SCHOOL OF COMPUTING AND INFORMATION MANAGEMENT

Masters of Science in Information Systems Management

A ROADMAP MODEL TOWARDS UNIFIED SOFTWARE REQUIREMENTS ENGINEERING PROCESS MANAGEMENT: SYSTEM DYNAMICS APPROACH

PRESENTER: GITHINJI DAVID NJOROGE

REG. NO: 14/02104

This dissertation has been submitted is in partial fulfillment for the requirement of master of Science degree, Information Systems Management (ISM) in the faculty of Computing and Information Management of KCA University. Kenya.

OCTOBER 2018

Declaration

I hereby declare that this project report is my own work and in my knowledge have not been previously published or submitted elsewhere to any other institution of higher learning for award of a degree. I also declare that this research study contains no material written or published by other people except where due reference is made and author acknowledged.

1. **Student Name:** GITHINJI DAVID NJOROGE
2. **Reg. No. :** 14/02104

   Signature: …….. Date ……..

This dissertation submitted has been in partial fulfillment for the requirement of masters of Science degree, Information Systems Management (**ISM**) in the faculty of Computing and Information Management of KCA University, Kenya.

I do hereby confirm that I have examined the master's research project of GITHINJI DAVID NJOROGE and I have certified that all revision recommendations addressed are adequate.

Supervisors

3. Dr Henry Mwangi

   Signature ……Date……

4. Rachael Kibuku

   Signature ……Date……

Dedication

I dedicate this project to my family; wife Miriam, children Eunice and Margaret who helped me proofread the research paper and my mother for the prayers and encouragement. Thank you all for making this research project a success.

Acknowledgement

I acknowledge the Mighty Hand of God for His Grace that enabled me complete this course successfully. I am greatly indebted to my supervisors Dr. Henry Mwangi and Rachael Kibuku for their continuous guidance, evaluating, validating my model and supporting me in my research work. I appreciate my family, college mates and the KCA family for the extended motivation and support throughout my studies and research project. Thanks also go to my employer who allowed me undertake this course. May the Almighty God, abundantly reward you always.

I pray that the knowledge and experience acquired in my Master's degree journey helps further my career and impact people's lives.

Githinji David Njoroge
October 2018

Abstract

The Success of software projects heavily and critically depends on the effectiveness of Requirements Engineering (RE) and the Requirements Engineering Process Improvement (REPI). This research study adopts and applies System Thinking/System Dynamics (SD) approach to the complex and dynamic REPI process. The research paper presents a unified model for improving quality software and delivery. Review of the state-the-art practice in RE and REPI literature indicates six categories of problem that motivated the research work reported in this paper. Poor RE and REPI processes make projects to fall behind schedule, encounter budget over-shoots and poor software specification and development. The research study seeks to understand these problems from a feedback control point of view due to lack of quantitative data and agreement on the nature of deficiencies in the current RE and REPI processes. The model developed therefore not seen to be an answer to the existing RE and REPI problems, but as an aid tool for research, researchers and RE stakeholders to advance a deeper understanding needed to answer them. The study identifies several strategies for performing REPI research from empirical to paradigm shift and isolates hot areas of research that address RE and REPI needs for effective software product delivery. Development of the model contributes to research by providing foundation for theory building on RE and RE improvement management of software projects in learning institutions, RE, REPI and software stakeholders.

Key Words: ***Requirements, Requirement Engineering, Requirements Engineering Process Improvement, "Software Crisis", Software Quality, System Dynamics, Systems Development Life Cycle, Quality Assurance***

TABLE OF CONTENTS

Declaration i
Dedication i
Acknowledgement ii
Abstract iii
TABLE OF CONTENTS iv
List of Figures viii
List of Acronyms and Abbreviations xi
List of Table xii
CHAPTER ONE: INTRODUCTION 1
1.2 Background Introduction 1
1.2 Problem Statement 2
1.3. Definition of Key Terms 3
1.4 Purpose of the Research Study 4
1.5 Research Objectives 4
1.6 .Dynamic Hypothesis 4
1.7 Casual Loop Diagrams (CLD) 4
1.7.1 Reinforcing Feedback Loops (R) 5
1.7.2 Balancing Feedback Loops (B) 6
1.7.3 Exponential Growth 6
1.7.4 Oscillation Behavior 6
1.7.5 Explanations for the Feedback Loops 7
1.8 Reference Mode 10
1.9 Motivation of the Research Study 11
1.9.1 REPI Expectations 11
1.9.2 The REPI Vision 11
1.9.3 The Software REPI Initiatives Action Plan 12
1.9.4 The Business Motivations to REPI 12
1.9.5 The Software Project Improvement (REPI) Guiding Principles 12
1.9.6 Planning the REPI Program 13
1.9.7 Monitoring the REPI Program 13
1.9.8 Scope of the Study 13
1.10 Justification of the Research Study 13
CHAPTER TWO: LITERATURE REVIEW 15
2.1 Introduction 15
2.2 Requirement Engineering (RE) 16
2.2.1 Building Blocks of RE 16
2.2.2 Classifications of Software Requirements 16

2.2.3 Requirements Engineering Improvement Process 17
2.2.4 The State-of-the Art REPI Research Process 18
2.2.5 Requirements Engineering Process Improvement (REPI) 19
2.2.6 Requirements Engineering Methods 20
2.2.7 Requirements Checking Process Techniques 21
2.2.8 Requirements Negotiation Techniques 22
2.2.9 Consequences of REPI Failure 22
2.2.10 Managing the RE Improvement Process 24
2.2.11 Requirements Process Analysis (RPA) 24
2.3 Software Crisis 25
2.3.1 REPI and The "Software Quality Crisis" 25
2.3.2 Causes of REPI and "Software Crisis" 26
2.3.3 Early Signs of Software Crisis 26
2.3.4 Software Crisis in Modern Time 28
2.3.5 REPI Alignment Challenges 29
2.4 REPI and Software Policy Analysis 29
2.5 REPI Conceptual Framework 30
2.8 Derived Reference Mode 30
2.9 Summary 31
CHAPTER THREE: RESEARCH METHODOLOGY 32
3.1 Introduction 32
3.2 Research Strategy 32
3.2.1 Research Objectives 32
3.2.1.1 Population 34
3.2.1.2 Interviews 34
3.2.1.3 Questionnaire 34
3.2.2 Research Design, Stock and Flow Diagrams (SFD) 34
3.2.3 Simulation 35
3.3 Research Design 35
3.4 Research Strategy Stages 35
3.4.1 Problem Statement (Stage 1) 35
3.4.2 Field Studies (Stage 2) 35
3.4.3 Model Building (Stage 3) 36
3.4.4 Case Study (Stage 4) 36
3.4.5 Model Simulation Experiments (Stage 5) 36
3.4.6 Policy Analysis (Stage 6) 37
3.5 System Dynamic Development Methodology 37
3.5.1 Pre-Project 37

3.5.2 Project Life-Cycle (PLC) 37
3.5.3 Post-Project 38
CHAPTER FOUR: THE MODEL AND RESULTS OF THE MODEL 39
4.1 Introduction 39
4.2 Model Variables 39
4.3 System Model Boundary 41
4.4 Time Scope 41
4.5 The System Model Structure for the USREPM System 41
4.5.1 User Interface: 42
4.5.2 System Stock and Flow Diagrams (USREPM System/Subsectors) 42
4.5.2.1 Software Project Management Sub-System/Sector 42
4.5.2.2 Human Resource Management System/Sector 42
4.5.2.3 Manpower Allocation Sub-System/Sector 43
4.5.2.4 Development & Productivity Sub-System/Sector 43
4.5.2.5 Quality Assurance & Re-Work Sub-System/Sector 43
4.5.2.6 System Testing Sub-System/Sector 44
4.5.2.7 Controlling Sub-System/Sector 44
4.4 System Model Stock & Flows Diagram Relationships 44
4.5 Causal Loop Diagrams (CLD) 45
4.6 Model Simulation Graphs/Results 49
Re-Work Manpower Effort 50
Error Rework Rate 50
Error Detection and Error Detection Rates 50
Quality Assurance Subsystem & Rework /Sub-Sector 51
Productivity, Rework Rate and Work Force Needed 52
Effects of Productivity, Re-Work Rate and Work Force Absorbed 52
Effects of Experience and Learning on Staff Overall Productivity 52
Effects of Communication on Staff Allocations 53
Manpower Per Average Error against Efficiency 53
Software Bugs Fixing 54
Staff Motivation and Exhaustion 54
Effects of Workforce Exhaustion on Software Development Projects 55
Determination of Project Staff Levels in Software Projects 55
Effects of Turn-Over on Projects 56
CHAPTER FIVE: DISCUSSION OF RESULTS 57
5.1 Introduction 57
5.2 Software Errors and Rework Process 58
5.3 Re-Work Manpower Effort 58

5.4 Software Error Rework Rate 60
5.5 Error Detection and Rework Detection Rate 60
5.6 Error Generation and Error Generation Rate 61
5.7 Software Error Densities 62
5.8 Workforce Productivity, Rework Rate and Workforce Needed 62
5.9 Effects of Staff Experience & Learning on the Overall Productivity 63
5.10 Effects of Communication on Staff Allocations and Production 63
5.11 Software Bug fixing. 64
5.12 Staff Motivation 64
5.13 Effects of Workforce Exhaustion on Software Development 65
5.14 Effects of Staff Experience on Productivity 65
5.15 Determining of Project Staff Levels 65
5.16 Effects of Turn-Over/Quit-Rate & Schedule on Projects 66
5.17 Product Quality Assurance, Continuous Testing and Error Rework 66
5.18 Cost of Errors in Projects 67
5.2 Software Project Sub-systems and Sub-sectors 67
a) Planning Sub-System 67
b) Controlling Sub-System 68
c) Software Production Sub-System 72
d) Manpower Allocation Subsystem/Subsector 73
e) Software Development Productivity Sub-Sector 75
f) The System Testing Sub-system 81
5.3 Field Discussion Groups Research Findings 83
5.3.1 Validity and Reliability Statistics 83
5.4 Conclusion 86
5.5 Models Sub-Sectors and Associated Formulas and Calculations 87
CHAPTER SIX: CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK 93
6 Introduction 93
6.1 Achievements of the Research Study 94
6.2 Limitations of the Research Study 95
6.3 Future Research 95
6.4 Advantages of Using the USREPM Model 97
References: 98
Appendix 1: Research Questionnaire 106
Appendix 3: Research Budget 109
Appendix 2: Implementation Schedule 110
Appendix 4: Implementation Schedule 111

