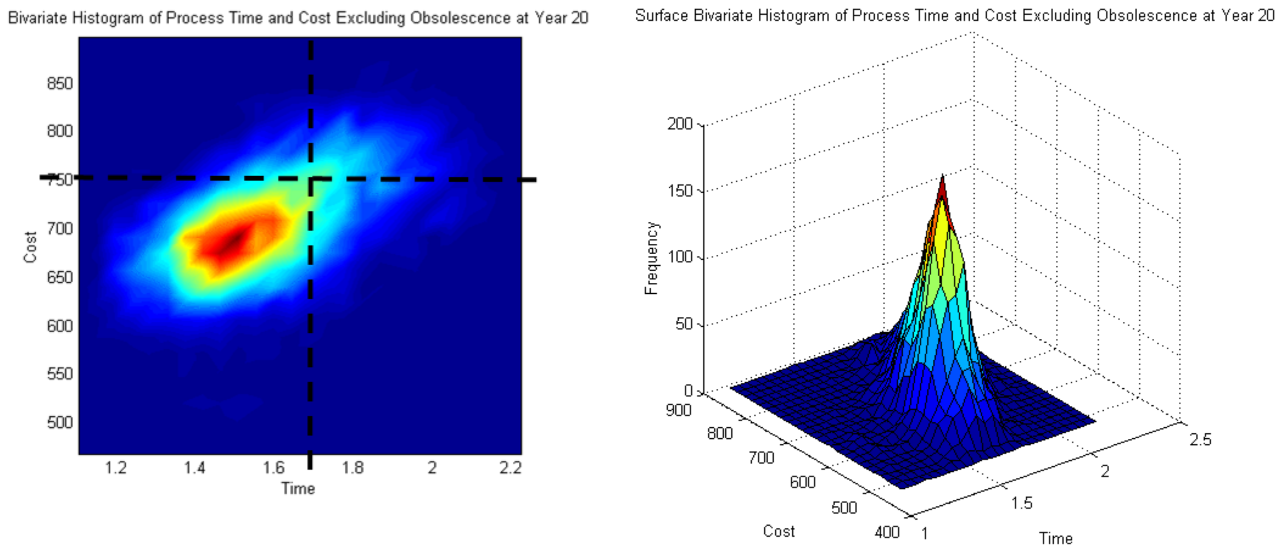


Figure 4.27: Bivariate Histogram With Constraints At Year 20 Of Case Study Not Considering Obsolescence. (Made by Obeng [27])

4.5 Sensitivity Analysis For Case Study

4.5.1 Manipulating Rework Coefficients

Figures 4.28 through 4.32 were created by making intervals from the area of most concern (red and dark red areas) after PDFs made from changing the Rework Coefficients by ± 5 and 10 percent were examined. This was done to prevent clutter from placing multiple plots in a single figure. In these figures, the expected values are coincident with each other.

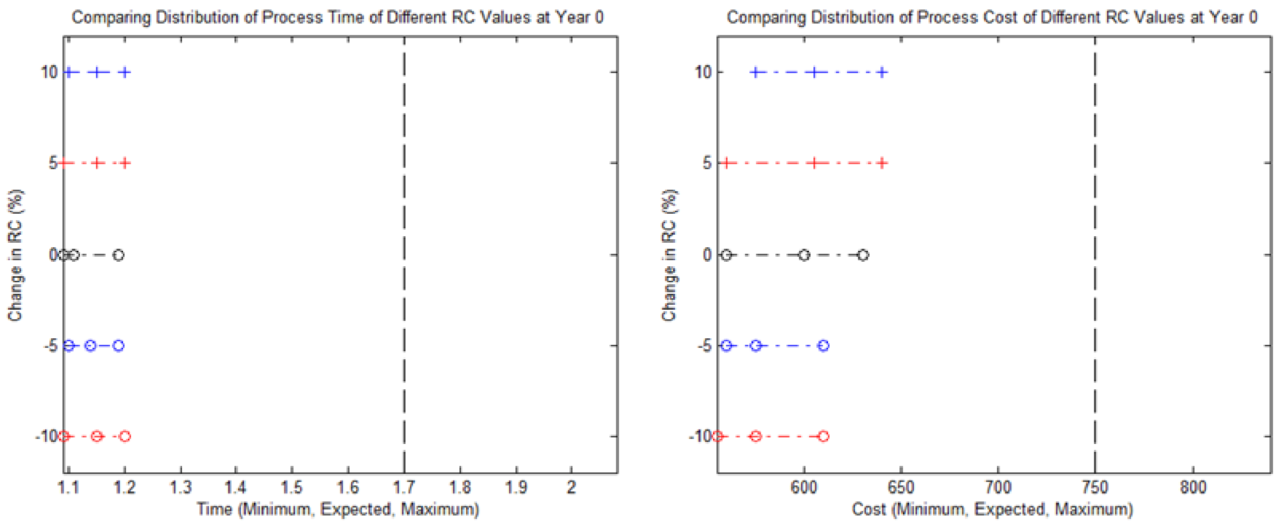


Figure 4.28: A Comparison Of Duration/Cost Distributions Of Different Rework Coefficients At Year 0.

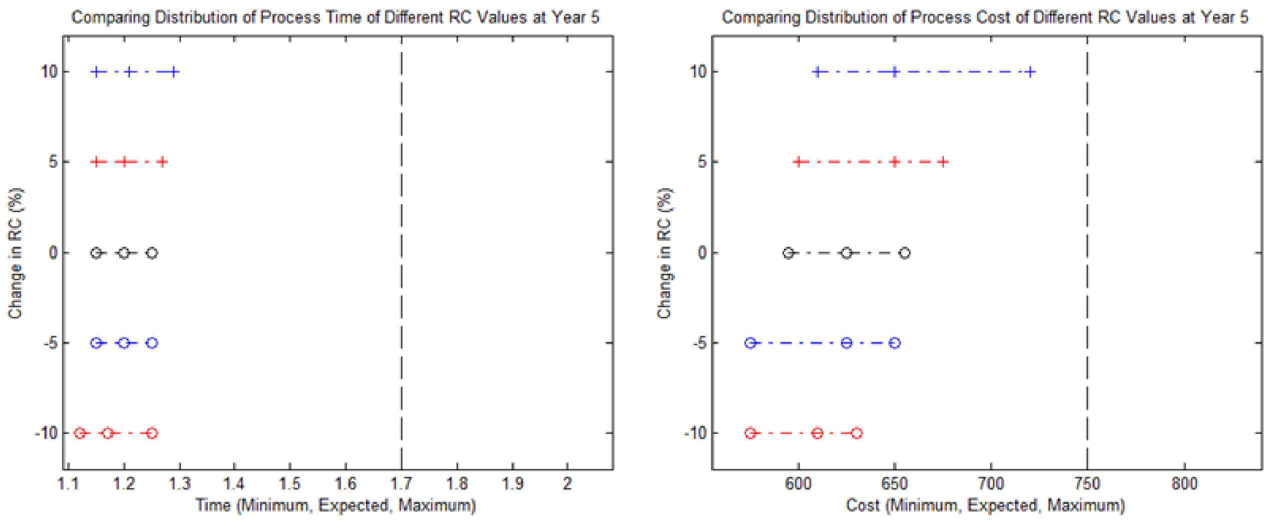


Figure 4.29: A Comparison Of Duration/Cost Distributions Of Different Rework Coefficients At Year 5.

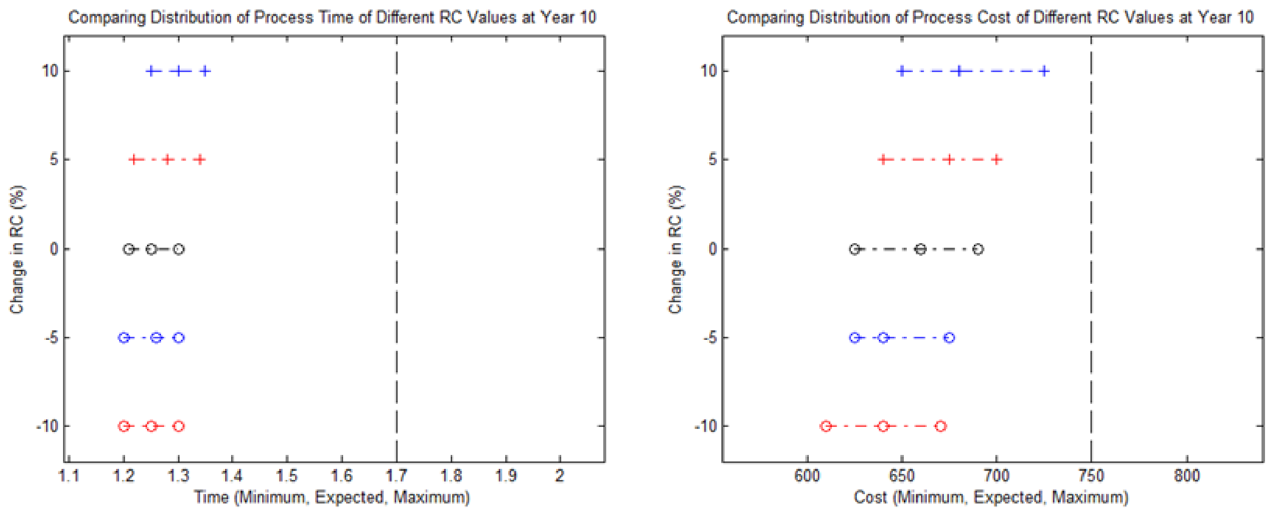


Figure 4.30: A Comparison Of Duration/Cost Distributions Of Different Rework Coefficients At Year 10.

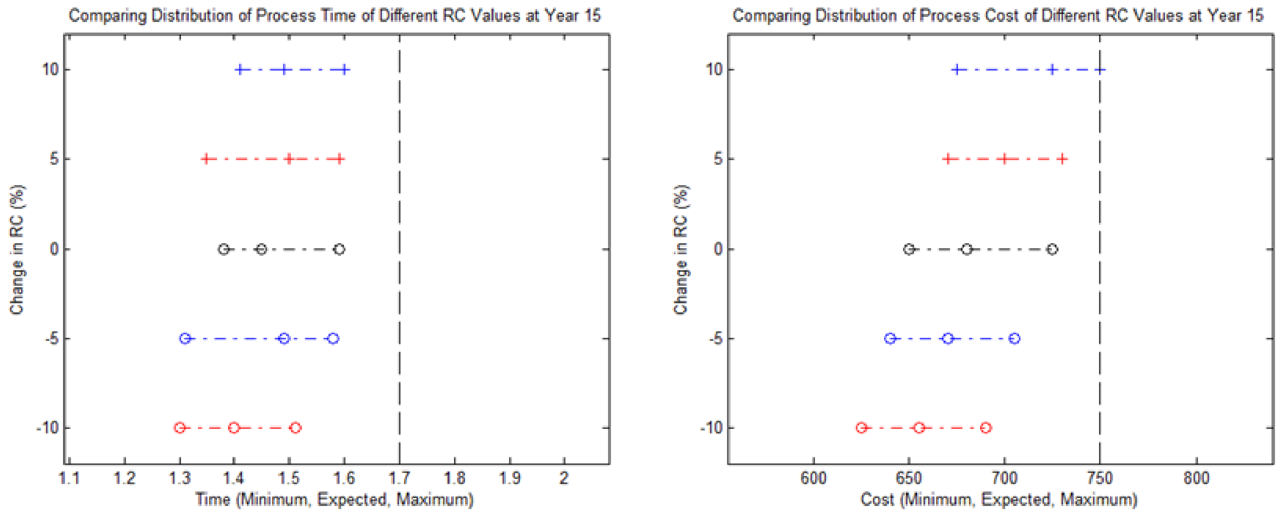


Figure 4.31: A Comparison Of Duration/Cost Distributions Of Different Rework Coefficients At Year 15.

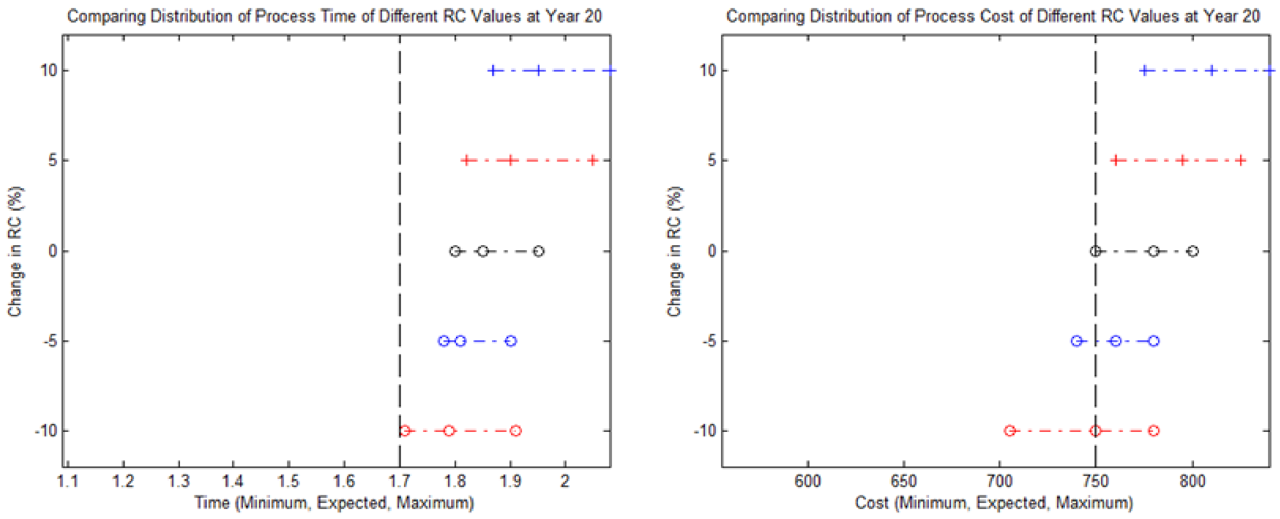


Figure 4.32: A Comparison Of Duration/Cost Distributions Of Different Rework Coefficients At Year 20.

4.5.2 Manipulating Failure Curve Rates

Figures 4.33 through 4.37 were created by making intervals from the area of most concern (red and dark red areas) after PDFs made from changing the rates of the failure curves by ± 5 and 10 percent were examined.

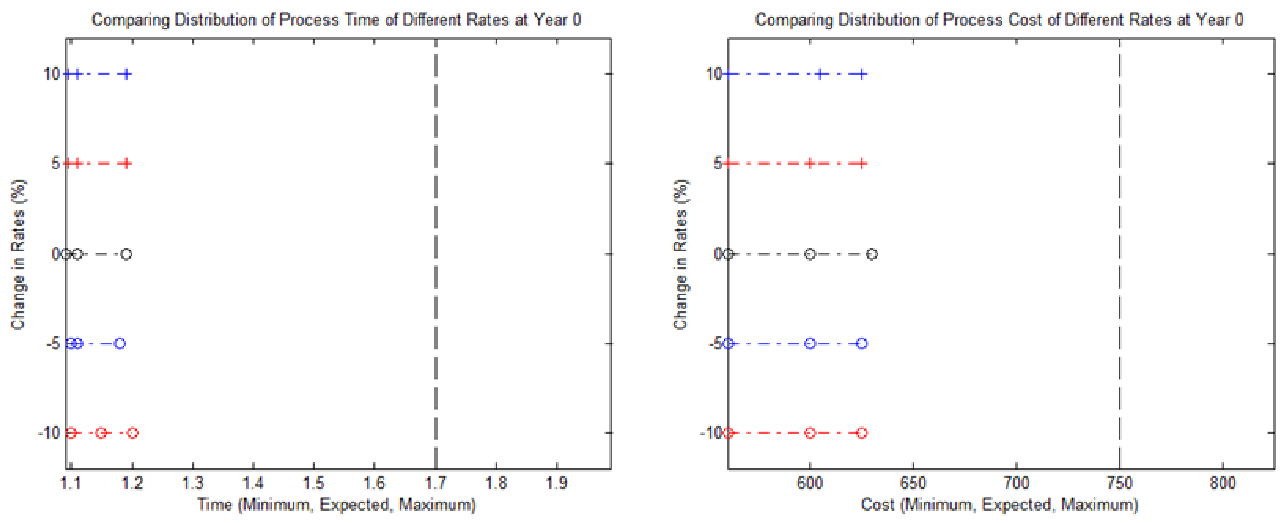


Figure 4.33: A Comparison Of Duration/Cost Distributions Of Different Failure Rates At Year 0.

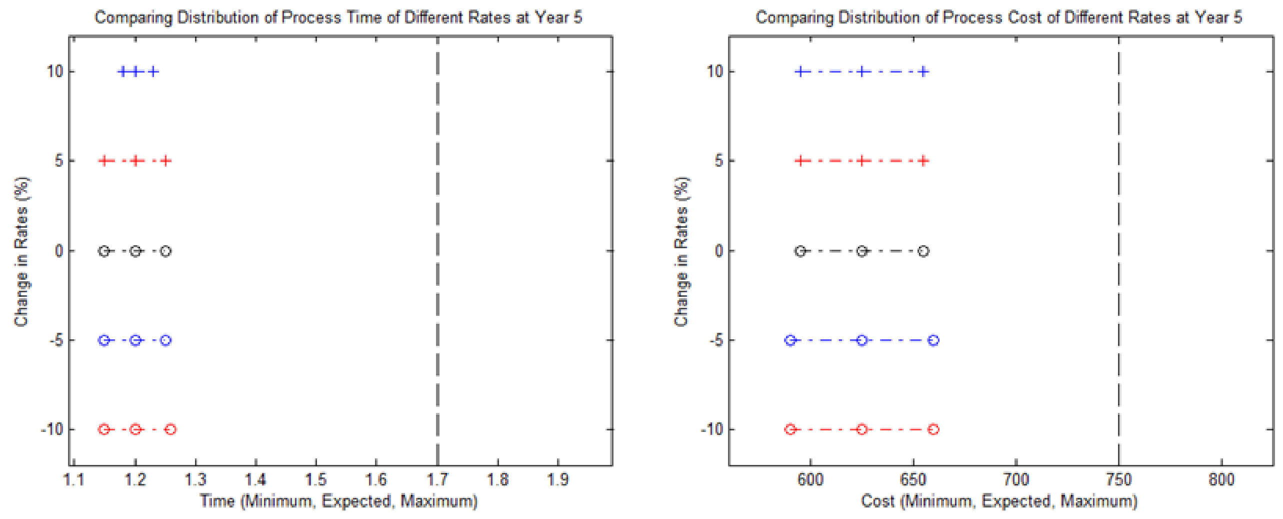


Figure 4.34: A Comparison Of Duration/Cost Distributions Of Different Failure Rates At Year 5.

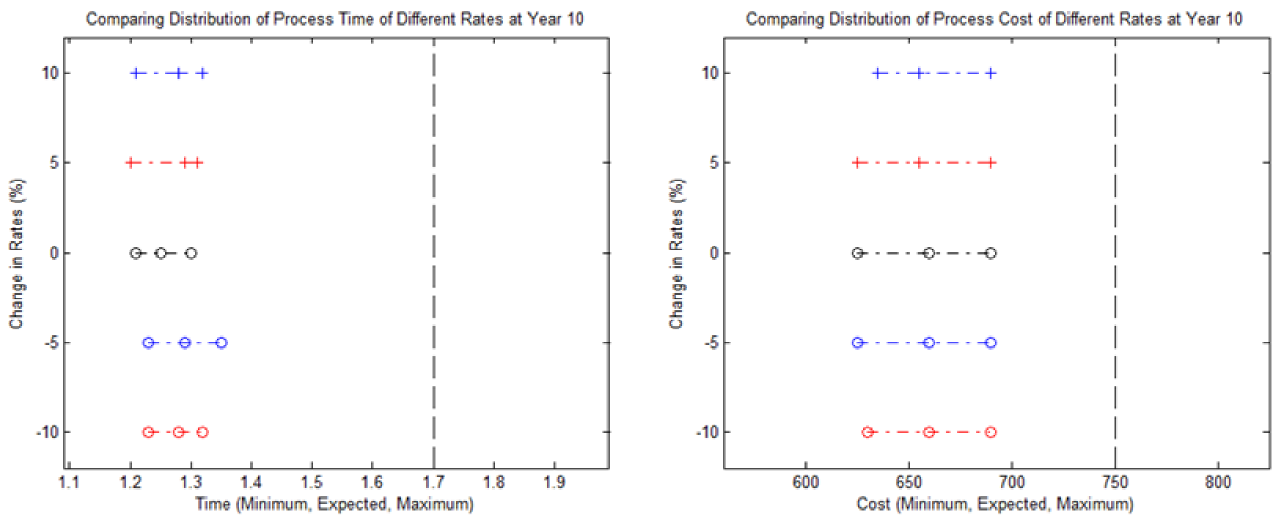


Figure 4.35: A Comparison Of Duration/Cost Distributions Of Different Failure Rates At Year 10.

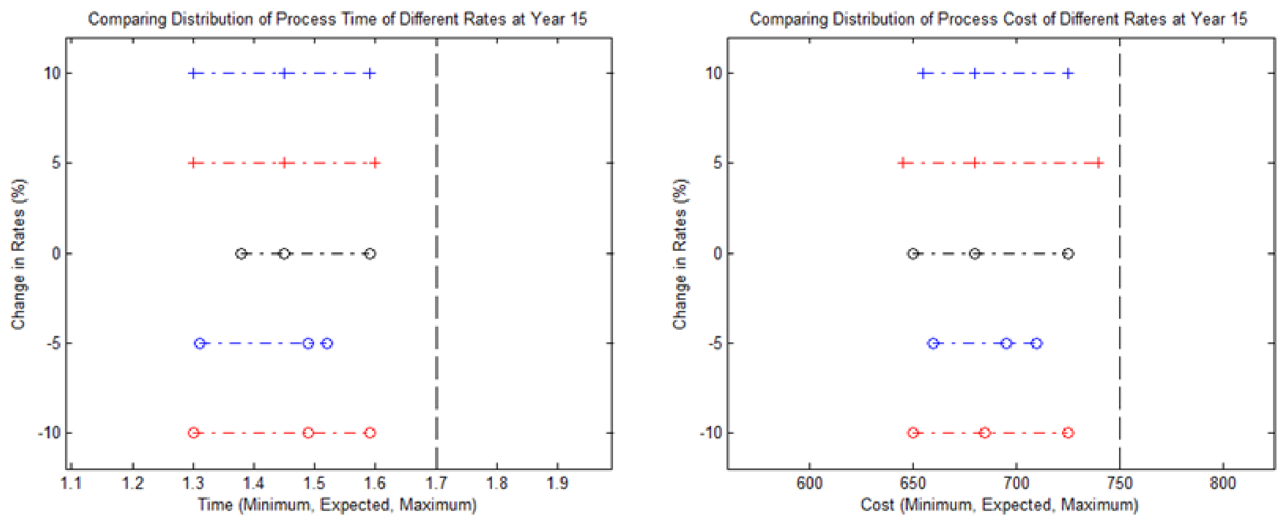


Figure 4.36: A Comparison Of Duration/Cost Distributions Of Different Failure Rates At Year 15.

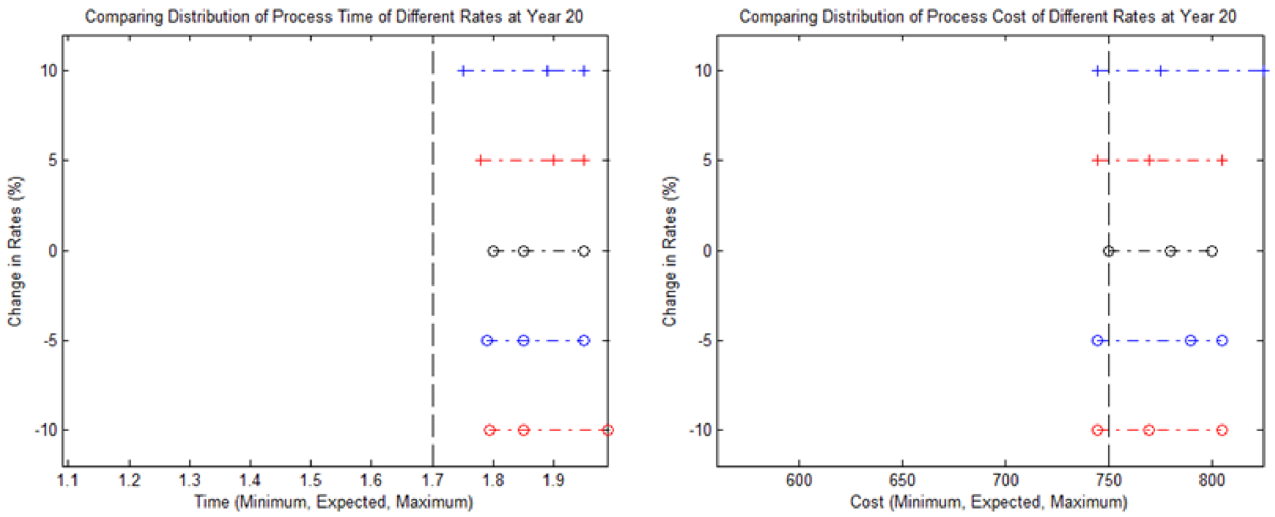


Figure 4.37: A Comparison Of Duration/Cost Distributions Of Different Failure Rates At Year 20.

4.6 Discussion of Results

4.6.1 Comparison of Joint Duration/Cost Distribution

This case study has confirmed the research hypothesis specified in Chapter 1. As time passed, the quality of the interfaces between the tasks in a process degraded as obsolescence affected them. Obsolescence is a kind of failure; it can make the probability of rework higher and, in turn, cause the time and money needed to complete a process to increase. To make comparison of PDFs easier, only the areas of highest frequency were compared. For each scenario, the areas of most concern (shown in Fig. 4.38) were plotted on the same figure to examine how the duration/cost distributions moved over time and changed their shapes.

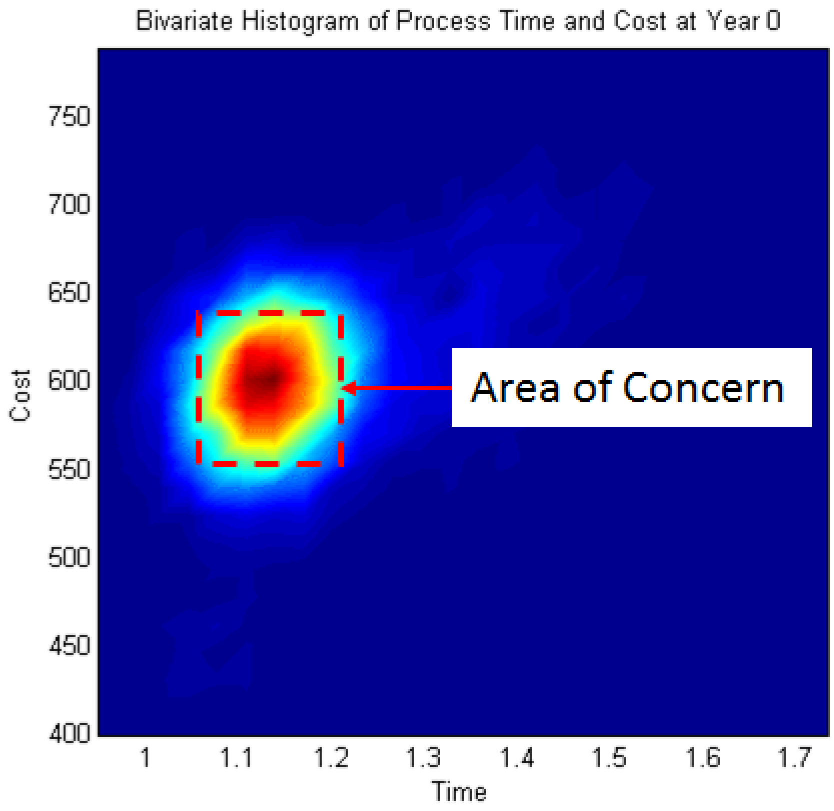


Figure 4.38: Location Of Area Of Concern On Duration/Cost PDF.