List of Figures

Figure 1.1: Cause- Effect Relationship Diagram 5

Figure: 1.2: Positive and Negative Polarity Link 5

Figure 1.3: Exponential Growth Polarity Link 6

Figure 1.4: Oscillation Behavior 7

Figure 1.5: Dynamic Hypothesis 7

Figure 1.6: Effects of Staff Learning & Experience on Productivity 10

Figure 1.7: Effects of Schedule Pressure on Staff Productivity 11

Figure 2.1: Unified Software RE Process Management Conceptual Model (**USREPM**) 30

Figure 2.2: Derived Reference Mode 31

Figure: 3.1: Research Strategy Design 32

Figure 3.2: Dynamic Synthesis Methodology 37

Figure 4.1: The USREPM System Model Structure 41

Figure 4.2: Software Project Management System/Sector 42

Figure 4.3: Human Resource Management System/Sector 42

Figure 4.4: Manpower Allocation Sector 43

Figure 4.5: Software Development & Productivity Sub-System/Sector 43

Figure 4.6: Quality Assurance & Re-Work System/ Sector 43

Figure 4.7: System Testing System/Sub-Sector 44

Figure 4.8: Controlling Sub-System Sector 44

Figure 4.9: USREPM System SFD/Relationship Diagram 44

Figure 4.10: System Testing Causal Loop Diagram 45

Figure 4.11: Human Resource Management Causal Loop Diagram 45

Figure 4.12: Planning Sub-System Causal Loops 45

Figure 4.13: Controlling Sub-System Causal Loops Diagram 46

Figure 4.14: Software Development Productivity Sub-Sector Causal Loops 46

Figure 4.15: Software Production Sub-Systems Causal Loop Diagram 46

Figure 4.16: Quality Assurance & Re-Work Sub-Sector 47

Figure 4.17: Enhancement of Effort for Re-Work Process. 47

Figure: 4.18: Error Detection and Re-Work Detection Rate 48

Figure 4.19: Effects of Error Generation Rate (CLD) 48

Figure 4.20: Effects of Error Densities on Re-Work, QA and Staff Allocation (CLD) 48

Figure 4.21: Manpower Allocation Sub-System/Sub-Sector (CLD)..... 49

Figure 4.22: Human Resource Management Causal Loops Diagram..... 49

Figure 4.23: Effects of Error Re-work Processes on Software Quality 49

Figure 4.24: Effects of Rework Manpower Effort on Software Quality 50

Figure 4.25: Effects of Error Re-Work Rate on the REPI Process..... 50

Figure 4.26: Effects of Error Detection and Detection Rate on REPI and Software Quality..... 50

Figure 4.27: Effects of Error Detection Rate on Software Quality Assurance 51

Figure 4.28: Effects of Error Generation Rate on Workforce Level 51

Figure 4.29: Effects of Error Densities on Re-Work and QA Staff Allocation 51

Figure 4.30: Productivity, Rework Rate and Workforce Needed..... 52

Figure 4.31: Effects of Productivity, Re-Work Rate and Workforce Absorbed. 52

Figure 4.32: Effects of Experience & Learning on Staff Productivity 52

Figure 4.33: Effects of Communication and Project Briefing on Staff Allocations 53

Figure 4.34: Manpower per Average Error against Efficiency 53

Figure 4.35: Effect of Early Error Detection and Fixing (Goal Seeking) of Software Quality 54

Figure 4.36: Effects of Over-Work and Staff Motivation on staff Productivity (S-Shaped) 51

Figure 4.37: Effects of Workforce Exhaustion on Software Development 55

Figure 4.38: Effects of Staff Experience on Software Process 55

Figure 4.39: Decisions to Determine Project Work-Force Level 55

Figure 4.40: Effects New Hire, Staff Assimilation and Turnover on Workforce Productivity 56

Figure 4.41: Cost of Errors on Software Projects 56

Figure 5.1: Number of Errors as a Major Cause of Software Failure..... 85

Figure 5.2: Effects of Staff Productivity on REPI and Software Product 85

Figure 5.3: Poor Error-Rework Cause Software Failure..... 86

Figure 5.4: Effects of Poor Communication and Schedule Pressure on Software Product Quality..... 86

List of Acronyms and Abbreviations

CLD	Causal Loop Diagram
ESD	Traditional Exploratory Systems Dynamics
EMA	Exploratory Modeling and Analysis
ESSU	European Service Strategy Unit
ESDMA	Exploratory Systems Dynamics Modeling and Analysis
FDG	Field Discussion Groups
KTDA	Kenya Tea Development Agency
I.T	Information Technology
ICT	Information Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
REPI	Requirements Engineering Process Improvement
PLC	Product Life Cycle
PLCM	Product Life Cycle Management
QA	Quality Assurance
SD	Systems Dynamics
SFD	Stock and Flow Diagram
RE	Requirement Engineering
REMP	Requirements Engineering Management Process
R & D	Research and Development
SDLC	System Development Life Cycle
SE	Software Engineering
SI	Systems Information
UK	United Kingdom
USA	United States of America
PM	Portfolio Management

List of Table

Table 2.1: Requirements Types 16-17

Table 2.2: Elicitation Techniques in Practice 20-21

Table 2.3: Requirements Checking Techniques 21-22

Table 2.4: Requirements Negotiation Techniques 22

Table 2.5: Modern Resolution on Time, on Budget with a Satisfactory Results 23

Table 2.6: Resolution of the Software Projects by size 23

Table 2.7: System Analysis Techniques 24-25

Table 2.8: Traditional Project Management Techniques and Tools 26-27

Table 3.1: REPI Model Key Base Variables, Definition and Source..... 36

Table 4.1: The Base Model Endogenous, Exogenous and the Excluded Processes 40

Table 4.2: Endogenous, Exogenous and Excluded Processes in the Enhanced Base-Model 41

Table 5.1: Case Processing Summary 84

Table 5.2: Field Study Item Summary Statistics 84

Table 5.3: Scale Statistics 84

Table 5.4: Validity & Reliability Statistics 84

CHAPTER ONE: INTRODUCTION

1.2 Background Introduction

This chapter outlines the background of the study and discusses characteristics of "software crisis" outlining general and specific objectives and the motivations behind the study. The chapter states and outlines the scope, boundary, justification, benefits and beneficiaries of the study. This chapter debates major causes of low software quality, high production costs, budgetary and schedule overruns, delivery delays, examines early and modern times software crisis as a continuing phenomenon. (Zawedde, A.S.A. et al., 2013), (Kamuni, S.K., 2015) (Putnam-Majarian, T. & D. Putman, 2015) and (Barbara Gladysz, et al. , 2015)

According to Barbara Gladysz, et al. (2015) poor RE improvement process causes poor software quality, projects run over-budget and over-time, making software projects unmanageable and difficult to maintain. Uncoordinated project planning, schedule estimations and change management, low productivity and failed policies historically continue to be major difficulties associated with software quality. (Morrison, B.J., 2012), (Kartik Rai, Lokesh Madan & Kislay Anand, 2014) and (Putnam-Majarian, T. & D. Putman, 2015)

RE improvement involves activities in software development process namely: requirements gathering, analysis, validation of software properties and components delivered to customers that have varied satisfaction based on expected product quality. Traditional REPI approaches are impractical today due to the complexity of system development. REPI is an incremental and interactive process not performed in parallel with other software development activities such as design, implementation, testing as well as requirements documentation. (Parviainen, et al., 2003), (Jalote, 1997), (Pandley & Ramani, 2009), (Mijwaart, 2012), (Annet Reilly, 2011) and (Yaniv & Dov Dori, 2017)

Understanding and aligning REPI reduces risks of unsatisfactory software when stakeholders are involved in building and aligning it to the organization's goals and successful utilization. REPI must focus itself to software quality strategy than on a quality plan. (Hassenzahl, Beu & Burmesster, 2001), (Gorschek & Wohlin, 2006), (Glinz, & Fricker, 2013) and (Annet Reilly, 2016)

1.2 Problem Statement

According to Sterman , (2000), Beecham et al., (2005), Somerville & Ransom, (2005), Mohapatra, S. & Gupta, K., (2011), Annet Reilly, (2014), and Zawedde, et al. (2016), poor RE improvement process (REPI) is a universal problem in software development. The authors continue to indicate that the problem is rooted in failing REPI methods such as Bespoke to capture the dynamics of the process and existing variables interrelationship. The proposed method deals with the steadiness of these problems. According to Gorschek, T., & Davis, A. M., (2007), D.W. ,Williams, (2000), Zawedde, A., & Williams, D. , (2013, 2014) & Philip Morris International, (2015), the problem can best be dealt with by exploring existing methodologies such as the agent based modelling (ABM), system thinking (SD), group dynamics, group model building (GMB), structured equation modeling (SEM), dynamic synthesis methodology (DSM) and analytical modeling (AM). Most existing methods such as Petri Nets, Monte Carlo, Complex theory, decision theory and the Bayesian Belief Network follow a static, probabilistic and mathematical approach considered hierarchical and difficult to model complex systems. They all fail to capture the entire dynamics and greatly address short-term fixes. (Gorschek, T., & Davis, A.M., 2007), (Zawedde, A., & Williams, D., 2013, 2014) and (Philip Morris International 2015)

According to Gorschek T., & Davis, A. M., (2007), (Glinz, & Fricker, (2013) and Zawedde, A., (2016), REPI variations exists due to diversities between variables, lack of timely and accurate information, communication delays and excessive error rework.

1.3. Definition of Key Terms

Requirements: Expressions of needs and constraints placed on a software to solutions of some real-world problem. (Juristo, Moreno, & Silva, 2002) and (Michael, M. et al., 2017)

System Development Life Cycle (SDLC**)**: A description of system development stages from the initial feasibility study to delivered software maintenance. (Kroenke, 2015)

Requirement Engineering (RE): A set of objectives concerned with requirement identification and contexts in which the system runs. (Juristo, Moreno, & Silva, 2002)

Requirement Engineering Process Improvement (REPI**):** A systematic software process aimed at controlling changes in the requirements process, improvements for requirements specifications at reduced costs and product delivery. (Zawedde, A.S.A. et al., 2011)

Domain: Definition of common requirements, terminologies and functionality for a product with a purpose of solving problems in areas of software development. (Bjorner, D. 2006)

Software Engineering (SE): An engineering discipline that spells aspects of software production for a specific customer or market. (Kotonya & Somerville, 2006) and (Glinz, M., & Fricker, S., 2013)

System Dynamics (SD): Modeling and simulation of complex system dynamic behavior over time to generate useful imminent results through deduction, understanding and explaining the behavior of interrelated processes. (Williams, D., 2003a, 2003b), (Harris & William, 2005), (Pruyt, E., 2010) and (Michael, M. et al., 2017).