Figure 4.39 shows how the joint duration/cost distribution of the process changed as the design ages while considering obsolescence. The circles in the center of each box represented the most likely duration and cost of each distribution. The numbers to the centers are the times (in years) to which the distribution applies. As time passed the distribution showed small cycles of growth and contraction. This was caused by the likelihood of rework of some tasks to be 50%. This caused half of the simulations to include this rework and half to not consider the rework. As time passed, the likelihood of rework of some tasks was more than 50%. This caused a majority of the simulations to include the rework. Figure 4.40 shows how the joint duration/cost distribution of the process changed as the design aged without considering obsolescence. Without considering obsolescence, the expected duration and cost were substantially lower.

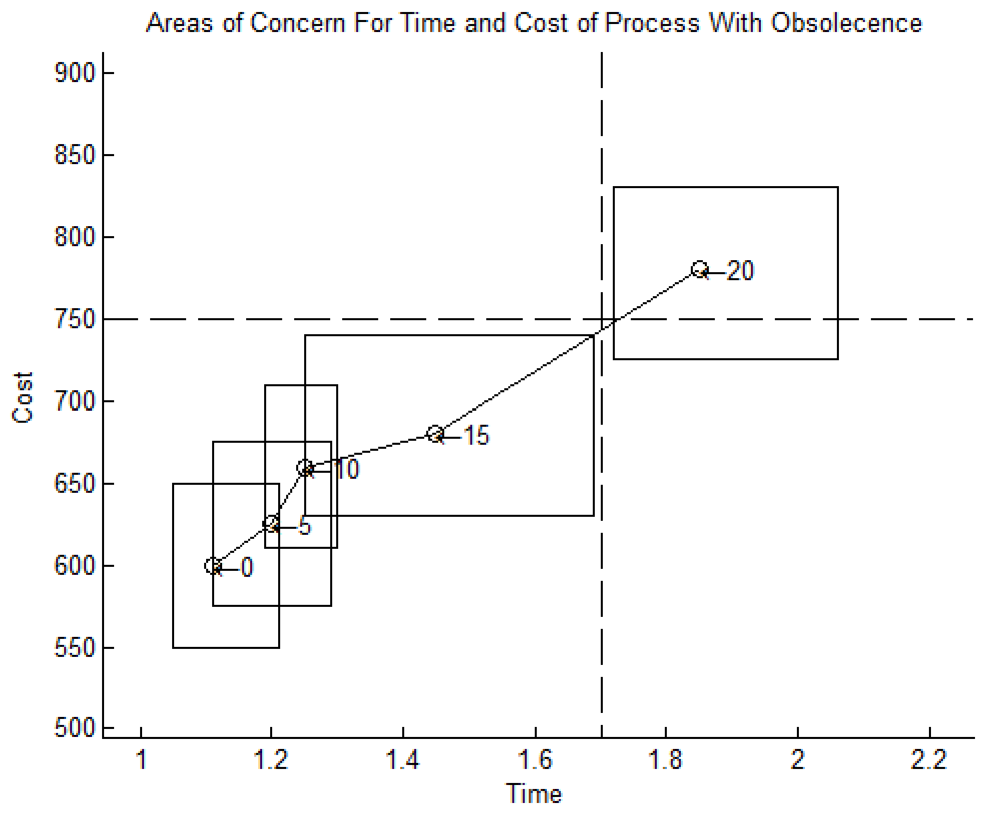


Figure 4.39: Plot Of Areas Of Concern On Duration/Cost Considering Obsolescence.

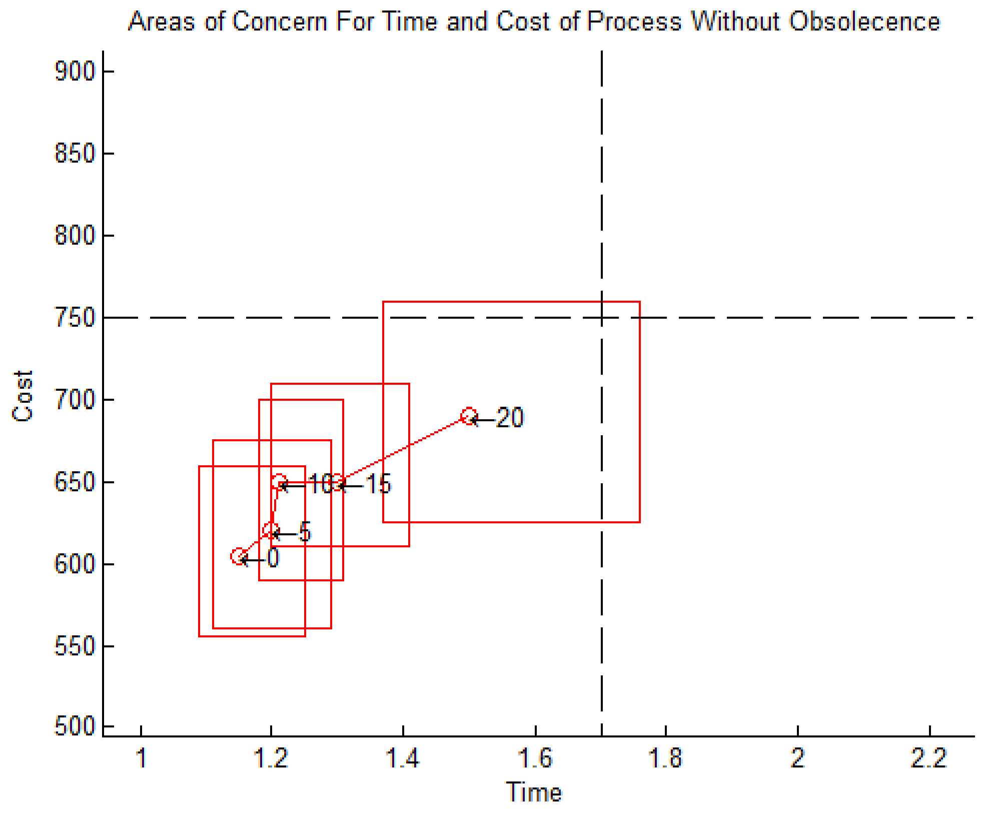


Figure 4.40: Plot Of Areas Of Concern On Duration/Cost Not Considering Obsolescence.

4.6.2 Discussion of Sensitivity Analysis

Increasing the rework coefficients had the greatest effect on the shapes of the distribution. The larger the rework coefficients, the faster the distributions advanced towards the project boundaries in time and cost. Also, the distributions had more volatility. Changing the failure rates had little effect. Just like the rework coefficients, increasing the rates made the distribution advance towards the project boundaries but not as fast as changing rework coefficients. The distributions had little to no volatility in size especially at the beginning of the process's life. In both cases, the distributions looked almost the same as at the start of life with the exception of the distributions involving the 10% increase in RC. Also in all cases, the distributions advanced slowly in the beginning and advanced faster as time passed.

CHAPTER 5

CONCLUSION

The recognition of the influence of obsolescence has caused process planners to consider its effects if they want to monitor the design of a process during its life span. This chapter will present an overview of how to use a common forecasting tool called the growth curve and a powerful systems engineering tool known as the DSM to model the obsolescence of a complex system in the first section, Section 5.1. The second section, Section 5.2 will provide suggestions for future work.

5.1 Contributions of This Dissertation

5.1.1 Outline of Tools Used to Develop Methodology

Growth curves are a common type of curve used by technological forecasters to model obsolescence after noticing the similarities between the physical failures of different types of devices. There has been suggestions for using a growth curve to represent the combined probability of failure by obsolescence (a loss in utility) and failure by getting wear and tear. The reason why the s - shaped curve was used to model the failure of interfaces is because this curve is a continuous function and few parameters are needed to define it.

The DSM is a square matrix that is used in process design. The reason why DSM is selected to represent the interfaces between any pairs of tasks in a process is because it is better at representing the feedforward (below the diagonal in this dissertation)and feedback interactions (above the diagonal) present in complex processes compared to Gantt charts or critical path networks. The DSM can also model task interfaces that are not binary to model interfaces with different weights.

5.1.2 Overview of Methodology

Figure 5.1 contains a shortened version of the flowchart in Fig. 3.1. The section in red represents the preparation needed to model the duration and cost of a process. The section in blue represents the creation of the PDFs themselves.

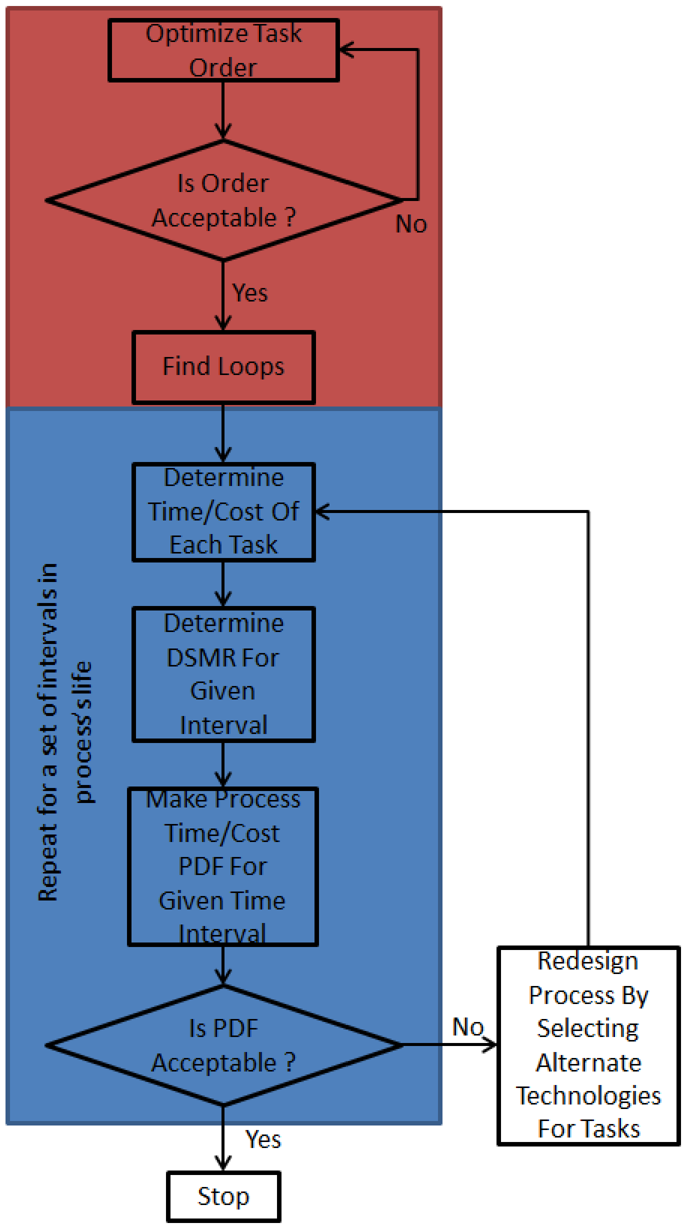

Optimize Task Order
Is Order Acceptable ?
No
Yes
Find Loops
Determine Time/Cost Of Each Task
Determine DSMR For Given Interval
Make Process Time/Cost PDF For Given Time Interval
Repeat for a set of intervals in process's life
Is PDF Acceptable ?
No
Redesign Process By Selecting Alternate Technologies For Tasks
Yes
Stop

Figure 5.1: A Compact Flowchart Of Methodology.

One of the contributions of this dissertation as shown in the red section is the creation of a loop master which is a tool to keep track of which tasks in a process form loops. Another contribution is the steps involved in making corrections in finding loops to make the powers of adjacency matrix method more effective.

The largest contribution of this dissertation is involved in the blue section which is a DSM - based model (DSMR) in which the state of interactions can be determined at any time in a process's life. To be able to do that, three DSMs (DSMRStart, DSMRate, and DSMMAT) were made from the failure curves of the technologies used to complete the tasks. They are the key inputs for the DSMR. This can be used to determine how obsolescent a process's design is based on how the various technologies used to complete the process becomes obsolescent. The technology used to complete each task used a growth curve made of two failure curves (failure by physical means and by obsolescence). A process's resistance to obsolescence can be determined by examining how joint duration/cost distributions move and change shape as the process's design ages. The methodology discussed in this dissertation is intended for use during the design phase of a process.

5.2 Suggestions for Future Work

5.2.1 Simulation Preparation Improvement

Even with the automation of the methodology through computer programs, a great deal of preparation was needed to create the inputs for the simulations. This can be subject to errors. The biggest concern is in the DSM optimization. The software requires a human to check if the DSMTS is acceptable before anything else can be done because the optimizer might not have been given enough generations and/or population size to converge to an acceptable solution. To help with this, extra code should be added to the optimizer that will determine an appropriate amount of generations and population size before starting the optimization.

5.2.2 Simulation Modifications

Another issue that process designers, planners, and operators have to deal with is the increasing complexity of the process. As an organization uses a process, upgrades are sometimes made to the process's design so that it can keep up its utility. Usually, the upgrades come in the form of add-ons to the number of tasks and/or the equipment used to complete the tasks. Figure 5.2 shows an example of how the tasks are assumed to interface with each other in this dissertation. The task in question (shown in blue in the center) is dependent on a dominant technology and so are its predecessor(s) (shown in red on the left) and its successor(s) (shown in orange on the right).

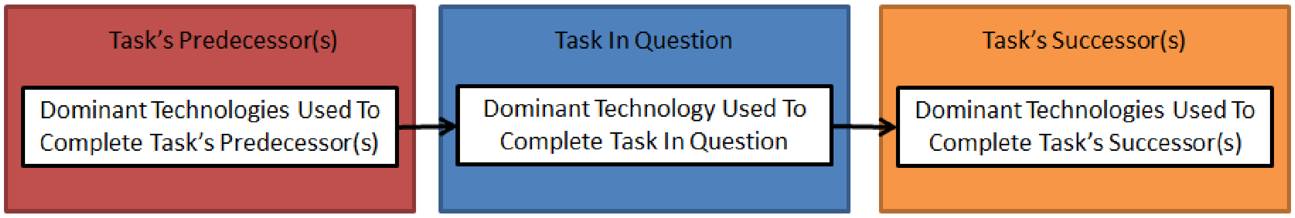

Task's Predecessor(s)
Dominant Technologies Used To Complete Task's Predecessor(s)
Task In Question
Dominant Technology Used To Complete Task In Question
Task's Successor(s)
Dominant Technologies Used To Complete Task's Successor(s)

Figure 5.2: This Dissertation's Illustration Of How A Task Interfaces With Its Predecessor(s) And Successor(s).

If a process's design is upgraded with add-ons to the equipment used to complete a task in question, more than one technology might dominate in its completion. Such a situation is illustrated in Fig. 5.3. What happens is that one or more technologies are used to handle inputs from a task's predecessors and one or more technologies are used to create and handle outputs for the task's successors.

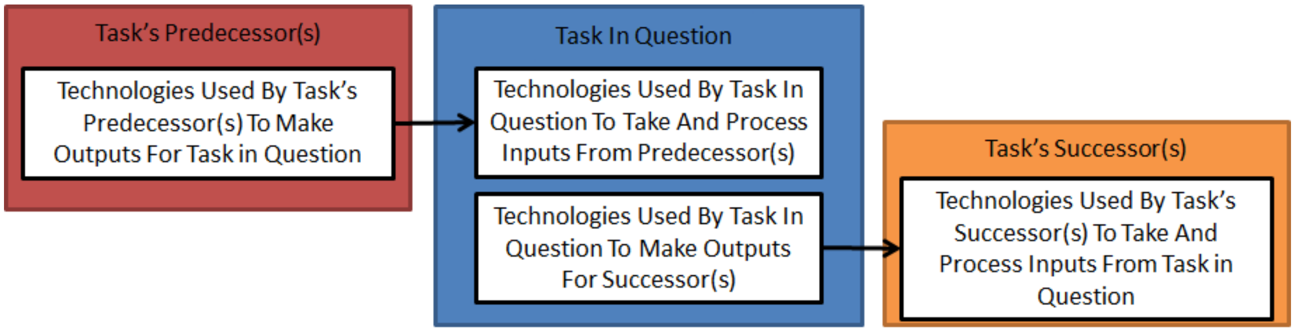

Task's Predecessor(s)
Technologies Used By Task's Predecessor(s) To Make Outputs For Task in Question
Task In Question
Technologies Used By Task In Question To Take And Process Inputs From Predecessor(s)
Technologies Used By Task In Question To Make Outputs For Successor(s)
Task's Successor(s)
Technologies Used By Task's Successor(s) To Take And Process Inputs From Task in Question

Figure 5.3: A More Complex Illustration Of How A Task Interfaces With Its Predecessor(s) And Successor(s).

If this occurs, expressing the RCs in the form of a vector will not be sufficient for calculating the time and money wasted on rework because the technologies used to complete a task in question are not the same. Here, the RCs will have to be expressed in the from of another DSM such as in a model by Browning and Eppinger [18]. The DSM that they used in calculating rework is read as "the percent of task i has to be reworked because of an error in task j is (i, j) percent."

Another suggestion for future work may be the method of changing how the rework cycles. In this dissertation, the number of times a task is reworked is based on the number of interactions it has with the other tasks and the probabilities of rework. Incidences of rework only occur once per simulation. In reality, that might not always be true. As processes become more complex, it becomes increasingly difficult to correctly complete tasks. Platanitis, Pop-Iliev, and Barari [28] and Lee, Ong, and Khoo [29] suggested using eigenvalues and eigenvectors of DSMs like the DSMRs in this dissertation to determine the number of times tasks are reworked while Smith and Eppinger [30] suggested using functions that are derived from the development times of coupled pairs in a given process. All of these suggestions may be incorporated in the method advocated in this dissertation.

REFERENCES

[1] Locatelli, G., Mancini, M., Romano, E., 2014, “Systems Engineering to Improve the Governance in Complex Project Environments.” International Journal of Project Management. **32**. pp. 1395-1410.

[2] Martino, J.P., 1993, *Technological Forecasting For Decision Making 3rd Edition*, McGraw-Hill Inc, Chapter 3.

[3] Nielsen, J. 1998, “Nielsen’s Law of Internet Bandwidth.”, `http://www.nngroup.com/articles/law-of-bandwidth/`.

[4] Kucharavy, D., De Guio, R., 2007, “Applications of S-Shaped Curves,” *“TRIZ-Future Conference 2007,”* Frankfort, Germany.

[5] Intepe, G., and Koc , T. 2012, “The Use of S Curves in Technology Forecasting and its Application On 3D TV Technology.” International Scholarly and Scientific Research & Innovation. **6**(11). Pages 1429-1433.

[6] Lam, C. W., Lim, S. R., Schoenung, J. M. 2012, “Linking Material Flow Analysis with Environmental Impact Potential: Dynamic Technology Transition Effects on Projected E-waste in the United States.” Journal of Industrial Ecology, **17**(2), pp 299-309.

[7] Ryu, J., Byeon, S. C. 2011, “Technology Level Evaluation Methodology Based on the Technology Growth Curve.” Technological Forecasting & Social Change, **78**, pp 1049-1059.

[8] Miranda, L., Lima, C. 2013, “Technology Substitution and Innovation Adoption: The Cases of Imaging and Mobile Communication Markets.” Technological Forecasting & Social Change, **80**, pp 1179-1193.

[9] Solomon, R., Sandborn, P., Pecht, M. 2000, “Electronic Part Life Cycle Concepts and Obsolescence Forecasting.” IEEE Transactions on Components and Packaging Technologies, **23**(4), pp 1179-1193.

[10] Karls, J., Dickens, H., Shon-Roy, L. 1998, “Status 1998, A Report on the Integrated Circuit Industry.” ICE.

[11] Bartels, B., Ermel, U., Sandborn, P., Pecht, M., 2012, *Strategies to the Prediction, Mitigation and Management of Product Obsolescence*, John Wiley & Sons Inc, Chapters 5 and 6.

[12] Barreca, S.L., 2000, “Technology Life Cycles and Technological Obsolescence.” Technical Report. BCRI Inc.

[13] Obeng, G. K., 2013, “Design and Implementation of Software for Technological Forecasting and Modeling Technological Obsolescence, M.S. thesis, Department of Mechanical Engineering, Old Dominion University.

[14] Browning, T., 1998, “Use of Dependency Structure Matrices For Product Development Cycle Time Reduction,” *Fifth ISPE International Conference on Concurrent Engineering*, Tokyo, Japan.

[15] Gunawan, I. (2012), “Analysis of Design Structure Matrix Methods in Design Process Improvement.” International Journal of Modeling and Simulation. **32**(2).

[16] Tripathy, A., Eppinger, S., (2011), “Organizing Global Product Development for Complex Engineered Systems.” IEEE Transactions on Engineering Management, **58**(3), pp 510-529.

[17] Abdelsalam, H. M. E., & Bao, H. P. 2007, “Re-sequencing of Design Processes With Activity Stochastic Time and Cost: An Optimization-Simulation Approach.” Journal of Mechanical Design, **129**(2).

[18] Browning, T. R., Eppinger, S. D., 2002, “Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development,” IEEE Transactions on Engineering Management, **49**(4).

[19] Khadke, K., and Gershenson J., 2007, "Technology Change Analysis For Product and Product Platform Change." International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Los Vegas, United States.

[20] Devereaux, J., 2010, "Obsolescence: A Systems Engineering Management Approach For Complex Systems," M.S. thesis, Department of Engineering Management, Massachusetts Institute of Technology.

[21] Nomaguchi, Y., Tsutsumi, D., Fujita, K., 2012, "Planning Method of Creative and Collaborative Design Process With Prediction Model of Technical Performance and Product Integrity," Concurrent Engineering: Research and Applications, **20**(4), pp 315-334.

[22] Gebala, D. A., and Eppinger S.D., 1991, "Methods for Analyzing Design Procedures." *Third International Conference on Design Theory and Methodology*, Miami, United States.

[23] Maurer, M. S., 2007, "Structural Awareness in Complex Product Design," PhD Dissertation, Department of Mechanical Engineering, Institute of Product Development.

[24] Obeng, G. K., Bao, H. P., 2016, "Integrating Technological Obsolescence in Quantitative Forecasting of Memory Device Failure Cost," International Journal of Engineering and Mathematical Modeling, **3**(1).

[25] Obeng, G. K., Bao, H. P., 2014, "Consideration of Technological Obsolescence in Quantitative Forecasting and Economic Life Analysis," *International Conference on Engineering and Applied Sciences Optimization*, Kos Island, Greece.

[26] Rogers, J. L., 2014, "Tools and Techniques for Decomposing and Managing Complex Design Projects," Journal of Aircraft, **36**(1) pp 266 - 274.

[27] Obeng, G. K., Bao, H. P., 2016, "Design Structure Matrix with Consideration of Technological Obsolescence For Complex Engineering Projects." *Advanced Technology Symposium 2016*, Villanova, United States.

[28] Platanitis, G., Pop-Iliev, R., Barari, A., 2012, "Development of a DSM-Based Methodology in an Academic Setting," Journal of Mechanical Design, **134**.

[29] Lee, S. G., Ong, K. L., Khoo, L. P., 2004, "Control and Monitoring of Concurrent Design Tasks in a Dynamic Environment," Journal of Mechanical Design, **12**(1), pp 59 - 66.

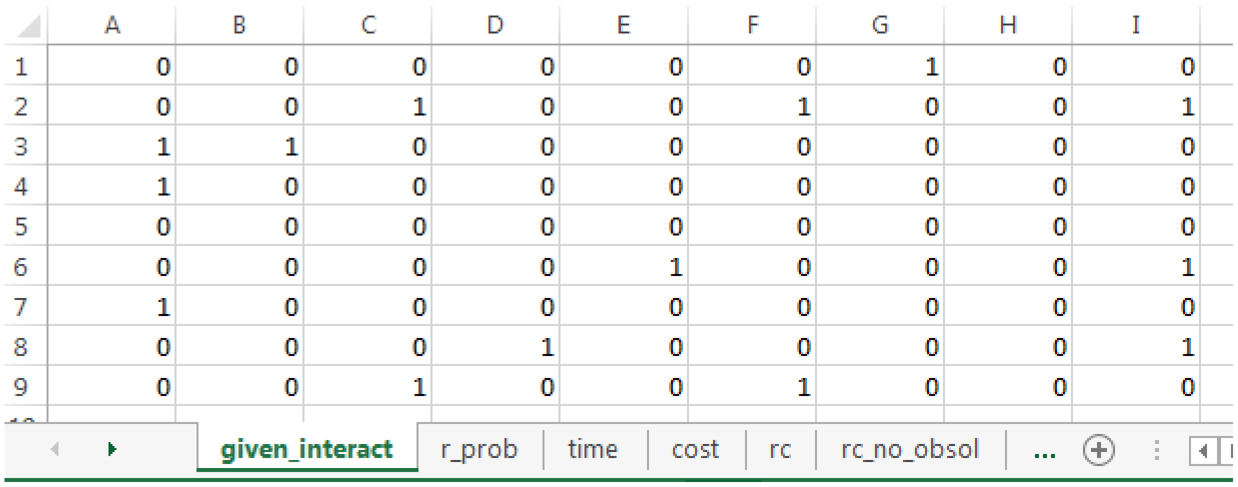
[30] Smith, R. P., Eppinger, S. D., 1997, "Identifying Controlling Features of Engineering Design Iteration," Management Science, **43**(3), pp 276 - 293.

APPENDICES

A. Program Codes For Methodology

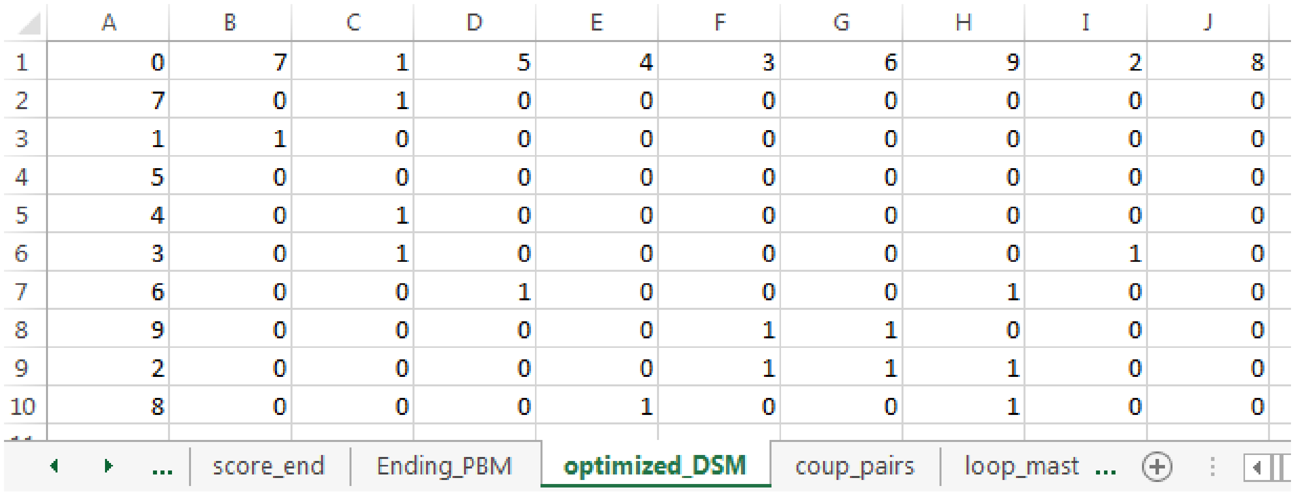
A.1. Inputs For Program and Extra Notes

Figures A.1 through A.5 show how the needed inputs for the simulation in the case study are made in Excel. The file name that contains all the inputs is shown in codes shown later in the section. The sheet names for each input are underlined and shown in green. If an unsolved DSM considering task precedence is being used, there is no need to add headings, open and execute "starter.m". The headings in Excel that show element location will be used as the headings in for the DSM. Treat the column headings as the matching number in the English alphabet (A is 1, B is 2, C is 3, etc.). In either case of using unsolved or solved DSM considering task precedence, the DSMRStart must be configured to match the headings in Excel. If a solved DSM considering task precedence is being used, open and execute "skipopt.m".