Quality Assurance (QA): Planned and systematic pattern of action required for the provision of adequate confidence in a product conformity and establishment of technical requirements. The degree to which software satisfies stated and implied needs of the stakeholders and provide value. (Philip Morris International, 2015). (ISO/IEC 25023: 2016) and (ISO 12207: 2017)

Software Crisis: Difficulties of writing useful and efficient programs in a timely manner due to quality, rapid increase in computing power and problems complexity in a dynamic world. (Putnam-Majarian, T. & Putman, D., 2015)

1.4 Purpose of the Research Study

The research study undertakes a holistic and dynamic systems approach to the RE process improvement (REPI) and focuses on the dynamic feedbacks within the software project processes. The approach offers a more rigorous way for describing, exploring and analyzing complex software projects to expose areas of weakness in the RE improvement process.

1.5 Research Objectives

The main research objective is to develop a model towards an RE improvement process (REPI), analyze, and simulate factors influencing the process. This will help organizations, stakeholders and researchers make sound decisions for a complete RE improvement process.

The study seeks to:

(i) Identify dynamics that affect the RE improvement process.

(ii) Define factors that distract the effectiveness of the RE improvement process.

(iii) Design an SD model incorporating key variables that comprise of the RE improvement process.

(iv) Implement a systems dynamic model using the Stella modelling tool.

(v) Test and validate the systems as a tool for analyzing the RE improvement process.

1.6 .Dynamic Hypothesis

Based on thorough research based investigations, the dynamic hypothesis (DH) statement can be proved either true or false. According to Sterman, (2010) the DH is a working theory of how the problem arose as indicated by Olivia, (2003) "It is a theory about how structure and decision policies generate the observed behavior". According to Ranganath & Rodrigues, (2008), the dynamic hypothesis in SD represented may be in a statement, causal loop diagram (CLD) or stocks and flow diagram (SFD). In this research study, the DH based was on CLD. The dynamic hypothesis draws out and tests consequences of feedback loops. The SD model is built on the understanding of the feedback loops and the DH of the RE improvement process shown below. (Sterman, 2010, p.95) and (Ranganath & Rodrigues, 2008, p.7)

1.7 Casual Loop Diagrams (CLD)

According to Sterman, (2000), D.W., William, (2000) and Zawedde, et al., (2016), causal loops show a systematic behavior and exhibit the most suitable way to capture the system processes is to understand

its feedback mechanisms. Causal loops consist of systems process variables linked by arrows that show causal influence among various process variables. Causal loops give a mind-map causal-effect-relationship between system variables. Variables are linked with arrows of two possible polarity states either positive (+) or negative (-) as shown below. (Mohapatra, S. & Gupta, K., 2011)

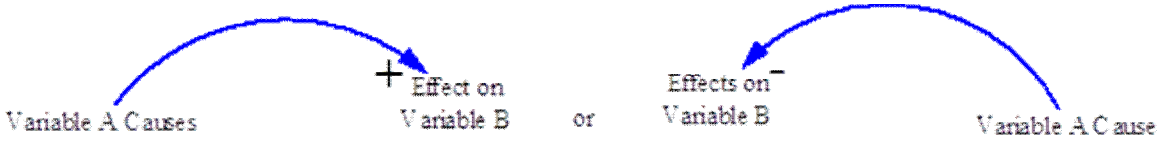

Figure 1.1: Cause and Effect Relationship Diagram

Variable-linking arrows begin with the "cause variable" and head to the "effect variable". When the positive polarity link exists, increase (Decrease) in variable A, respectively causes an increase (Decrease) on variable B. The two variables move in opposite directions resulting to either positive (+) or negative (-) causal link. (Zawedde, A., 2016)

Positive and Negative Causal Links

Figure 1.2 Positive and Negative Polarity Link

In a positive causal link, increase (Decrease) in variable A, leads to an increase (Decrease) on variable B. An increase (Decrease) in variable A causes an increase (Decrease) on variable B. The system responds in a specific manner because of changeable or constant influences on it and represents systematic feedback loops of events on its variables on processes that may lead to cause and effect chain of events. Reinforcing (R) and balancing (B) link feedback loops cause system behavior. (Sterman, 2000), (Sterman, C.D., 2003) and (Zawedde, A.S.A. et al., 2011)

1.7.1 Reinforcing Feedback Loops (R)

According to Pruyt E., (2010), reinforcing loops (R)/ (Positive Feedback) (+) results when a causal element A, results to positive (+) influence on variable B. The implications are; increase (+) of variable A responds to B value with a positive (+) increase. (Richardson, 1986), (Zawedde, A.S.A. et al., 2016) and (Michael Mutingi, et al., 2017)

1.7.2 Balancing Feedback Loops (B)

A balancing loop (B)/(Negative Feedback) (-) influence indicates that a causal variable A, has a negative influence (-) on variable B while at the same time an increase (+) of variable A leads to a decrease (-) of variable B respectively. The system decomposes series of linkages and feedback loops in interlinked frames. Causal loop diagrams enable demonstration of system behavior. (Richardson, 1986), (D.W. William, 2000), (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

1.7.3 Exponential Growth

System exponential growth behavior is because of system's positive and self-reinforcing feedbacks. According to Richardson, (1998), efforts to demonstrate the exponential growth in a system clearly vindicate that a change in one systems variable causes a positive change in another variable within the system. This is clearly equally demonstrated in our system that a positive increase in RE staff productivity may lead to a more stable normal monthly productivity. This clearly demonstrates that an exponential growth in the net RE productivity may be achieved. (Putnam-Majarian, T. & Putman, D., 2015) and (Zawedde, A.S.A. et al. , 2011)

Figure 1.3 Exponential Growth Polarity Link

According to Sterman, (2010), Williams, D., (2003a, 2003b), Zawedde, A. & Williams, D., (2013, 2014) and Zawedde, A., (2016), the goal seeking behavior normally results from a negative self-balancing loop where the system state demonstrates a comparison against the desired system state goals. As a result, a corrective action is undertaken resulting to discrepancies that appear in the system. The corrective action's desire is to try to bring the system back towards the desired state.

1.7.4 Oscillation Behavior

A system's oscillation behavior takes place in event of a delay that happens in the negative feedback mechanism as demonstrated in the figure below. The goal seeking behavior of a system is similar to an oscillating system behavior. However, the later presents a delay in the system process. Demonstrated in the figure below are negative feedback loops seeking to drive the system towards the goal desired. However, instantly the goal never be reached. Therefore, a delay is obvious though the two feedback

mechanisms tend to drive the system in the same direction. In an attempt to move the system towards this goal direction, sometimes this results to system "goal "overshoots. Similarly, the negative loop seeks to bring the system state towards the goal but due to the role delay play in correcting the discrepancies, undershooting to do so occurs. (Sterman, 2010), (Williams D., 2003a, 2003b), (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, A.S.A. et al. , 2011) and (Zawedde, A., 2016)

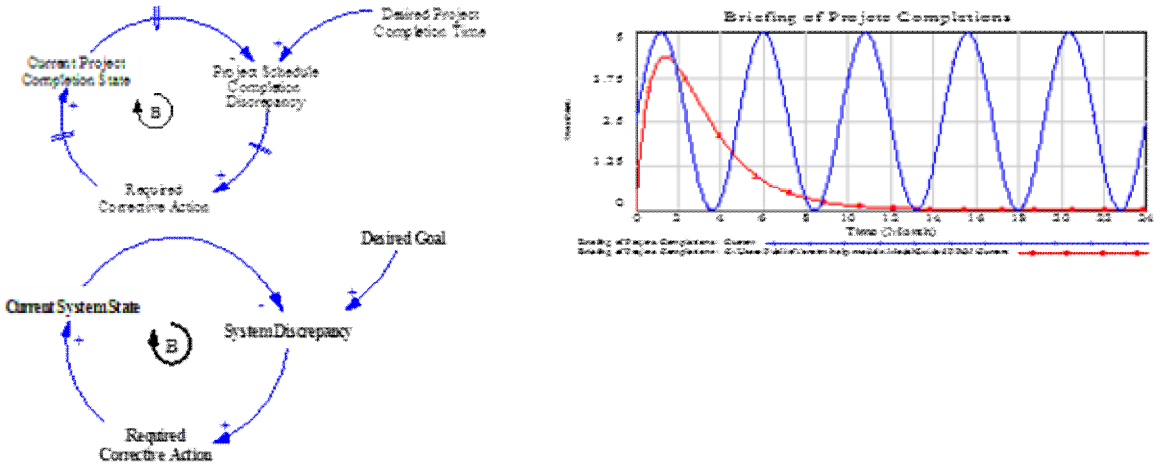

Figure 1.4: Oscillation Behavior

The corrective actions may be achieved by moves aimed at achieving the full potential RE productivity through training reducing the number of unrestrained requirement engineers who undertake the project towards completion time. The demand for new further training compares well with the fraction of work currently completed over the project time-schedule versus the existing RE's productivity at the current time. This results to a need to hire new staff to meet the project schedule. (S.C. Davar, & M., Parti, 2013) and (Zawedde, A., 2016)

1.7.5 Explanations for the Feedback Loops

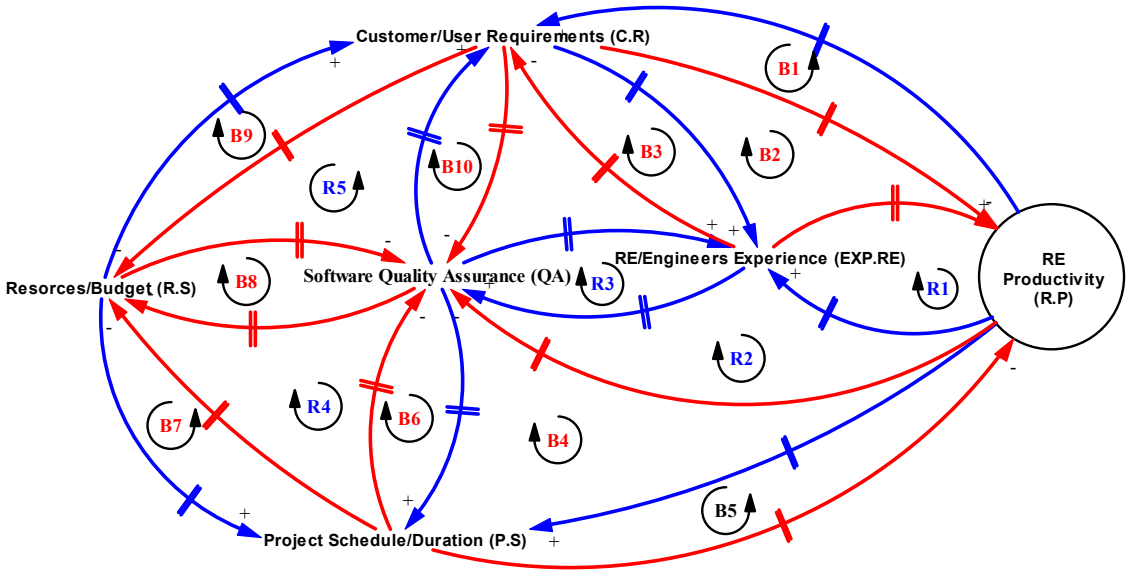

Figure 1.5: Dynamic Hypothesis

The figure above shows the key variables that broadly contribute to software crisis and form the basis of the REPI model. Balancing (B1-B9) and reinforcing loops (R1-R5) influence RE improvement process. Improved model loops, interacting and interrelated variables discussion are in chapter five in models sub-systems and sub-sectors. These extra variables also impede the RE improvement process. (Williams D., 2003a, 2003b) and (Zawedde, A. & Williams, D., 2013, 2014)