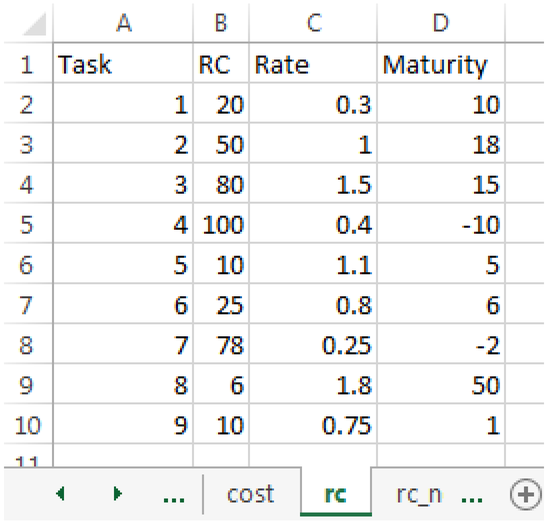
	A	B	C	D	E	F	G	H	I
1	0	0	0	0	0	0	1	0	0
2	0	0	1	0	0	1	0	0	1
3	1	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	1
7	1	0	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	1
9	0	0	1	0	0	1	0	0	0

Figure A.1: Excel Input For Unsolved DSMTS For Case Study.



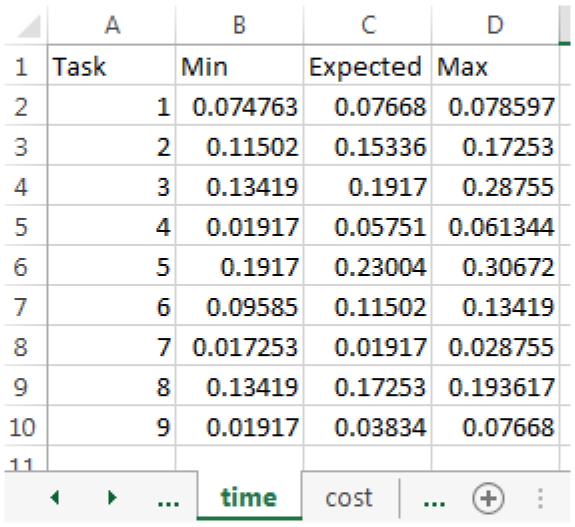
	A	B	C	D	E	F	G	H	I	J
1	0	7	1	5	4	3	6	9	2	8
2	7	0	1	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0
4	5	0	0	0	0	0	0	0	0	0
5	4	0	1	0	0	0	0	0	0	0
6	3	0	1	0	0	0	0	0	1	0
7	6	0	0	1	0	0	0	1	0	0
8	9	0	0	0	0	1	1	0	0	0
9	2	0	0	0	0	1	1	1	0	0
10	8	0	0	0	1	0	0	1	0	0

Figure A.2: Excel Input/Output For Solved DSMTS For Case Study.



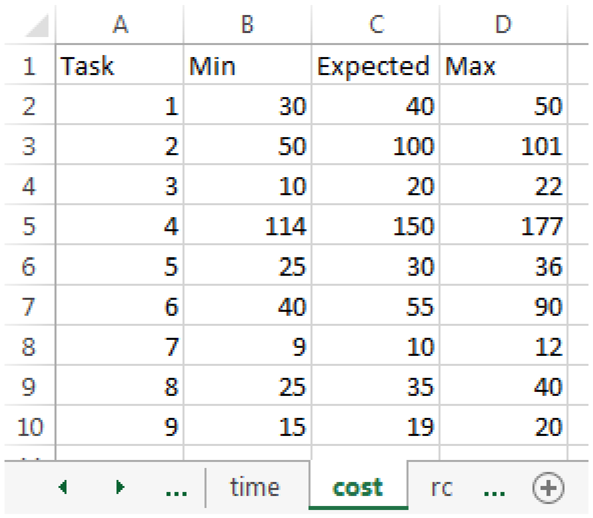
	A	B	C	D
1	Task	RC	Rate	Maturity
2	1	20	0.3	10
3	2	50	1	18
4	3	80	1.5	15
5	4	100	0.4	-10
6	5	10	1.1	5
7	6	25	0.8	6
8	7	78	0.25	-2
9	8	6	1.8	50
10	9	10	0.75	1

Figure A.3: Excel Input For Rework Coefficient, Rate, And Maturity Of Dominant Technologies Used To Complete Task For Case Study.



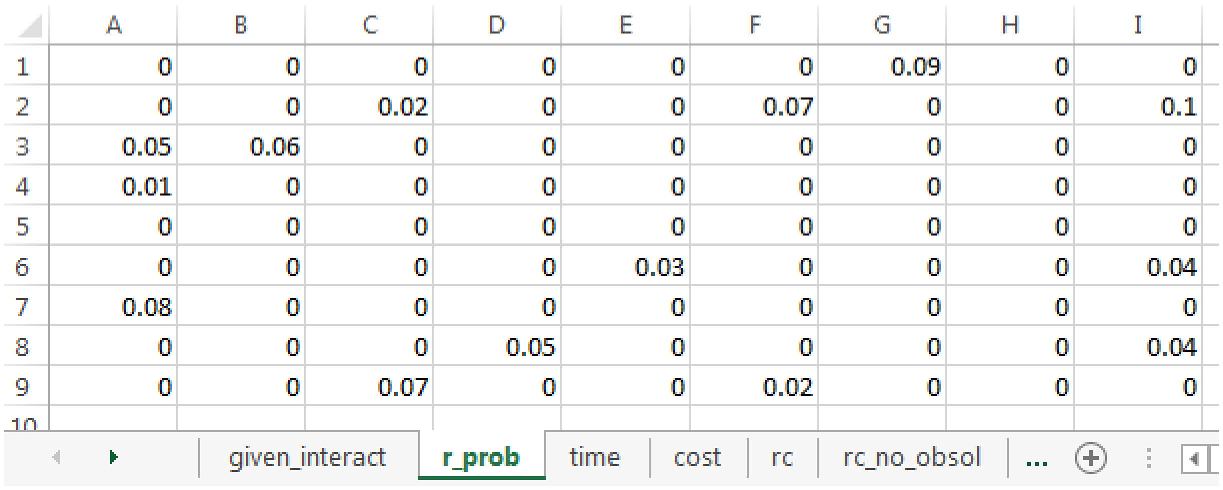
	A	B	C	D
1	Task	Min	Expected	Max
2	1	0.074763	0.07668	0.078597
3	2	0.11502	0.15336	0.17253
4	3	0.13419	0.1917	0.28755
5	4	0.01917	0.05751	0.061344
6	5	0.1917	0.23004	0.30672
7	6	0.09585	0.11502	0.13419
8	7	0.017253	0.01917	0.028755
9	8	0.13419	0.17253	0.193617
10	9	0.01917	0.03834	0.07668

Figure A.4: Excel Input For Duration Of Tasks For Case Study.



	A	B	C	D
1	Task	Min	Expected	Max
2	1	30	40	50
3	2	50	100	101
4	3	10	20	22
5	4	114	150	177
6	5	25	30	36
7	6	40	55	90
8	7	9	10	12
9	8	25	35	40
10	9	15	19	20

Figure A.5: Excel Input For Costs Of Tasks For Case Study



	A	B	C	D	E	F	G	H	I
1	0	0	0	0	0	0	0.09	0	0
2	0	0	0.02	0	0	0.07	0	0	0.1
3	0.05	0.06	0	0	0	0	0	0	0
4	0.01	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0.03	0	0	0	0.04
7	0.08	0	0	0	0	0	0	0	0
8	0	0	0	0.05	0	0	0	0	0.04
9	0	0	0.07	0	0	0.02	0	0	0
10									

Figure A.6: Excel Input For DSMRStart For Case Study.

A.2. Codes for “starter.m”

```
% The purpose of the code is to be the starting point.
% The following steps will be used...
%     1) Optimize given interaction DSM with "optimizer."
%     2) Examine the optimized DSM for task precendence. If is
   acceptable,
%         go to Step 3, otherwise go back to Step 1.
%     3) Loops (or circuits) are determined with "circuiter."
%     4) Simulations are done by "simulator."

clear all
close all
clc

while 1 == 1
    optimizer %run optimizer

    % Give choice to accept optimized dsm
    choice = menu('Optimization complete. Is optimized DSM
        acceptable?','No','Yes');

    if choice == 1
        continue %redo optimization
    end

    if choice == 2
        % Give message to start finding loops
```

```
          circuiterstart = ['Now looking for loops ...'];
          disp(circuiterstart)
          circuiter %run circuiter
          break
     end
end
```

A.3. Codes for “evaluate.m”

```
function [ obf ] = evaluate( triali ,trialj)
obf = 0; %(objective function of optimizer)

global info; %from optimizer

% Objective is to minimize feedback by minimizing distance
% from diagonal.
obf = sum(abs((trialj-triali).*info(:,5)));

end
```

A.4. Codes for “optimizer.m”

```
% For lower is better (depend down / provide across) notation
% Also called IR/FAD (Input is Row/ Feedback Above Diagonal)
    notation
% Looks at PBM then score. Use To obtain DSMTP

clear all
clc

%% _____Give message to start optimimization____
optstart = ['Starting Optimization ...'];
disp(optstart)

global best_info;
global best_dsm;
global best_order;
global popi_best;
global popj_best;
global v;
global info;
global cir_input;

%% ______Inputs from user _____
cir_input = 'rogers_case_study.xlsx'; %file that contains inputs
cir_output = 'rogers_case_study.xlsx'; %file that outputs are
    placed

```

```
%% ___Sorting data for optimization__
v = (xlsread(cir_input,'given_interact')); %read data from excel
    file
info = zeros(length(v)^2,6); %intitailize info

global master;
master = 1:length(v):length(v)^2; %used for referance in making
    info matrix

for m = 1:length(v):length(v)^2
    for k = 1:1:length(v)
    loc = find(master==m); %used for referance in making info
        matrix
    info(m:m+length(v)-1,1) = ones(length(v),1)*loc;  %depend (no
         change)
    info(m:m+length(v)-1,2) = 1:length(v); %provide (no change)
    info(:,3) = info(:,1); %row(i) (will change during
        optimization)
    info(:,4) = info(:,2); %column(j) (will change during
        optimization)
    end
end

for n = 1:length(v)^2
    info(n,5) = v(info(n,1),info(n,2)); %values from original dsm

    info(n,6) = (abs(info(n,4)-info(n,3))*info(n,5)); %original
        dsm scores
```

```
end

total_moves = sum(info(:,5)); %total moves

bm = 0; %backtracking moves

% If a move is above the diagonal, then it is a backtracking move
    .
for n = 1:length(v)^2
    if (info(n,4) > info(n,3)) && (info(n,5) ~= 0)
        bm = bm+info(n,5);
    end
end

Original_PBM = (bm/total_moves)*100; % percent backtracking moves

% Initialing vectors,
triali = zeros(1,length(v)^2)'; %trial for row

trialj = zeros(1,length(v)^2)'; %trial for column

gen_max = 100; %generations

% gen = 1:gen_max; scores = zeros(1,gen_max); %use for
    convergence tracking

pop_size = 100; %population size
popi = zeros(length(v)^2,pop_size); %initialing i's
```

```
popj = zeros(length(v)^2,pop_size); %initialing j's
cgscore = zeros(1,pop_size);  %initialing  current generation
    scores of children
cgpbm = zeros(1,pop_size);  %initialing  current generation of
    PBM children
lgscore = ones(1,pop_size)*sum(info(:,6)); %last generation score
     of children
lgpbm = ones(1,pop_size)*Original_PBM; %last generation PBM of
    children
%score of give dsm is staring point for comparsion

x1 = popi;
x2 = popj;

% Making stating point for population comparision.
% If better order cannot be found, then the orginal is given back
    .
for i = 1:pop_size
    x1(:,i) = info(:,3);
    x2(:,i) = info(:,4);
end

%% ___Solving DSM for task precedence___
for it = 1:gen_max


for p = 1:pop_size % generate initial test population
% _Mutation__
```

```
    mutmaster = randperm(length(v),length(v))'; %referance for
        mutation
    for m = 1:length(v):length(v)^2
    for k = 1:1:length(v)
    local = mutmaster(find(master==m)); %used for referance in
        making info matrix
    triali(m:m+length(v)-1) = ones(length(v),1)*local;  %triali
    trialj(m:m+length(v)-1) = mutmaster; %trialj
    end
    end

popi(:,p) = triali;
popj(:,p) = trialj;

end

% _Evaluating Fitness_

% PBM
score_ref = ones(1,pop_size)*1000;  % reference for pbm
    comparsion

for p = 1:pop_size
    triali = popi(:,p);
    trialj = popj(:,p);

    bm = 0;
    for n = 1:length(v)^2
    if (trialj(n) > triali(n)) && (info(n,5) ~= 0)
        bm = bm+info(n,5);
    end
    end
    cgpbm(p) = (bm/total_moves)*100;
end

% Scores
for p = 1:pop_size
    triali = popi(:,p);
    trialj = popj(:,p);

    [cgscore(p)] = feval('evaluate',triali,trialj);  %current
       generation
end

% Finding the current generation "elite"
[~,cgelite_loc] = find(cgpbm == min(cgpbm));

% Inserting pbms of current generation elite in reference
    locations
% Matching the scores for easier reference
for i = 1:length(cgelite_loc)
    score_ref(cgelite_loc(i)) = cgscore(cgelite_loc(i));

end

% Determining the "best"
```

```
[~,cgbest_loc] = find(score_ref == min(score_ref)); % best has
    lowest score of elite

% If current generation of children is better than the last, the
    current
% population is the kept, otherwise keep last generation.
if min(cgpbm) < min(lgpbm) && (min(score_ref) < min(lgscore))
     cgen_best_pbm = min(cgpbm)
     cgen_best_score = min( score_ref)
     sample_i = popi(:,cgbest_loc); %duplicates of  parents of
         best might exist
     sample_j = popj(:,cgbest_loc); %only one is needed
     popi_best = sample_i(:,1);  % parents of
     popj_best = sample_j(:,1);   % best children
     x1 = popi;
     x2 = popj;
else
    cgscore = lgscore;
    cgpbm = lgpbm;
    [~,cgelite_loc] = find(cgpbm == min(cgpbm));
    score_ref = ones(1,pop_size)*1000;
    for i = 1:length(cgelite_loc)
     score_ref(cgelite_loc(i)) = cgscore(cgelite_loc(i));
    end
    [~,cgbest_loc] = find(score_ref == min(score_ref));
    sample_i = x1(:,cgbest_loc); %duplicates of  parents of best
        might exist
    sample_j = x2(:,cgbest_loc); %only one is needed
    popi_best = sample_i(:,1);  % parents of
    popj_best = sample_j(:,1);   % best children
    cgen_best_pbm = min(cgpbm)
    cgen_best_score = min( score_ref)
end
lgscore = cgscore;
lgpbm = cgpbm;

end

%% _________Making dsm from info  ____________

best_info = zeros(length(v)+1,length(v)+1);

best_dsm = zeros(length(v),length(v));

org_order = 1:length(v);

best_order = zeros(length(v),1);

% Determine order from the population
% Example if popj_best is [4 1 2 3]
% then do task 1 forth, task 2 first, task 3 second, and task 4
     third
for i = 1:length(v)
     [r,~] = find(popj_best(1:length(v)) == i);
     best_order(i) = r;
end
```

```

% Filling in column headings
best_info(1,2:1+length(v)) = best_order(:,1);

% Filling in row headings
best_info(2:1+length(v),1) = best_order(:,1)';

% Filling dsm
for coord = 1:length(v)^2

    best_dsm(popi_best(coord),popj_best(coord)) = info(coord,5);

end

% Inserting best_dsm into best_info
best_info(2:end,2:end) = best_dsm;

% Display best order
best_order

% Display starting and ending scores and PBMs
score_start = sum(info(:,6))
Original_PBM

score_end = cgen_best_score
Ending_PBM = cgen_best_pbm