Balancing Loops : (B1 to B9)

Dynamic Hypothesis Balancing Loop (B1): Increase in user requirements reduces workforce productivity and increases unresolved requirements over time. (Sabaliuskatie, G., et al., 2010)

Dynamic Hypothesis Balancing Loop (B2): Increase in requirements reduces the workforce productivity, increases project schedule and resource demands causing a budget creep. Failure to further fund RE improvement process increases unresolved need. (Van Oorchot, K. Langerak, F. & Ngupta, K.S., 2011)

Dynamic Hypothesis Balancing Loop (B3): Efforts to resolve requirements lead to improved workforce experience to work and rework on the software, greatly reducing the number of unresolved system errors. (S.C., Davar & M. Patri, 2013)

Dynamic Hypothesis Balancing Loop (B4): Poor workforce productivity leads to increased error rework. When errors increase, over already existing ones, more time is required to correct the errors as well as work on the initial user requirements, which increase project time. More resources are required to meet the gap. When more resources availed are not in time, unresolved requirements increase leading to further decrease in productivity. (Putnam-Majarian, T., & Putman, D., 2015)

Dynamic Hypothesis Balancing Loop (B5): Poor RE productivity leads to increased project time and demoralized staff since achieving the project schedule cannot be, hence further decreasing productivity. (Zawedde, A., 2016)

Dynamic Hypothesis Balancing Loop (B6): Increased project duration leads to reduced software quality and a surge in project duration. (Yaniv Mordecai & Dov Dori, 2017)

Dynamic Hypothesis Balancing Loop (B7): Increasing project duration under a constrained budget invites budget creep and late software delivery. To fit the project within a constrained budget, quality is

compromised leading to increase of unresolved user requirements, leaving firms with no alternative but to seek for more funds/resources to fund the project leading to a budget runoff. (Morisson, B.J., 2012)

Dynamic Hypothesis Balancing Loop (B8): Failure to achieve software quality wastes resources, which renders the software unusable and the project terminated. If the company resolves to continue using the poorly engineered software, users are demoralized and feel unsatisfied. (Philip Morris International, 2015)

Dynamic Hypothesis Balancing Loop (B9): With constrained project resources and budget, software product requirements are unattainable, leading to resource wastage and frustrated customers. (Morisson, B.J., 2012)

Dynamic Hypothesis Balancing Loop (B10): An increase in the number of unresolved errors leads to decreased workforce productivity, elongated project time and delayed software delivery. (Kamuni, S.K., 2015)

Reinforcing Loops: (R1 to R5)

Dynamic Hypothesis Reinforcing Loop (R1): Increase in REs productivity improves software quality, workforce experience and further improves staff performance and efficiency. (S.C., Davar & M. Parti, 2013)

Dynamic Hypothesis Reinforcing Loop (R2): When the perceived workforce productivity achieved is not through improved performance and efficiency, delivered software fails to meet the desired quality standard, leading to increased incomplete requirements, constrained staff, further leading to decrease in the actual staff productivity. (Kabaale, E. Manyoka, K.G., & Mbarika, I., 2014)

Dynamic Hypothesis Reinforcing Loop (R3): Improved staff experience improves actual productivity and efficiency to handle all user requirements. When there is achieved productivity and efficiency, customers are satisfied with REPI and the organization gets their Returns on Investment (ROI).

Experienced staff easily resolve errors to meet the desired product standard. Quality software qualifies contracted firms more contracts, leading to increased demand for software products by other firms. This demand increases the demand for REs and availability of experienced staff that reduces the cost to train and induct newly hired staff. (Gloria, P. et al., 2014) and (Damian, D. & Chisan, J., 2006)

Dynamic Hypothesis Reinforcing Loop (R4): Increased software delivery time leads to loss of resources, constrained budget, extended project duration and poor software quality. (Cuellar M., 2011)

Dynamic Hypothesis Reinforcing Loop (R5): An increase in the number of unresolved software errors and those awaiting rework increases user requirements, demanding resource and budgetary boost. With unreachable budget, the compromised software fails to perform well in production thus customers and stakeholders remain disappointed. The client also fails to perform the desired daily duties, leading to poor work performance. (D.W., Williams, 2000), (Gloria, P., et al., 2014) and (Daneva, M., 2016)

1.8 Reference Mode

According to K. Saced, (1999) and Ranganath & Rodrigues, (2008), p.8.), reference mode (RM) represents system characteristics and behavior rather than its trend. It depicts a far different picture from historical data and qualitative descriptions. Saaed, (2008) argues that RM is a graph pattern that represents the different actual system variable's behavior over time that clarifies the modeler's problem and the client in a pictorial format. Reference mode is a fabric that explains the complex pattern rather than a collection of historical time series. These behaviors form the guiding principles for the system-dynamics modeling process. Reference mode demonstrates system variable's behavior over time graph (BOT), reference behavior or reference conditions. (Khan & McLucas, 2008, p. 24), (Majiwaart, R., 2012) and (Zawedde, A., 2016)

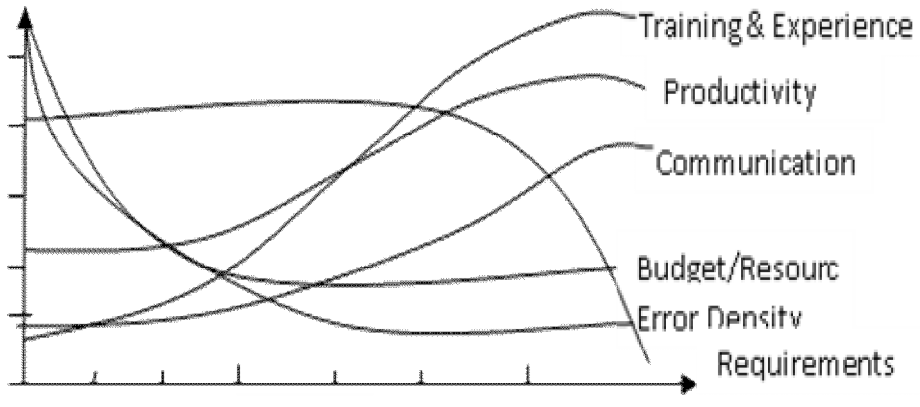

Figure 1.6: Effects of Staff Learning & Experience on Productivity.

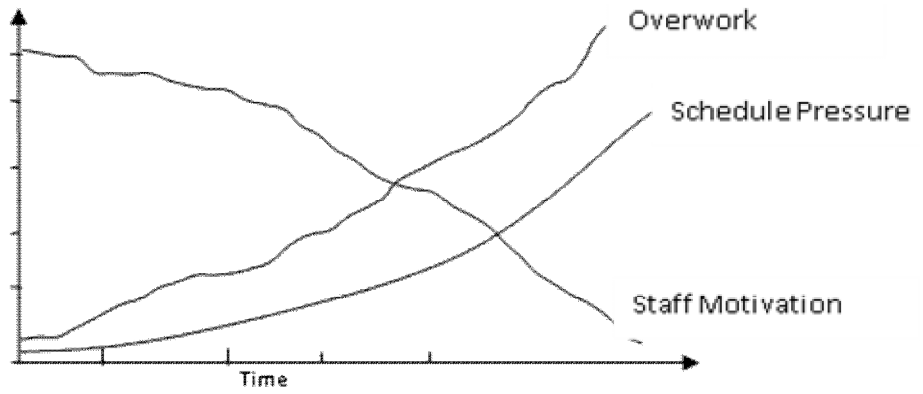

Figure 1.7: Effects of Schedule Pressure on Staff Productivity

1.9 Motivation of the Research Study

Standard bodies such as ISO provide motivation for organizations to initiate the processes of improving a particular process to comply with the global standards. Organizations are compelled to build and comply with the standards. International standards therefore provide a starting point to build the REPI program.

Management and other stakeholders must understand "Why", "What" and "How" to undertake the process improvement initiative to support vision and strategic goals. All cases must determine how well the perceived REPI effort supports the organization's business objectives. Understanding the underlying motivation helps understand the benefits, determine the key performance indicators (KPI's) and probability of success. The performance indicators help gauge the extent of effort (s) required to initiate the process improvement and what it takes to keep the process going. (Philip, A., Laplate, 2017)

1.9.1 REPI Expectations

To achieve a successful RE improvement process, cooperation between all levels of management, practitioners and stakeholders is paramount to understand "What" expected is of them, the cost implications, senior management roles and benefits of committing to the REPI program. (Williams, D., 2003a, 2003b) and (Zawedde, A. & Williams, D., 2013, 2014)

1.9.2 The REPI Vision

If REPI initiatives fail to support the management's vision, their support to the process remains very low and slow and projects may fail to setoff or take long period to implement. (Williams, D., 2003a, 2003b) and (Tricentis, 2018)

1.9.3 The Software REPI Initiatives Action Plan

For effective RE & REPI development, the improvement process initiative, actions and implementation plans must be in place. The action plan requires senior management to present the business "Vision" and goals, clearly stating how the effort supports them. Stakeholders must get an overview of the requirements engineering improvement initiatives in an action plan template. The reviewed major action plan via working group focus areas forms an action plan schedule. (Williams, D., 2003a, 2003b) and (Zawedde, A. & Williams, D. 2013, 2014)

1.9.4 The Business Motivations to REPI

According to D.W., Williams, (2000), listed below are the motivations of business to the Requirements Engineering Improvement Process:

a) Software product quality improvements
b) Cycle time reduction
c) Improved schedule performance
d) Reduced internal rework and wasted rework effort
e) Reduced staff turnover and increased morale
f) Reduced cost of product production
g) Increased customer satisfaction

1.9.5 The Software Project Improvement (REPI) Guiding Principles

According to Williams, D.W, (2000), below are the guiding principles for software requirements engineering improvement process and initiatives:

1 To address business, technical, project management and software quality that has the highest R.O.I value, management must explain to stakeholders why the proposed REPI activities and deliverables are important to them and the business.

2 The software product must be concise, usable and must add value.

3 The REPI initiatives and efforts focus on influencing examples and use of appropriate existing artifacts.

4 Project team, organization, staff and customers (internal and external) must understand the change process.

5 The REPI initiative emphasize on the importance of complying with the domain environment, policies, statutory and international standard procedures.

1.9.6 Planning the REPI Program

The development of strategic REPI activities and action plans entails reviewing the findings and recommendations from field discussion groups (FDG) from base lining activities. The FDG's inputs provide the reference point for REPI's development strategic plan. Findings and recommendations align to organization's vision, mission, strategic plans and business requirements and determines content, priority and sequence of plan activities. (Forester, J.W., 1991), (D.W. Williams, 2000) (Williams, D., 2003a, 2003b), (Zawedde, A., & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

1.9.7 Monitoring the REPI Program

According to Williams, D.W., (2000), the evaluation of activities to monitor the REPI program includes all features of the REPI program and the researcher asks questions such as:

- Right done are things?
- Done are the things rightly?
- Has expected benefits been achieved?
- Is the REPI project on schedule?
- Is the project costs within budget?
- Are you satisfied with the REPI and software product quality?

The REPI program measurement and monitoring evaluates the system's development and REPI's process progress status. The program selected metrics study, evaluates and defines the REPI status and progress. (Williams, D.W, 2000)

1.9.8 Scope of the Study

The study focuses on RE improvement initiative processes from the inception, sending of requirement specifications to customers for approval and the time of finished software product delivery to the client. Exploration and analysis of the existing dynamic relationships between variables and their influence on the REPI process, determine its success through FDGs and establish how best to derive maximum value from it. (Gorschek, T., & Davis, A. M., 2007) and (Mijwaart, 2012)

1.10 Justification of the Research Study

Successful REPI process implementation bridges existing software quality gaps. The SD approach assumes a high-level approach to REPI provides a test and perhaps delivers a clear and more realistic possible ways to resolve software problems. The approach considers diverse subjective factors extensively ignored by the traditional operational models this study considers static, short-range and not

all-inclusive in their assumptions. While traditional, static, probabilistic, mathematical and SD approaches provide project estimation focus, cost and schedule from a REPI's eye, the later tool adopts a more dynamic and strategic view to the REPI initiatives. (William D., 2003a, 2003b), (Zawedde, A.S.A. et al., 2011), (Zawedde, A. & Williams, D., 2013) and (Yaniv Mordecai & Dov Dori, 2017)

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

According to Liuguo, S., Shijing, Z. & Jianbai, H., 2012) establishment of successful RE improvement remains a difficult technical and organizational problem. This chapter addresses the REPI research question by forming a discussion of existing process improvement methods and their relationships to the problem addressed in this study. Strengths and weaknesses of existing RE and REPI approaches contain literature review in context of "software crisis". The conceptual REPI model framework debate in reference to established relationships between literature reviews and predictions of the RE improvement process.

This part of the paper forms a solid theoretical foundation for research, as established in Abdel Hamid's original model, its scientific foundation in SD and establishment of traditional REPI process research publications. Literature reviews offer solid foundations of REPI in system dynamics approach (SDA), dynamics synthesis methodology (DSM) and establishes a strong foundation on the traditional and current school thinking on Total Quality Management (TQM). SD approach understands the complex system behavior over time (BOT) using causal loop, stock and flow diagrams, table functions and time delays. (Abdel-Hamid, 1991)

This chapter establishes the importance of feedback mechanisms in software REPI process. RE forms the initial fundamental building blocks that combine RE improvement processes in system product life cycle and remain the root cause of persistent software errors and failures from integration and system testing. (Michael, M. et al., 2017) and (IEEE, 2017)

When requirements are erroneous, the system ends up released out of schedule, costs more, customers remain dissatisfied and may end up using the faulty software or decide to scrap it altogether. The delivered system ends up being unreliable in use, containing regular systems errors and crush and distracting normal business operations. If the client's choice is to continue using the system, the cost of maintaining and evolving faulty software are high. (Abdel-Hamid & Madnick, 1991), (Zawedde, A.S.A. et al. (2011), (Zawedde, A. & Williams, D., 2013, 2014) and (Kamuni, S.K., 2015)

The research study aims to elucidate, simulate and discuss results of poor RE improvement processes that introduce project cost overruns, schedule pressure and poor software quality standards. (Philip Morris International, 2015)

2.2 Requirement Engineering (RE)

2.2.1 Building Blocks of RE

Software product performance is a descriptive definition. According to Graham, (1991), (Gloria, P. et al., (2014), and (Michael, M. et al., (2017) software performance is categorized in two broad ways; the condition or capability needed by a user to solve a problem or achieve an objective. Secondly, as a condition or capability that achieved or possessed must be by the software or its components to satisfy a contract's standard, specification or other formerly imposed documents.

Requirements not only include the user's needs, but also those arising from general and varied stakeholders, organizational, governmental and industrial standards. Requirements depend on design; explaining "What" the system must do rather than "How" it does it. This, according to Davis, (1999) and Michael, M., et al., (2017) may not be practically possible in real industrial implementations because of varied user and stakeholders viewpoints.

2.2.2 Classifications of Software Requirements

Requirements are classified in various ways. Though difficult, Berry, D. M., Czarneeki, K., Antkiewiez, M., & Abdelrazik, M., (2010) clearly distinguish the different types in practice. For example, though security is classified a non-functional requirement in practice or in the implementation phase, other requirements emerge, which are distinguishably functional e.g. authorizations which fall well under security needs. Functional requirements define what a system does while the non-functional ones define constraints that meet functional needs.

The table below outlines and describes various requirement types. (Chung L.,Yu E., Mylopoulos, J. & Nixon, B., 2000), (Kotonya & Somerville I., 2006) and (IEEE, 2014), pp. 1-138)

Table 2.1: Requirements Types *[Adapted from Chung L.,Yu E., Mylopoulos J., & Nixon B., (2000)*]

Type	Description	Author (s)
Behavior	Systems sequence state, artifacts or class response to event triggers.	-Whittele and Schumann, (2000) - Kniberg & Skarin, (2010)
Formal	Automated tools used to test for	-Kobayashi & Maekawa, (2001)

Property	correctness, stability and totality.	- Bernard, Bidoit et al., (2010)
Functional	System reaction to inputs or specified actions.	- Lam, McDermid & Vickers, (1997) p. 102-113
Interface	Connects a system to the environment, users, and/or other software.	-Maiden, Gizinkis & Robertson, (2004), p. 68-75 - Martin and Meinik, (2008), p.68-75
Process	Actions and operations undertaken and applied to fulfill the desired goal.	-McGrath, (2001)
Quality (Non-functional)	Software features: performance, reliability, security, compatibility stability, portability, robustness, usability and maintainability.	- Melao & Pidd, (2000), p.105-129 - ISO/IEC, (2010)
System Structure	Hardware, software, memory and storage requirements.	
Glossary	Abbreviations, acronyms, synonyms and homonyms.	- Chung, Nixon, et al. (2000), - Pohl and Rupp, (2011)
Scenario	How users interact with the system to achieve the desired goal (s).	- Miilne & Maiden, (2012), p. 83-98 - Alexander and Maiden , (2005)
Stakeholders	Actors or denoted agents: users, groups or organizations that gain or lose something with the software.	- Van Lamsweerde, (2001) - Alexander & Robertson, (2004), p.23-27 - Arlow and Neustadt, (2005)
Structure	Systems entities, attributes and their relationships.	- Arlow and Neustadt (2005) - Glinz, Seybold, et al. (2007)

2.2.3 Requirements Engineering Improvement Process

Many firms significantly consider schedule performance and time to market as key distinguishing factors between market leaders and followers. A contest between schedule commitment and shortening life-cycle time until product release assures the perception as a reliable supplier, as well as the overall profit optimization.

Demands to hasten project handover and product commercialization have improved execution of research and development during the past years. However, today the world continues facing instrumentation challenges of cross-functional coordination that result to prolonged cycle-time and overall project delays.

Software requirements improvement and contracts have in many times been committed without proper alignments, coordination, project management and marketing to boost short-term revenues. These misalignments lead to insufficient capacity planning, poor software development and resource allocation and delayed projects, which lead to ultimately failed REPI.

Successful projects clearly identify user needs and market domain translating them into a product vision executed, following predefined scope and sound management principles. Clearly identified and defined RE improvement process remains the initial and major building block that brings and coordinates different phases of the product life cycle (PLC).

Eveleens, L. & Verhoef, C., (2010) research indicates that only half of the originally allocated software requirements appear in the final release version, consequences of failed REPI. Elbert, C. & Dumke, R., (2012) argued that the effects of the REPI activities require to be defined, developed, implemented and phased-out in a software and its related variants or releases.

Successful RE improvement processes and product management must outline rules governing software development from its inception to market and generate high ROI. Lastly, REPI guides products and desired solutions from inception through refinement, to delivery of the desired ROI to stakeholders and finally the Portfolio Management (PM). (Zawedde, et al., 2016)

2.2.4 The State-of-the Art REPI Research Process

The motive behind RE and REPI activities, is to explore the strengths and weaknesses in software product development. The study samples a survey of the state-of-the-Art RE and REPI research, with assessment of existing models, frameworks and technologies. The section presents a brief collection of reports and surveys to address the effects of the RE improvement process.

According to Kabaale, E., Mayoka, K.G., & Mbarika, I., (2014), the aim of REPI is to introduce engineering principles into practice rather than traditional REPI methods analysis. The RE and REPI must remain a systematic and strictly disciplined process that follow structured repeatable process activities. (Leite, 1987). The ability to identify problems and give suggestions to improve the RE and REPI processes opens a major potential to increase the software project's success rate. Research must capture software gaps through continuous software projects examinations to improve the current RE and REPI processes.

This research study efforts to understand and model current REPI processes for continuous improvement with hope to raise the software projects success rate. According to Madhavji et al., (1994), and Zawedde, A. & Williams, D. (2013) many existing descriptive REPI models in literature provide a clear description of common RE activities and their sequence. However, Nguyen & Swatmann, (2000) and Houdek & Pohl, (2000) say these models are different, bear conflicting nature and range from linear and incremental to cyclical and iterative in structure. REPI models in practice differ from commonly accepted REPI process models in literature. Further, Macaulay, (1996), Katonya, (2006) and Michael Mutingi, et al., (2017) saw existing RE & REPI models as rather situation independent and influenced by customer-supplier relationships, product, industry, technical maturity, multidisciplinary involvement and the organizations culture.