```

```
%% Export starting and ending scores, PBMs, and optimized DSM to
    excel
xlswrite(cir_output,score_start,'score_start')
xlswrite(cir_output,Original_PBM,'Original_PBM')

xlswrite(cir_output,score_end,'score_end')
xlswrite(cir_output,Ending_PBM,'Ending_PBM')

% Export optimal dsm to excel
xlswrite(cir_output,best_info,'optimized_DSM')
% Warning! Make sure that the Excel sheet is not exietant and
    file is
% closed before the progrom is executed.
```

A.5. Codes for “skipopt.m”

```
% This code is used for finding loops and doing simulations of
    time and
% cost given the solved DSM for task precedence and other
    appropiate data.

clear all
close all
clc

global v;
global best_info;
global best_dsm;
global best_order;
global master;
global popi_best;
global popj_best;
global cir_input;


%% ____Inputs from user_______
cir_input = 'rogers_case_study.xlsx'; %file that contains inputs
best_info = (xlsread(cir_input,'optimized_DSM'));

best_dsm = best_info(2:end,2:end);
v = best_dsm;
best_order = best_info(2:end,1);
```

```
master = 1:length(v):length(v)^2;

%% _____Making popi_best and popj_best________
popi_best = zeros(1,length(v)^2)';
popj_best = zeros(1,length(v)^2)';

% Remaking mutmaster
mutmaster = zeros(1,length(v))';
for i = 1:length(v)
     mutmaster(i) = find(best_order==i);
end

for m = 1:length(v):length(v)^2
     for k = 1:length(v)
           mut = mutmaster(find(master==m));
           popi_best(m:m+length(v)-1) = ones(length(v),1).*mut;
           popj_best(m:m+length(v)-1) = mutmaster;
     end
end

%% _____________Give message to start finding
    loops_________________
circuiterstart = ['Now looking for loops ...'];
disp(circuiterstart)
circuiter %run circuiter
```

A.6. Codes for “circuiter.m”

```
% The purpose of this code is to take a DSM and find the circuits
    (loops).

%% ____Inputs from user_______
cir_output = 'rogers_case_study.xlsx'; %file that outputs are
    placed

%% ____Get data from optimizer_____
global best_info;
global best_dsm;
global best_order;

%% _____Give data to simulater _______
global loop_ref
global loop_ref_named
global coup_pairs
global it_size
global loop_master

[l,~] = size(best_info); %dimensions of best_info

%% ____ Finding Coupled pairs_________

pair_ref = [];
for i = 2:l
    for j = 2:l
```

```
        if (best_info(i,j) == best_info(j,i)) && (best_info(i,j) ~= 0) && (best_info(j,i) ~= 0)
            pair_ref((i-1),1) = best_info(i,1);
            pair_ref((i-1),2) = best_info(j,1);
        end
    end
end

[l,w] = size(pair_ref);
if l ~= 0
    % Removing entries with zeros
    pair_ref( ~any(pair_ref,2), : ) = [];  %rows
    pair_ref( :, ~any(pair_ref,1) ) = [];  %columns

    % Finding and sorting duplicates
    for i = 1:length(pair_ref)
        sort_ref = pair_ref(i,1); %using first number in row as referance.
        [r,~] = find(sort_ref == pair_ref(:,2)); %find duplicate location
        if r >= i
            pair_ref(r,1) = pair_ref(r,2);
            pair_ref(r,2) = pair_ref(i,2);
        end
        % The duplicate is made look like original
    end
end

```

```
% Removing duplicates and display
coup_pairs = unique(pair_ref,'rows','stable')

% Export list of iterartive pairs to excel
[pair_size,~] = size(pair_ref);
if pair_size ~= 0
    xlswrite(cir_output,coup_pairs,'coup_pairs')
end
%% ____ Finding Interitve loops_________

% Making adjacency matrix
a = best_dsm; % starting point for making adjacency matrix

% Make non zero entries one
for i = length(best_dsm)
    for j = length(best_dsm)
        if a(i,j) ~= 0
            a(i,j) = 1;
        end
    end
end

loop_ref = zeros(length(best_dsm),length(best_dsm));
% ^^^ Referance for filling in loops. ^^^
% Also, rows are powers of a and columns are task.

for i = 2:length(best_dsm) % start at power 2
    a = best_dsm^i ;

    % Make non zero entries one in current power of a
    for j = 1:length(best_dsm) % row
        for k = 1:length(best_dsm) % column
            if a(j,k) ~= 0
                a(j,k) = 1;
            end
        end
    end

    % Search diagonals
    for j = 1:length(best_dsm)
        if a(j,j)==1
            loop_ref((i),j) = 1;
        end
    end
end

%%  ----Sorting data before simulations--------

% Finding standalone tasks
stand_alones = zeros(1,length(best_dsm));
% ^^^ Referance for stand alone tasks. 1 is yes. 0 is no. ^^^

for i = 1:length(best_dsm)
    if sum(loop_ref(:,i)) == 0
        stand_alones(1,i) = 1;
    end
```

```
end

% Removing connected loops and its copies
index_ref = 1:length(best_dsm);
index_ref = index_ref';

% Overall, a row in loop_ref has connected loops if the sum of a
    row and
% in row's number do not match. Example row 2 in loop_ref has a
    sum of 6.
% this means that they are 3 loops of 2, not 1 loop of 6.
    Suspicion can
% arise if the loop_ref sum and row number match. Example is a
    row
% with sum of 5 but having a loop of size 2 and another of size
    3.

% For row and sum are equal
for i = 3:length(loop_ref)
    start_indx = 0; %"start in loop"
    end_indx = 0; %"last in loop"
    for c = 1:length(best_dsm)
        if loop_ref(i,c) == 1
            start_indx = c;
            break
        end
    end
    for d = length(best_dsm):-1:1
```

```
        if loop_ref(i,d) == 1
            end_indx = d;
            break
        end
    end
    if best_dsm(c,d) == 0 && c < d
        loop_ref(i,:) = zeros(1,length(best_dsm));
    end
end

% For sum and row mismatch
for i = 2:length(best_dsm)
    if sum(loop_ref(i,:)) > i
        loop_ref(i,:) = zeros(1,length(best_dsm));
    end
end

% If all loops are removed (multiple loops of equal size exist) ...
[lc,~] = size(loop_ref);
if sum(loop_ref) == 0;
    % loop_ref = zeros(length(best_dsm),length(best_dsm));
    k = 1; %row of loop_ref for referance in reconstructing loop master
    for i = 1:length(best_dsm)
        for j = 1:length(best_dsm)
            if (i < j) && best_dsm(i,j) == 1
                loop_ref(k,i:j) = ones(1,(j-i+1));
                k = k+1;
            end
        end
    end
end

% Adding coupled pairs (if any) to loop_ref
[pr,pc] = size(coup_pairs);
[r,~] = size(loop_ref);

if pr ~= 0
    for i = 1:pr
        [~,pair_loc1] = find(best_order' == coup_pairs(i,1));
        [~,pair_loc2] = find(best_order' == coup_pairs(i,2));
        loop_ref(r+i,pair_loc1) = 1;
        loop_ref(r+i,pair_loc2) = 1;
    end
end

loop_ref = unique(loop_ref,'rows','stable');

% Removing rows with zeros
loop_ref2 = loop_ref;
loop_ref2(~any(loop_ref,2),:) = [];
loop_ref = loop_ref2;


[r,c] = size(loop_ref);
```

```

% Give name to loops
loop_ref_named = loop_ref;
for i = 1:r
     loop_ref_named(i,:) = loop_ref(i,:).*best_order';
end

%% ----- Removing "embedded" loops -----------
[r,~] = size(loop_ref);

row_check = zeros(r,r); %referance for checking

% If the loop in loop_ref_named row, curr_r is part of
    loop_ref_named
% colunm new_r, then row_check(curr_r,new_r) is made to be 1.

for curr_r = 1:r
    for new_r = 1:r
        combo = loop_ref(curr_r,:)+loop_ref(new_r,:);
        meeting = find(combo == 2);
        [~,inter] = size(meeting);
        if inter ~= 0 % If any 2's exist, then one loop in
           another
           sum_curr = sum(loop_ref(curr_r,:));
           sum_ref = sum(loop_ref(new_r,:));
           if curr_r == new_r % A row always intersects with
              self.
              row_check(curr_r,new_r) = 1;
```

```
            elseif sum_curr < sum_ref % Loop curr_r inside loop new_r.
                row_check(curr_r,new_r) = 1;
            end
        end
    end
end

% Now the if the sum of a row in the row_check matrix is more than one,
% then the loop is part of another and is removed
row_check_sum = sum(row_check,2);

loop_ref((row_check_sum > 1),:) = [];
loop_ref_named((row_check_sum > 1),:) = [];

%% ____Constucting master for loops_____

[r,~] = size(loop_ref);
it_size = r + 1;
% ^^^ Sum of number of loops, coupled pairs, and one for stand alone tasks

loop_master = zeros(it_size,length(best_order));

% Adding loops
loop_master((1:r),:) = loop_ref((1:r),:);

```

```
% Adding stand alone tasks
loop_master(it_size,:) = stand_alones;

%% _____Making loop master for export_______

loop_master_ex = zeros(it_size+1,length(v)+1); %loop master for
    export
loop_master_ex(1,2:end) = best_order; % Adding column heading
loop_master_ex(2:end,2:end) = loop_master; % Adding info

% Adding labels for loop classes. Numbers are for class
% Class 1 is clean loop with 3 or more tasks
% Class 2 is clean, coupled pair
% Class 3 is stand alone task(s)
for i = 1:it_size
    if sum(loop_master(i,:)) >= 3
        loop_master_ex(i+1,1) = 1;
    end
    if sum(loop_master(i,:)) == 2
        loop_master_ex(i+1,1) = 2;
    end
    if i == it_size
        loop_master_ex(i+1,1) = 3;
    end
end

% Print loop_master_ex
loop_master_ex
```

```

% Export name of loop_master to excel
xlswrite(cir_output,loop_master_ex,'loop_master')

% Determining sizes
loop_sizes = zeros(it_size,1);

for i = 1:it_size
    loop_sizes(i) = sum(loop_master(i,:));
end

% Print sizes of loops and notice
loop_sizes

notice = 'Note that the last sum is for stand alone tasks.';
disp(notice)

% Export name of loop sums to excel
xlswrite(cir_output,loop_sizes,'loop_sizes')

%% ____Give message to simulations__________

simstart = ['Now starting simulations ...'];
disp(simstart)
simulator %start simulations
```

A.7. Codes for “simulator.m”

```
% The purpose of this code is simulate project time and cost.

%% ______User inputs _____

life_span = 20; %life of project

int_size = 5; %size of interval for plots

mc_trial = 10000; %number of runs in each monte carlo simulation

sim_output = 'Orignial_DSMRS.xlsx'; %where to put outputs

%% ____Get data from optimizer or skipopt_____

global best_info;
global best_dsm;
global best_order;
global v
global popi_best;
global popj_best;
global master
global cir_input;

%____Get data from circuiter and excel files_____

global it_size
global loop_master

% Taking loop master from excel
loop_master = (xlsread(cir_input,'loop_master')); %read data from
    excel file
loop_master = loop_master(2:end,2:end); %removing headings

% Taking rework prob and impact infomation from excel and
    reoganize

% Probability of rework
r_prob = (xlsread(cir_input,'r_prob')); %read data from excel
    file

% Times
time = (xlsread(cir_input,'time')); %read data from excel file

% Cost
cost = (xlsread(cir_input,'cost')); %read data from excel file

% Rework Coeffiecent, Alpha, and Beta
rc = (xlsread(cir_input,'rc')); %read data from excel file

%% _______Reogranizing given to match optimal _________

% Probability of rework
prob_info = zeros(length(v)^2,5); %intitailize info

```

```
for m = 1:length(v):length(v)^2
    for k = 1:1:length(v)
    loc = find(master==m); %used for referance in making info
        matrix
    prob_info(m:m+length(v)-1,1) = ones(length(v),1)*loc;  %
        provide (no change)
    prob_info(m:m+length(v)-1,2) = 1:length(v); %depend (no
        change)
    prob_info(:,3) = prob_info(:,1); %row(i) (will change during
        optimization)
    prob_info(:,4) = prob_info(:,2); %column(j) (will change
        during optimization)
    end
end

for n = 1:length(v)^2
    prob_info(n,5) = r_prob(prob_info(n,1),prob_info(n,2)); %
        values from original dsm

end

% Inserting better population into info
prob_info(:,3) = popi_best;
prob_info(:,4) = popj_best;

new_prob = best_info;

% Filling dsm
```

```
for coord = 1:length(v)^2
    DSMR_start_info(popi_best(coord),popj_best(coord)) =
        prob_info(coord,5);
end

new_prob(2:end,2:end) = DSMR_start_info;

%Export DSMR_start (DSM for Rework at time zero of their tech
    releases )
xlswrite(sim_output,new_prob,'DSMR_start')

% Reoganizing given time/cost estimates and improvment to match
    optimal

opt_time = time; %time estimates in optimal order

opt_cost = cost; %cost estimates in optimal order
% Column 1 is min. 2 is likely. 3 is max for time and cost.

opt_rc = rc; %rework estimates in optimal order

for r = 1:length(v)
    r_ref = find(best_order == r);
    opt_time (r_ref,:) = time(r,:);
    opt_cost (r_ref,:) = cost(r,:);
    opt_rc (r_ref,:) = rc(r,:);
end
```

```
rc_horz = opt_rc(:,2)'/100; %want vector to be horizontal
    starting with given

% Making DSMRate(DSM for Rates). Made of rates

DSMRate = best_info;

DSMRate_info = best_dsm;

 % Make non zero entries one in DSMRate_info for proper filling
    for j = 1:length(best_dsm) % row
        for k = 1:length(best_dsm) % column
            if DSMRate_info(j,k) ~= 0
               DSMRate_info(j,k) = 1;
            end
        end
    end

%  Filling in DSMRate
for i = 1:length(best_dsm)
    DSMRate_info(i,:) = DSMRate_info(i,:).*opt_rc(:,3)';
end

DSMRate(2:end,2:end) = DSMRate_info;

xlswrite(sim_output,DSMRate,'DSMRate')

% Making DSMMAT(DSM for Maturity). Made of inflection points
```

```

DSMMAT = best_info;

DSMMAT_info = best_dsm;

% Make non zero entries one in DSMO_info for proper filling
    for j = 1:length(best_dsm) % row
        for k = 1:length(best_dsm) % column
            if DSMMAT_info(j,k) ~= 0
               DSMMAT_info(j,k) = 1;
            end
        end
    end

% Filling in DSMMAT
for i = 1:length(best_dsm)
    DSMMAT_info(i,:) = DSMMAT_info(i,:).*opt_rc(:,4)';
end

DSMMAT(2:end,2:end) = DSMMAT_info;

xlswrite(sim_output,DSMMAT,'DSMMAT')

%% ________Monte Carlo Simulaion Runs____________

% Make time and cost vectors
mc_time_v = zeros(mc_trial,length(v)); % want time vectors
    horizontal
```

```
mc_cost_v = zeros(mc_trial,length(v)); % want cost vectors
    horizontal

mc_master = zeros(mc_trial,3); %master for time and cost
    simulation
% ^^^ Column 1 is simulation. column 2 is time. column 3 is cost
    ^^^.

for curr_time = 0:int_size:life_span %for each time interval

    curr_DSMR_info = best_dsm; %current dsm for rework

    % Evaluating current dsm for rework
    for i = 1:length(v)
        for j = 1:length(v)
            if best_dsm(i,j) ~= 0
                ll = DSMR_start_info(i,j);
                %^^ Lower limit (ll) is starting p(rework) ^^

                a = DSMRate_info(i,j);
                %^^ Alpha is rate of obsolecence from DSMRate ^^

                t = curr_time; %time is time in project life

                b = DSMMAT_info(i,j);
                % ^^When maturity of tasks's tech (beta) occurs
                    ^^

                curr_DSMR_info(i,j) = ll + ((1-ll)./(1+exp(-a*(t-
                    b))));
                %^^ Based on logistics curve ^^
            end
        end
    end

    % Getting current DSMR for display in Excel file
    curr_DSMR = best_info;
    curr_DSMR(2:end,2:end) = curr_DSMR_info;

    time4export = num2str(curr_time);
    xlswrite(sim_output,curr_DSMR,time4export)

for sim = 1:mc_trial
    % Filling in time and cost vectors
    for i = 1:length(v)
        % Time
        min_t = opt_time(i,2); %min estimate
        likely_t = opt_time(i,3); %likely estimate
        max_t = opt_time(i,4); %max estimate

        % Cost
        min_c = opt_cost(i,2); %min estimate
        likely_c = opt_cost(i,3); %likely estimate
        max_c = opt_cost(i,4); %max estimate

        % Spread of distributions
        spd_t = makedist('Triangular','a',min_t,'b',likely_t ,'c',
           max_t); %time
        spd_c = makedist('Triangular','a',min_c,'b',likely_c ,'c',
           max_c); %cost

        % Make random number from given spreads
        mc_time_v(sim,i) = random(spd_t,1,1);
        mc_cost_v(sim,i) = random(spd_c,1,1);
    end

    % Fill trial master
    redo = rand(length(v),length(v)); %probability of rework
       indicator
    loop_t = zeros(it_size-1,1); %time for set of loops
    loop_c = zeros(it_size-1,1); %cost for set of loops
    mc_t_curr =  mc_time_v(sim,:); %current time from monte carlo
        list
    mc_c_curr =  mc_cost_v(sim,:); %current cost from monte carlo
        list
    order1_mult = zeros(1,length(v)); %for multiples of 1st order
        rework

    % Determining time and cost 0th order rework
    order0_time = sum(mc_t_curr); %first pass time
    order0_cost = sum(mc_c_curr); %first pass cost

    % Determining time and cost 1st order rework
    for r = 1:length(v) %going through rework dsm
```

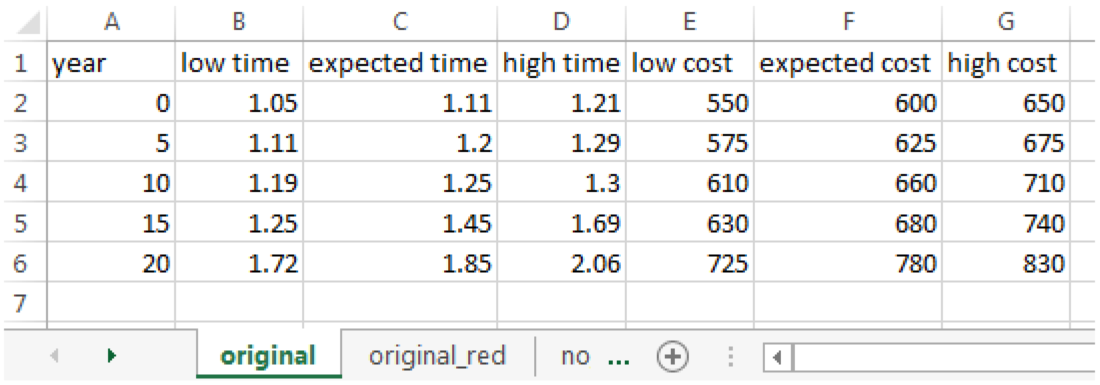
```
        for c = 1:length(v)
            if (r > c) && (redo(r,c) < curr_DSMR_info(r,c))
                % If redo is less than given probablity and is a forword
                % moving move, only single tasks have to be repeated.
                order1_mult(1,c) = order1_mult(1,c)+1;
            end
        end
     end
     order1_time = sum(order1_mult.*mc_t_curr.*rc_horz);
     order1_cost = sum(order1_mult.*mc_c_curr.*rc_horz);

    % Determining time and cost of 2nd order rework
    for i = 1:(it_size -1)
        for r = 1:length(v) %going through rework dsm
            for c = 1:length(v) %r = row, c = column
                if (r < c) && (redo(r,c) < curr_DSMR_info(r,c))
                    % If redo is less than given probablity and is a
                    % backtracking move, rework of an entire loop has to
                    % be done.
                    loop_t(i) = loop_t(i) + sum(loop_master(i,r:c).*mc_t_curr(r:c).*rc_horz(r:c));
                    loop_c(i) = loop_c(i) + sum(loop_master(i,r:c).*mc_c_curr(r:c).*rc_horz(r:c));
                end
            end
        end
        order2_t_tot = sum(loop_t); %total loop 2nd order rework time
        order2_c_tot = sum(loop_c); %total loop 2nd order rework cost
   end
    mc_master(sim,1) = sim; %simulation number
    mc_master(sim,2) = order0_time+order1_time+order2_t_tot; % total process time
    mc_master(sim,3) = order0_cost+order1_cost+order2_c_tot; % total process cost

end

%__Make bivariate histograms__

% 2D
figure
data = [mc_master(:,2),mc_master(:,3)];
hist3(data)

n = hist3(data,[25 25]); %Bin size for time and cost is 25
n1 = n';
n1(size(n,1) + 1, size(n,2) + 1) = 0;

xb = linspace(min(data(:,1)),max(data(:,1)),size(n,1)+1);
yb = linspace(min(data(:,2)),max(data(:,2)),size(n,1)+1);
```

```

h = pcolor(xb,yb,n1);

title(['Bivariate Histogram of Process Time and Cost at Year ',
    num2str(curr_time)])
xlabel('Time');
ylabel('Cost');
%colorbar %adds colarbar
shading interp % makes line go away

% 3D
figure
[n,c] = hist3(data,[25 25]); %Bin size for time and cost is 25
surf(c{1}, c{2}, n);
title(['Surface Bivariate Histogram of Process Time and Cost at
    Year ', num2str(curr_time)])
xlabel('Time');
ylabel('Cost');
zlabel('Frequency');
%colorbar

end

%% Give message showing simulations are complete
simend = ['Simulations are complete.'];
disp(simend)
```

B. Extra Code Used For Dissertation

B.1. Extra Notes For Using Code That Plots Distribution Motion

Figure A.7 show how to enter the distribution bounds into Excel.



	A	B	C	D	E	F	G
1	year	low time	expected time	high time	low cost	expected cost	high cost
2	0	1.05	1.11	1.21	550	600	650
3	5	1.11	1.2	1.29	575	625	675
4	10	1.19	1.25	1.3	610	660	710
5	15	1.25	1.45	1.69	630	680	740
6	20	1.72	1.85	2.06	725	780	830
7							

FIGURE A.7: Excel Input For Distribution Bounds.