2.2.5 Requirements Engineering Process Improvement (REPI)

Requirement engineering improvement and planning processes control the relationship between process documents and those produced during the RE and REPI processes. According to Stevens R., Brook, P., Jackson, K. & Arnold, S., (1998), RE and REPI processes begin with requirements management, activities identification and ends with change control after requirement development. During the REPI process, activities which continuously nature the RE development process are already over and completed during the product maintenance phase. (Glinz, M., & Fricker, S., 2013)

REPI provides a systematic approach to change control; the focus is to add value to quality requirement specifications at a reduced cost and delivery time within a specified schedule. This implies existing challenges in software alignment form, requirements engineering context and subsequent realignment that can best be located at the RE/ REPI stage (s) of the SDLC. (Solomon, B, Shahibuddin, S., & Ghai, A., 2009) and (Zawedde, A.S.A., et al., 2011)

According to Williams, D., (2003) and Cooper, et al., (2009), RE and REPI activities are dynamic and complex processes that require changes. The REPI must dynamically managed be while preserving relationships existing between variable and identifying existing inconsistencies among RE/REPI activities, devising corrective actions. Therefore the RE and REPI processes remain the main causes of software project failures.

2.2.6 Requirements Engineering Methods

There has been a long tradition of research and practice in RE and REPI. According to Cheng, B, & Atlee, J., (2007), p.10) and Pruyt, E., (2010), early influential research work described the RE and REPI process as an inquiry where requirements engineers' questions about software performance from stakeholders and develops product specifications. Traditional Exploratory Systems Dynamics (ESD), Exploratory Modelling and Analysis (EMA), Exploratory Systems Dynamics Modelling and Analysis (ESDMA) were fast to build, relatively small, simplified, quantitative approach (Quantitative Uncertainty Analysis Approach) and easy to use SD for quick exploration of possible conceivable scenarios

Pruyt, E., (2010) and Lin, & Mathieu, R., (2003) details REs requirements elicitation and expectations from stakeholders, model and analysis of model inputs on the proposed software in consultation with system developers while still seeking stakeholders' implementations acceptance. Chung, L., Yu, E. & Nixon, B. 2000)

Danevas, P., (2012), p.15-16) argued that if the RE improvement process (REPI) is well undertaken on a shared understanding, requirements stabilize and stakeholders are satisfied. Requirement elicitation process enables the REs understand the project vision: REPI and its underlying constraints and context of its deployment. (Conradi, 1998)

The table below outlines elicitation techniques applied in requirement elicitation processes to exposes users/stakeholder's views points, external systems interactions, respective backgrounds, interests and expected outcomes.

Table 2.2: Elicitation Techniques in Practice. *[Adapted from (Danevas, P. (2012), pp.15-16)]*

Technique	Description	Author
Archaeology	Systems analysis to understand their functionality, quality and usage.	-Perez-Casstillo-Gorcifa-Rodriguez de Gunzman, et al. ,(2011) p. 1023-1044 - Davis & Zowghi , (2006), pp. 1-3
Creativity	Create and generate innovative ideas to solve difficult problems.	- Davis & Zowghi , (2006), p. 1-3 - Denger, Berry, et al. (2003
Data Mining	Knowledge gathering method and filtering database requirements as customer needs.	- Dieste, Juristo, et al. (2008), p. 11-13 - Cleland-Huang & Mobasher, (2008)
Interview	Res and stakeholders discussion meeting on	- Alvarez & Urla, (2002), p. 38-52

	system requirements.	- Dwarakanath, A., (2013)
Observation	Study users to understand system usage, processes, strengths and weaknesses.	- El Emam & Madhavji (1985) - Bayer & Holtzblatt (1995) p. 45-52
Introspection	Use of domain knowledge in combination with reflection and empathy to base requirements on experience.	- Easterbrook, Lutz, et al. (1998), p. 1-11 - Velmersch, (2009), pp. 20-57 - Bjorner, D., (2006).
Reuse	Use of existing specifications to avoid reinvention of adequate requirements.	- Lam, McDermid, et al., (1997), p. 102-113
Workshops	Shared meeting between REs & stakeholders to set an agreement between workshop participants.	- Fricker, Glinz, (2010) - Gottesdiener, (2002)
Survey Questionnaire	Paper/electronic questions distributed to stakeholders for their opinions/overview.	- Ng, Barfield, et al. ,(1995), p. 113-127 - Eveleens & Verhoef, (2010), p. 30-36

2.2.7 Requirements Checking Process Techniques

The requirement checking process permits REs to check the RE and REPI approach appropriateness to fulfill the vision, stockholder's software goals and acceptance. The process initiates new query cycles for requirements required, standards or stakeholders that do not meet the set standards considered not good enough. A requirements checklist outlines the stakeholder's agreement with contents, project scope and the SRS document and RE improvements.

The table below outlines some of the selected techniques applied during the requirements and RE improvements process checking.

Table 2.3: Requirements Checking Techniques. *[Adapted from (Danevas, P., 2012) pp.15-16)]*

Technique	Description	Author
Automated Checking	Formal system specification testing to detect differing missing requirements.	-Perez-Castillo, Garcifa-Rodriguez de Guzman & Piattini, (2011), p. 1023-1044
Inspection	Formal review of requirement specifications by stakeholders. Effective at discovering existing problems and understanding the specifications.	-Porter, Votta et al., (1995), pp. 563-575 -Petersen and Wohlin, (2010), pp. 975-996
Peer reviews	Detailed REs feedback on review of quality assurance of the specification work.	
Prototype	Use of models in the roles-play for user and system	

review	acceptance checking.	
Simulations	Model estimates and reviews of the system's behavior using an appropriate tool for correctness check.	-Phaal, Farruk & Probert, (2003), pp. 5-26 - Glinz , Seybold, et al. , (2007)
Walk-through	Detailed and efficient review and discussion of the requirements specification with stakeholders.	

2.2.8 Requirements Negotiation Techniques

The techniques outlined below clearly lead to negotiations, dialogue and finally unclassified agreement between stakeholders and system developers about software. They outline contractual agreement, approved requirements specification as a guide to project management, RE improvement process and software release strategy. (Fricker, 2009)

Table 2.4: *Requirements Negotiation Techniques. [Adapted from, (Fricker, 2009)]*

Technique	Description	Author (s)
Conflict Management	Discovery and conflict resolving process among stakeholders and software development team.	- Pohl and Rupp, (2011) - Chung, Nixon et al. , (2000)
Handshaking	Reviewing and discussing with stated and unstated stakeholders, needs on implementation proposals to align planned software product implementation.	-Fricker, Gorsechek, et al., (2010), p.72-80 -Potts, Takahashi and Anton, (1994), p. 21-32
Negotiation Analysis	Analysis of possible dialogs and outcomes, setting-out fair agreement with a value-creating eye.	-Raiffa, (2007)
Power Analysis	Analysis of the power, influence of stakeholders and their interaction plan.	-Rea and Parker (2005), -Milne and Maiden, (2012), p. 83-98
Prioritizing	Ranking requirements to obtain order on their implementation plan by the project team.	-Achimugu, Selamat et al. ,(2014), -Holm (1979), p. 65-70
Strategy Alignment	Aligning requirements with business strategy through explicit trace-ability	-Alexander and Robertson (2004), p.23-27 - Gorscheck and Wohlin (2006)
Variant Analysis	Analysis and selection of alternative features as a way of solving problems.	-Retting (1994), p. 21-27 -Schobbens, Heymans et al. (2007) ,p. 456-479
Win-win negotiation	Structured tool-supported approach of identifying options for agreement and selection of the most appropriate option.	-Ross (1977), p. 16-34 -Boehm, Grunbacher et al., (2001) p. 46-55

2.2.9 Consequences of REPI Failure

According to Zawedde, A. & Williams, D., (2013) and Zawedde, A.S.A., et al., (2011), research work, software requirement specifications and effective REPI process stand as the base of effective software

functionality and critical determinants of software quality. Stated below, literature reviews show requirement errors are frequent in the software-life-cycle. They stand as the most expensive and time consuming to rectify.

According to Eveleens, L., & Verhoef, C., (2010) and (Hastie, S., (2015), Standish CHAOS report of 2015 shows a study of 50,000 projects around the world ranging from tiny enhancements to massive systems RE implementations. The studies indicate pending work and research challenges to achieve successful software projects. The reported summary shows projects research study outcomes from 2011 to 2015 using the new definition of success factors. (Hastie, S., 2015)

Table 2.5: Modern Resolution (on time, budget with a satisfactory result). *[Adapted from Hastie, S. (2015)]*

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

In the same report, a research study conducted on showed how small projects had higher likelihood success than larger ones. High project failure was also evidently high.

Table 2.6: Resolution of the Software Projects by size. *[Adapted from Hastie, S., (2015)], Standish* CHAOS Report of 2015

	Successful	Challenged	Failed
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
Total	100%	100%	100%

The above tables document a sad reality of "software depression". In most cases, the cost of RE improvements process relates to problems dramatically increasing in software development process. The reports show RE improvement process has a significant impact on the overall success of software projects.

According to Hastie, S. (2015), though highlighted study samples date back many years, the rate probably remains the same. Literature suggests that success depends on "*multiple dimensions*" other than only those in the research study. Joosten, Basten & Mellis, (2011) argue that dimensions in the

research study best represent and give the most appropriate approach towards definition of software projects success.