B.2. Code For Plotting Distribution Motion

```
% The purpose of this code is plot the bounds of time/cost
    distribution
% as time passes in the form of lines (one for duration and
    aonther for
% cost). This code is important to the work, but is not linked to
     other
% codes. Suggest intgrating this code into the simulater for
    faster and
% more efficent processing.

clear all
close all
clc

%% -- Data Collection and User Inputs ---

% Time and cost bounds of procces during life
time_bound = 1.7;
cost_bound = 750;

% Excel data
data_loc = 'dist_bounds.xlsx'; %location of data (manually
    gathered)

% Data for first case will come in the following form ...
% Column 1 is year.
% Column 2 is low time.
% Column 3 is expected time (location of most concentration).
% Column 4 is high time.
% Column 5 is low cost.
% Column 6 is expected cost (location of most concentration).
% Column 7 is high cost.

original = (xlsread(data_loc,'original')); %good design, with
    obsolecence
no_obsol = (xlsread(data_loc,'no_obsol')); %good design, no
    obsolecence

% Data for RC and rate cases will come in the following form ...
% Column 1 is % change.
% Column 2 is low time.
% Column 3 is expected time (location of most concentration).
% Column 4 is high time.
% Column 5 is low cost.
% Column 6 is expected cost (location of most concentration).
% Column 7 is high cost.

rc_0 = (xlsread(data_loc,'rc_0'));   %changing rc year 0
rc_5 = (xlsread(data_loc,'rc_5'));   %changing rc year 5
rc_10 = (xlsread(data_loc,'rc_10')); %changing rc year 10
rc_15 = (xlsread(data_loc,'rc_15')); %changing rc year 15
rc_20 = (xlsread(data_loc,'rc_20')); %changing rc year 20

rates_0 = (xlsread(data_loc,'rates_0'));   %changing rates year 0
```

```
rates_5 = (xlsread(data_loc,'rates_5'));   %changing rates year 5
rates_10 = (xlsread(data_loc,'rates_10')); %changing rates year
    10
rates_15 = (xlsread(data_loc,'rates_15')); %changing rates year
    15
rates_20 = (xlsread(data_loc,'rates_20')); %changing rates year
    20


%% ___Data Processing For Original Vs No Obsolecence___

% Finding size of data
data_size = size(original); %size of data (years , 7)

% Original data
ot_bounds = original(:,4) - original(:,2); %differance in lower
    and
oc_bounds = original(:,7) - original(:,5); %upper bounds for time
     and cost

% No Obsolecence Data
not_bounds = no_obsol(:,4) - no_obsol(:,2); %differance in lower
    and
noc_bounds = no_obsol(:,7) - no_obsol(:,5); %upper bounds for
    time and cost

% Plotting original bounds
for t = 1:data_size(1)
```

```
    low_ot = original(t,2);
    ot_diff = ot_bounds(t);
    low_oc = original(t,5);
    oc_diff = oc_bounds(t);

    rectangle('position',[low_ot, low_oc, ot_diff, oc_diff])
    hold on
end

% Plotting original expected values
plot(original(:,3),original(:,6),'k-o')

for t = 1:data_size(1) %adding labels
    o_label1 = '\leftarrow ';
    o_label2 = num2str(original(t,1));
    o_label = strcat(o_label1,o_label2);
    text(original(t,3),original(t,6),o_label)
end

% Making proper axis bounds (Have to do only once)
ll_time = min(original(1,2),no_obsol(1,2)); %lower limit for time
ul_time = max(original(end,4),no_obsol(end,4)); %upper limit for
    time
ll_time = ll_time - ll_time*.1; %want axis bounds to be slighty
    larger than
ul_time = ul_time + ul_time*.1; %distribution bounds

ll_cost = min(original(1,5),no_obsol(1,5)); %lower limit for cost
```

```
ul_cost = max(original(end,7),no_obsol(end,7)); %upper limit for
    cost
ll_cost = ll_cost - ll_cost*.1; %want axis bounds to be slighty
    larger than
ul_cost = ul_cost + ul_cost*.1; %distribution bounds

axis([ll_time,ul_time,ll_cost,ul_cost])

% Adding time/cost bounds (A-B is time bound. C-D is cost bound)
A = [time_bound,time_bound]; %(x1,x2)
B = [ul_cost,time_bound]; %(y1,y2)
hold on
line(A,B,'Color','k','LineStyle','--')

C = [0,ul_time]; %(x1,x2)
D = [cost_bound,cost_bound]; %(y1,y2)
hold on
line(C,D,'Color','k','LineStyle','--')

% Adding labels to plot
title('Areas of Concern For Time and Cost of Process With
    Obsolecence')
xlabel('Time');
ylabel('Cost');

% Plotting no obsolecence bounds
figure

```

```
for t = 1:data_size(1)
    low_not = no_obsol(t,2);
    not_diff = not_bounds(t);
    low_noc = no_obsol(t,5);
    noc_diff = noc_bounds(t);

    rectangle('position',[low_not, low_noc, not_diff, noc_diff],'
        EdgeColor','r')
    hold on
end

% Plotting no obsolecence expected values
plot(no_obsol(:,3),no_obsol(:,6),'r-o')

for t = 1:data_size(1) %adding labels
    no_label1 = '\leftarrow ';
    no_label2 = num2str(no_obsol(t,1));
    no_label = strcat(no_label1,no_label2);
    text(no_obsol(t,3),no_obsol(t,6),no_label)
end

axis([ll_time,ul_time,ll_cost,ul_cost])

% Adding time/cost bounds (A-B is time bound. C-D is cost bound)
hold on
line(A,B,'Color','k','LineStyle','--')

hold on
```

```
line(C,D,'Color','k','LineStyle','--')

% Adding labels to plot
title('Areas of Concern For Time and Cost of Process Without Obsolecence')
xlabel('Time');
ylabel('Cost');

%% __Data Processing For Original Vs Changing RCs__

plotstyle = {'-.or','-.ob','-.ok','-.+r','-.+b'}; %colors

% Determining plot bounds. Want to keep plot bounds constant to watch the
% distribution bounds "creep" as obsolecence sets in.
plot_time_min = min(rc_0(:,2)); %lower bound of x-axis of all time plots
plot_time_max = max(rc_20(:,4)); %upper bound of x-axis of all time plots
plot_cost_min = min(rc_0(:,5)); %lower bound of x-axis of all cost plots
plot_cost_max = max(rc_20(:,7)); %upper bound of x-axis of all cost plots

y_min = min(rc_0(:,1))*1.2;
y_max = max(rc_0(:,1))*1.2;

[l,~] = size(rc_0); %want number of lines.
```

```

% Year 0
figure %(for duration)
for dur = 1:1:l
    t_xs = rc_0(dur,2:4); %x-coordinates of line
    t_ys = ones(1,3)*rc_0(dur,1); %y-coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Time of Different RC
    Values at Year 0')
time_lim_bott = [time_bound,time_bound]; %bottom of line for
    duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
    c_xs = rc_0(cost,5:7); %x-coordinates of line
    c_ys = ones(1,3)*rc_0(cost,1); %y-coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
```

```
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Cost of Different RC
    Values at Year 0')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for
    duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')

% Year 5
figure %(for duration)
for dur = 1:1:l
    t_xs = rc_5(dur,2:4); %x-coordinates of line
    t_ys = ones(1,3)*rc_5(dur,1); %y-coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Time of Different RC
    Values at Year 5')
time_lim_bott = [time_bound,time_bound]; %bottom of line for
    duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
```

```
    c_xs = rc_5(cost,5:7); %x-coordinates of line
    c_ys = ones(1,3)*rc_5(cost,1); %y-coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Cost of Different RC
    Values at Year 5')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for
    duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')

% Year 10
figure %(for duration)
for dur = 1:1:l
    t_xs = rc_10(dur,2:4); %x-coordinates of line
    t_ys = ones(1,3)*rc_10(dur,1); %y-coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Time of Different RC
    Values at Year 10')
```

```
time_lim_bott = [time_bound,time_bound]; %bottom of line for duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
      c_xs = rc_10(cost,5:7); %x-coordinates of line
      c_ys = ones(1,3)*rc_10(cost,1); %y-coordinates of line
      plot(c_xs,c_ys,plotstyle{cost});
      hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Cost of Different RC Values at Year 10')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')

% Year 15
figure %(for duration)
for dur = 1:1:l
      t_xs = rc_15(dur,2:4); %x-coordinates of line
      t_ys = ones(1,3)*rc_15(dur,1); %y-coordinates of line
      plot(t_xs,t_ys,plotstyle{dur});
```

```
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Time of Different RC
    Values at Year 15')
time_lim_bott = [time_bound,time_bound]; %bottom of line for
    duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:1
    c_xs = rc_15(cost,5:7); %x-coordinates of line
    c_ys = ones(1,3)*rc_15(cost,1); %y-coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Cost of Different RC
    Values at Year 15')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for
    duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')
```

```

% Year 20
figure %(for duration)
for dur = 1:1:l
      t_xs = rc_20(dur,2:4); %x-coordinates of line
      t_ys = ones(1,3)*rc_20(dur,1); %y-coordinates of line
      plot(t_xs,t_ys,plotstyle{dur});
      hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Time of Different RC
    Values at Year 20')
time_lim_bott = [time_bound,time_bound]; %bottom of line for
    duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
      c_xs = rc_20(cost,5:7); %x-coordinates of line
      c_ys = ones(1,3)*rc_20(cost,1); %y-coordinates of line
      plot(c_xs,c_ys,plotstyle{cost});
      hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
```

```
ylabel('Change in RC (%)')
title('Comparing Distribution of Process Cost of Different RC
    Values at Year 20')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for
    duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')

%% __Data Processing For Original Vs Changing Rates__

plotstyle = {'-.or','-.ob','-.ok','-.+r','-.+b'}; %colors

% Determining plot bounds. Want to keep plot bounds constant to
    watch the
% distribution bounds "creep" as obsolecence sets in.
plot_time_min = min(rates_0(:,2)); %lower bound of x-axis of all
    time plots
plot_time_max = max(rates_20(:,4)); %upper bound of x-axis of all
     time plots
plot_cost_min = min(rates_0(:,5)); %lower bound of x-axis of all
    cost plots
plot_cost_max = max(rates_20(:,7)); %upper bound of x-axis of all
     cost plots

y_min = min(rates_0(:,1))*1.2;
y_max = max(rates_0(:,1))*1.2;

[l,~] = size(rates_0); %want number of lines.
```

```

% Year 0
figure %(for duration)
for dur = 1:1:l
    t_xs = rates_0(dur,2:4); %x-coordinates of line
    t_ys = ones(1,3)*rates_0(dur,1); %y-coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Time of Different Rates at Year 0')
time_lim_bott = [time_bound,time_bound]; %bottom of line for duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
    c_xs = rates_0(cost,5:7); %x-coordinates of line
    c_ys = ones(1,3)*rates_0(cost,1); %y-coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
```

```
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Cost of Different Rates
    at Year 0')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for
    duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')

% Year 5
figure %(for duration)
for dur = 1:1:l
    t_xs = rates_5(dur,2:4); %x-coordinates of line
    t_ys = ones(1,3)*rates_5(dur,1); %y-coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Time of Different Rates
    at Year 5')
time_lim_bott = [time_bound,time_bound]; %bottom of line for
    duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
```

```
    c_xs = rates_5(cost,5:7); %x-coordinates of line
    c_ys = ones(1,3)*rates_5(cost,1); %y-coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Cost of Different Rates at Year 5')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')

% Year 10
figure %(for duration)
for dur = 1:1:l
    t_xs = rates_10(dur,2:4); %x-coordinates of line
    t_ys = ones(1,3)*rates_10(dur,1); %y-coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Time of Different Rates at Year 10')
```

```
time_lim_bott = [time_bound,time_bound]; %bottom of line for duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
    c_xs = rates_10(cost,5:7); %x-coordinates of line
    c_ys = ones(1,3)*rates_10(cost,1); %y-coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Cost of Different Rates at Year 10')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')

% Year 15
figure %(for duration)
for dur = 1:1:l
    t_xs = rates_15(dur,2:4); %x-coordinates of line
    t_ys = ones(1,3)*rates_15(dur,1); %y-coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Time of Different Rates
    at Year 15')
time_lim_bott = [time_bound,time_bound]; %bottom of line for
    duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','--')

figure %(for cost)
for cost = 1:1:l
    c_xs = rates_15(cost,5:7); %x-coordinates of line
    c_ys = ones(1,3)*rates_15(cost,1); %y-coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Cost of Different Rates
    at Year 15')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for
    duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')
```

```

% Year 20
figure %(for duration)
for dur = 1:1:l
    t_xs = rates_20(dur,2:4); %x−coordinates of line
    t_ys = ones(1,3)*rates_20(dur,1); %y−coordinates of line
    plot(t_xs,t_ys,plotstyle{dur});
    hold on
end
axis([plot_time_min,plot_time_max,y_min,y_max]);
xlabel('Time (Minimum, Expected, Maximum)')
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Time of Different Rates
    at Year 20')
time_lim_bott = [time_bound,time_bound]; %bottom of line for
    duration bound
time_lim_top = [y_min,y_max];  %top of line for duration bound
line(time_lim_bott,time_lim_top,'Color','k','LineStyle','−−')

figure %(for cost)
for cost = 1:1:l
    c_xs = rates_20(cost,5:7); %x−coordinates of line
    c_ys = ones(1,3)*rates_20(cost,1); %y−coordinates of line
    plot(c_xs,c_ys,plotstyle{cost});
    hold on
end
axis([plot_cost_min,plot_cost_max,y_min,y_max]);
xlabel('Cost (Minimum, Expected, Maximum)')
```

```
ylabel('Change in Rates (%)')
title('Comparing Distribution of Process Cost of Different Rates
    at Year 20')
cost_lim_bott = [cost_bound,cost_bound]; %bottom of line for
    duration bound
cost_lim_top = [y_min,y_max];  %top of line for duration bound
line(cost_lim_bott,cost_lim_top,'Color','k','LineStyle','--')
```

VITA

Gershom Kwaku Obeng
Department of Mechanical and Aerospace Engineering
238 Kaufman Hall
Norfolk, VA 23529-0247

Education and Experience:
Bachelor of Science (Mechanical Engineering), Old Dominion University, May 2011
Minor: Engineering Management

Master of Science (Mechanical Engineering), Old Dominion University, December 2013
Option Area: Design and Manufacturing

Currently working at Old Dominion University as a graduate research assistant (areas: life cycle and naval engineering). Currently researching process planning methods while considering obsolescence. Did past work on a project sponsored by the Naval Engineering Education Consortium (NEEC). Was on a research team that developed models for the life cycle cost of naval systems. Taught an engineering graphics class at Thomas Nelson Community College.

Publications:
Obeng, G. K., Bao, H. P., 2016, "Design Structure Matrix with Consideration of Technological Obsolescence for Managing Complex Engineering Projects, Advanced Technology Symposium 2016, Villanova, Pennsylvania.

Obeng, G. K., Bao, H. P., 2016, "Integrating Technological Obsolescence in Quantitative Forecasting of Memory Device Failure Cost, International Journal of Engineering and Mathematical Modeling, 3(1), Pages 17-26.

Obeng, G. K., Bao, H. P., 2014, "Consideration of Technological Obsolescence in Quantitative Forecasting and Economic Life Analysis, International Conference on Engineering and Applied Sciences Optimization, Kos Island, Greece.

Obeng, G. K., 2013, "Design and Implementation of Software for Technological Forecasting and Modeling Technological Obsolescence. MS Thesis, Mechanical Engineering, Old Dominion University.

Bao, H.P., Mohammad, Y., Obeng, G., 2012, "Cost Considerations in a System of Systems Design and Dealing With System Integration," International Forum of Systems and Mechatronics, Virginia Beach, Virginia, pp 68-76.

The word processor for this dissertation was Texmaker, a form of $\LaTeX$.