2.2.10 Managing the RE Improvement Process

The RE and REPI are all life cycle activities related to gathering, documenting and managing needs. The common requirements activities entail elicitation, interpretation and structuring (analysis and documentation), negotiation, verification and validation, change management as well as requirements traceability. (Koboyashi, A. & M. Maekawa, 2001)

2.2.11 Requirements Process Analysis (RPA)

Requirements engineering improvement process (REPI) outlines selected system analysis techniques. Described in the table below, RE process analysis techniques seek to understand and identify requirements in depth. The RPA process distinguishes current systems features, proposed new features and introduces them into a product while taking into account those that are not required. According to Fricker, S., 2008), the process seek to understand how requirements are bound to be implemented in a software, considerations in the development plan, as well their application to system testing. (Glinz, M., 2010) & (Fricker, S., 2012)

Table 2.7: System Analysis Techniques. [Adapted *from Fricker, Rainer & Zwingli, (2015*)]

Technique	Description	Author
Domain Driven Development	Specification of relevant system concepts in context, to be implemented - those that must be implemented and respected by the system.	- Glinz, (2010) - Denevas and Garva , (2012) -Bjorner, D., (2006).
Formal Specifications	Mathematical and formal logical expressions that enable automated completeness, consistency and correctness checking.	- Holtmann, Meyer et al, (2011) - Glinz and Fricker , (2013)
Informal Modeling	Sketching a system model to reflect and discuss how system variables interrelate.	- Glinz , (2010) - Glinz & Fricker, (2014)
SD Modeling	SD modeling and tools that demonstrate system variables interrelationship and system BOT.	- Glinz, Seybold & Meier, (2007) - Gorsecheck, Fricker & Palm, (2010)
OOA Method	Use object-oriented language (UML) to specify the structure, functionality and system behavior.	- Arlow and Neustadt, (2005) - Glinz , Seybold et al., (2007)
Prototyping	A tool or paper based estimate of an end-system	- Rettig (1994), p. 21-27

	to achieve planned system tangibility and validity.	- Gorschek, Fricker, et al. ,(2010)
Quality Checks	Detailed system analysis establishing its goals, functionalism and requirements.	- Chung, Nixon et al., (2000) -Gorschek & Wohlin, (2006), pp.79-101
System Analysis(SA)	Specify systems structure, functionality and behavior using structured analysis language.	- Ross (1977), p. 16-34 -Hassenzahl, Beu et al. ,(2001), pp.70-76 - Gottesdiener, (2002)

According to Friker, S., & Glinz, M., (2010) and Fricker, S., (2012) organizations establish tasks, procedures, associations, methods, determine business object development and methodologies for conducting business needs analysis to identify business ideas. To fulfill the process, identification of stakeholders for requirement analysis is key. The second approach towards identified business requirements analysis is to focus on capturing new software requirements and RE improvement process. The third step is to classify requirements into four main groupings namely; functional, technical, operational and transitional requirements for easier analysis and system design. The fourth step is attempt to interpret and record new requirements for RE improvement process. The process defines requirements, eliminate ambiguous & vague definitions, and prioritize them based on limited project schedule and budgets. (Yaniv Mordecai and Dov Dori, 2017) & (Kabaale, E. Mayoka, K.G. and Mbarika, I., 2014)

Requirements analysis measures RE and REPI influence on software project, existing product processes and human capacity. Since requirements, conflicts are inevitable, established conflict resolution mechanisms resolve them through stakeholder's scenario analysis. The final major process determines how reliable and easy to use software will be through a detailed system analysis. This leads to a detailed RE and REPI and software analysis cycle. (Yaniv Mordecai & Dov Dori, 2017)

2.3 Software Crisis

2.3.1 REPI and The "Software Quality Crisis"

According to Jones and Bonsignour, (2012), though software has widely been used in human history, it stands to have the highest failure rates of any product in the same historical time due to poor software quality delivery. Society heavily relies on software products to operate. Software failures form major bases of serious consequences that go way beyond the cost-factor problems. (Eveleens, L., & Verhoef, C., 2010), (Hastie, S., 2015), (Philip Morris International, 2015) and (Tricentis, 2018)

2.3.2 Causes of REPI and "Software Crisis"

Firesmith, (2003) argued that RE and REPI fail because REs use technical words during system design that end-users do not understand, making systems irrelevant and unusable due to inadequately trained REs who deal with the stakeholders. In most scenarios, REs assume certain common ways of requirements and REPI implementation. This approach limits REs & REPI requirements installers and consequently the user thus failing to resolve the initial problem(s). (Firesmith, 2007), (Zawedde, A.S.A. et al., 2011), (S.C. Davar and M., Parti, 2013) & (Yaniv Mordecai and Dov Dori, 2017)

2.3.3 Early Signs of Software Crisis

Software product encompasses both internal factors, (probably controlled) and external factors that are difficult to adjust and control. The outlined problems range from, qualitative, quantitative or a mixture of the two. Successful RE and REPI processes are complex in nature and attribute to the rate of software expansion.

To reverse the REPI and software project failure rates and resolve "software crisis", several techniques such as object methodology (O.O) and System Dynamics Modelling (S.D.M) have been developed through research. Early RE and REPI methodologies (e.g. CMM, BOOTSTRAP, Trillium, SPICE and ISO 9000) focused on definition of identified project structure, detailed schedule, budget monitoring and controlling structures. (William, D. & Van Dyke, 2007), (D., Williams, 2000), (Pruyt, E. 2010), (Mwangi, H. et al., (2015).

In the table below, selected are traditional RE and REPI techniques used in software project management and their role. Most techniques assumed strict linear analysis based on control as an ideal system methodology.

Table 2.8: Traditional Project Management Techniques and Tools. *[Adapted from* Williams D. and Van Dyke, (2007)]

Technique/Tool	Intended purpose
Work Structure Breakdown	Definition of expected project work, schedule and cost estimations.
Role matrixes	Role assignments
Cost Schedule	Identification of project capital requirements for budget estimations

Project work techniques	Scheduling work for network technique to determine the impact and risks analysis, cost estimations, resource allocation and management analysis.
Use of charts (e.g. bar graph)	A Simple representation of project schedule without showing procedures and relationships between activities.
Project Control	Performance indexes generation to determine project over-runs and the required corrective actions and incorporating graphs in the technique.

A. Nasirikaljahi, (2012) stated that as early as 1979, the term "*Software Crisis*" coined in a public debate as recorded in a congressional report issued by the controller general, cited the scale of problem in the federal government and gave a summarized issue named: *"Controlling of Computer Software Development. Serious Problem Requires Management Attention to Avoid Wasting Additional Millions."* The report indicated that the US government got less than 2% of the total value for its investment (Abdel-Hamid, 1991: p.3), (Zawedde, A.S.A. at al., 2011) & (Kamuni, S.K., 2015), (Lech, P., 2013)

In the late 70's and 80's, reports on "software crisis" similarly revealed comparable trends and contained significant RE improvement errors that lead to additional development or rework, pushing costs and delays higher, leading to software slapping soon after its delivery. Examples of similar scenarios reported in the congressional reports of the 1979 and several other published papers. (Thayer, 1986), (Sclender, 1989), (Frank, 1983) and (Zawedde A.S.A. et al. 2011).

According to Kabaale E., Manyoka, K. G. & Mbarika, I. (2014), the initial REPI responses to software problems focused largely on cost overruns and delays. The focus directed towards software quality, though the main attentions were on cost and management. Early and young software industry suffered lack of tools for REPI analysis and management. Several published pioneer REPI studies focused on the questions addressing software production. (Abedel-Hamid, 1991)

In the early 1990's, the scholarly doctoral thesis work of Abedel-Hamid, (1991), contains early REPI attempts to build a complete SD model and identify contributions of RE and REPI in fueling software failure rate. Later, Adbed's base model used was in the concept development in the field of software. (Abdel-Hamid, 1991: p. 3-5)

From Abdel-Hamid's research findings, there emerge two distinct views on the emergence of software problems namely:

1. **Managerial:** Lack of routines and discipline ultimately results to poor management decisions within the software industry. The distinct root cause of the software crisis demonstrates lack of management efforts in the various stages of software development causing poor planning and cost estimations as well as poor staffing strategies. This deficiency further speeds up the software crisis when development demands more attention than perceived. (Abdel-Hamid, , 1991), pp. 3-5 , and (Jones C. and Bonsignour, O. , 2012)
2. **Technical Concerns:** This viewpoint of the "software crisis" concern itself with the technical software development handles that cause software delays and cost/budget overruns. Early and the young software technical tools of trade developed during production results to delays, trials errors, and studies that considered attempts to handle software development technical issues. However, with these early efforts and readily available tools, "software crisis" continues to influence the software industry. (Zawedde, A.S.A. et al., 2011), (Bjarnason, Wnuk, & Regnell, 2011) and (Kartik Rai, Lokesh Madan & Kislay Anand, 2014)

2.3.4 Software Crisis in Modern Time

The Software industry have revolutionized over time from the emergence of the "software crisis" of the late 90's and early 20's and completely changed the face of the software industry. Today, software development industry is a gigantic business present in any part of the world. With notable transformations in the industry's scope, width and depth, the big question of the software problems remain a big question, a replica of late 70's, 80's, early 20's and in the current software development industry. The big question in the "software crisis" as defined by Robert Charette's article in the IEEE (2014) reveals much-related situations characteristic to those reported in the congressional report. (Kartik Rai, Lokesh Madan & Kislay Anand, 2014) and (Yaniv Mordecai & Dov Dori, 2017)

According to Charette, (2010), software projects continue to suffer budget overruns, delivery delays and possible terminations with inconceivable losses across the globe. For example, in 2004 by according to Charette, (2010), "the USA spent $60 billion in software contracts and with a modest failure-rate at 5% that translated to an estimated loss of $3 billion". Charette, (2010), speculated that the true failure-rate could have been much higher as 15-20% of all contracts terminated or abandoned shortly before, or after delivery. The author believed "over abandoned software cost the economy between $25 and $75 billion" (Zawedde A.S.A., et al., 2011) and (Hastie S., 2015)

The second example, Charette, (2010) the Hudson Bay Corporation, experienced a serious problem with its inventory system called "Big Ticket" aimed to revolutionize its IT infrastructure. The company's poor software quality contributed to a huge loss of $33.3 million when the company failed to operate the huge and complex system it targeted to operate and manage. In Hastie. S., (2015) research paper, (Standish CHAOS report of 2015) in the last two decades there was insignificant change in the reversal of the observed software failure rate. This implies existing gaps in the RE and REPI efforts, and methods for software projects improvement and alignment. (Eveleens L. & Verhoef, C. (2010)

2.3.5 REPI Alignment Challenges

The REPI process aims to ensure meeting customer's expectations. However, to achieve RE and REPI verification and validation, alignments of software production activities of must ensure it meets the organization requirements. argues that the RE and REPI processes are poorly coordinated with development and testing tasks leading to the delay of software functionality when there is a large scale software development, RE and REPI challenges increase the cost of error rework and lowers software quality. (Kraut & Streeter,, 1995), (Damian & Chisan, (2006), (Gorschek & Davis, (2007), (Sabaliauskaite, et al., 2010), (Eveleens, L., & Verhoef, C., 2010) and (Hastie, S., 2015).

According to Nurmuliani, Zowghi and Fowell, (2004) the main challenges in the RE & REPI processes is failure to adjust requirements specifications during the development phase, making it difficult for users to create new uses. However, manual changes apply only for smaller systems and hence posing a manger challenge for the bigger systems. Berry, Dahistedt, Natt, Regnell & Persson, (2007), Czarneeki, Antkiewiez & Abdelrazik, (2010) argue that for successful RE and REPI's cross communication, is important to improve the two processes. According to the authors, most software RE and REPI challenges facing organizations are not technical.

In market drive large-scale software production, communication between RE's and the end-users is weak. The main cause of large-scale software production failures include the customer decision-making structure, engineer's temporal aspects, lack of common views and finally the scale. (Dahisted A.G. Natt, O. D. Regnell, B. & Persson, A., 2007) & (Bjarnason, Wnuk, & Regnell, 2011)

2.4 REPI and Software Policy Analysis

According to Michael Mutingi, et al., (2017) and Yaniv Mordecai & Dov Dori, (2017), RE and REPI policy analysis reveals existing policy gaps. Policy analysis, the first step to policy and institutional

change rules and practices heavily determined relevant policies. Policy analysis examines a number of questions in the REPI process to resolve software problems namely:

1. The issue at hand to tackle in software developments
2. Why is the issue important to the software crisis?
3. What are the relevant policy areas to the problem at hand?
4. Who are the main stakeholders?
5. What are the existing research issues on the software crisis?
6. Recommendations to address the complaint issue
7. What will the alternative policy recommendations produce?
8. How well to address possible constraints, resistance to overcome and recommendations translated into practice.

2.5 REPI Conceptual Framework

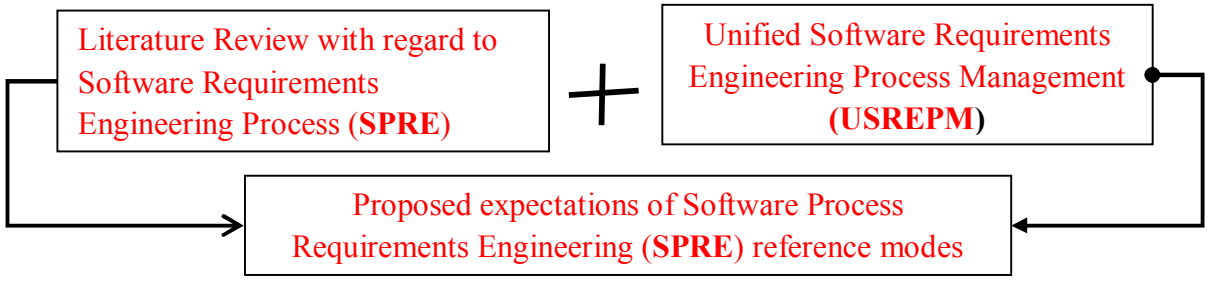

Figure 2.1: Unified Software RE Process Management Conceptual Model (**USREPM**)

The figure above represents proposed expectations, a conceptual model (Damian, D. & Chisan, J., (2006) of software process requirements Engineering (**SPRE**) Reference modes.

2.8 Derived Reference Mode

This is an abstract concept, in fact, it is not historical data and is arrived through careful analysis of historical data and a future inferred from it. Reference mode is a pattern of behavior conceptualized from historical time series data discussed and shown in the figure below:

An increase in new RE and REPI staff leads to an increase in the cumulative wage costs that increase pressure to complete the project in time. This will clearly show an exponential growth in the pressure to complete the project with an attempt to bring down the monthly RE staff wage cost by reducing the need to hire more new RE staff and training staff to reduce inexperience. Hiring more trained and experienced REs, brings down the levels of untrained workforce. The attempts to increase RE's staff productivity

takes a goal seeking approach because is a negative self-balancing loop. (Sterman, 2010) and (Hekimoglu, M. & Barlas, Y., 2010)

Increased monthly costs due the introduction of new staff, leads to an overall increase in the total project cost. However as the new staff get inducted into the system, this may eventually lead to an increase in the RE's productivity, improved and more stable work-flows even in the event of an increase in the project completion time. (Zawedde, A.S.A. et al., 2011), (Glinz, M. & Fricker, S., 2013), (Barbara Gladysz, et al., 2015) and (Annet Reilly, 2017)

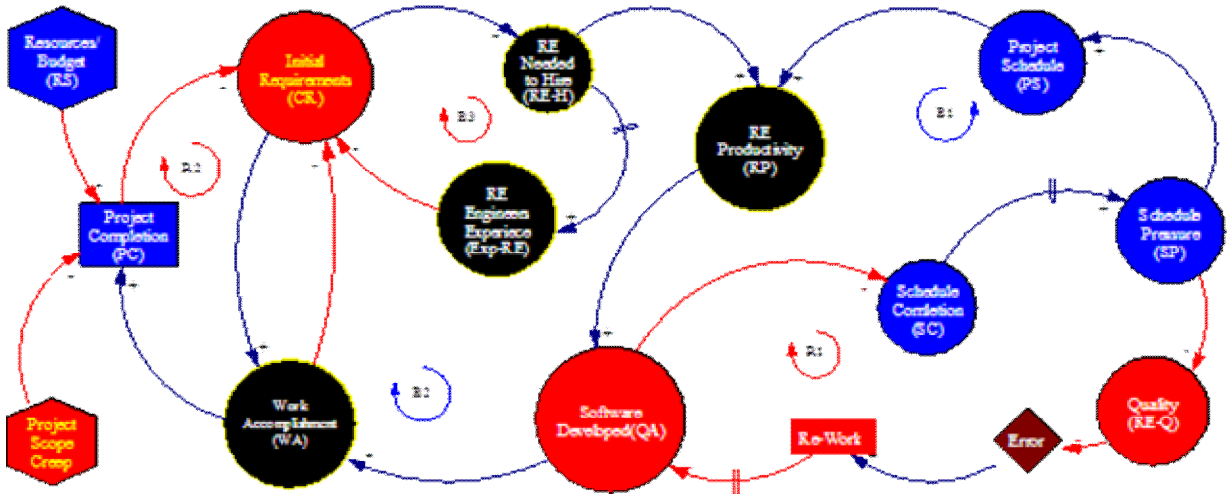

Figure 2.2: Derived Reference Mode

2.9 Summary

This chapter provided the review of literature guided by the research objectives. The literature assisted the researcher to come up with an appropriate research methodology in the next chapter. The reference mode forms the base for the research methodology, objectives, strategy and development and expansion of the REPI base model discussed in the next chapter.

CHAPTER THREE: RESEARCH METHODOLOGY

3.1 Introduction

Chapter 3 outlines the research strategy, design and the objectives of each strategy. The chapter discuss the systems dynamics modelling approach as indicated in the concepts and tools used for SD. The aim is to enlighten the reader on SDM and its application. SD approach methods, establishes the core concepts of systems dynamic modeling and tools used. The chapter outlines core concept of causal loop diagrams, the causal relationships between variables and feedback loops in the system. Selected visual aids, in the form of diagrams describe stock and flow diagrams and discuss the dynamic synthesis methodology. The chapter details the REPI model, stock flows diagrams. Causal loops defines delay, non-linearity and system BOT. (Gorschek, T., & Davis, A. M. 2007), (Michael Mutingi et al., 2017) and (Mwangi, H., Williams, D., Timothy, W., and Zipporah, N., 2015)

The research methodology outlines research study strategy that outlines the way in which research is to be undertaken and, among other things, identifies the methods used. These methods, discus how means or modes of data collection and specific result are calculated arrived at and. (Liuguo, S., Shijing, Z. & Jianbai, H., 2012) & (Michael Mutimgi, et al. (2017)

3.2 Research Strategy

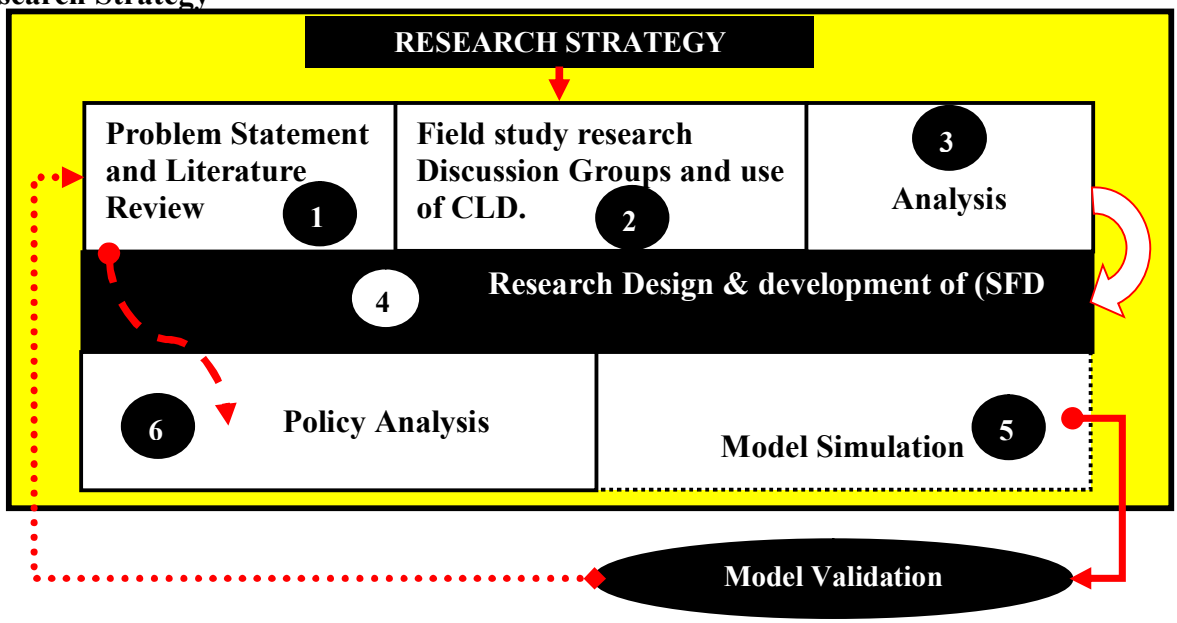

Figure: 3.1 Research Strategy Design

3.2.1 Research Objectives

The objective and problem are identified, problem statement formulated. Consulted literature review identify and reference earlier research work carried out. Conducted with help of field discussion focus groups (FDGs), study were about the existing problem. The focus field groups discursions considered were in the research because of the need to interact with the researchers, expert groups as well as aid in

gathering qualitative data. Qualitative data utilized by the dynamics synthesis methodology is key in development of causal loop diagrams (CLD). (Tveito, A. & Hasvold, P., 2002), (Zawedde, A. & Williams, D., 2013, 2014) and (Pandey, D. & Ramani, A.K., 2010)

Case Study for Research

The qualitative research adopted design and selection of a case study approach give the nature of a dynamic synthesis approach (DSM). The researcher selected one case study company was to participate in the study because it represented different application domains, experience in software development and company sizes. Based on this criteria, the researchers findings gave a general overview of the company, their attitudes towards RE and REPI and challenges they faced. The selected company operated in a wide range of application domains, business information systems, web based systems, office automations, ERP's among many others. (Tveito A. & Hasvold, P., 2002)

The target sample selected for this research study operated in a wide range of roles i.e. system users, system administrators, programmers, project leads/experts, project managers, quality assurance and system testers, services manager and general manager IT.

The sampling techniques methods used were the probability ones and included:

- Simple random sampling method considered was good for data collection since all the system users were available.
- Stratified sampling method which was suitable due to my targeted population of system users
- Systematic sampling method suitable since the managers or head of ICT services represented the users

Data Collection, Study Area, and Data Analysis, validation, and Sampling Techniques

Quantitative data was coded, collated and themed before being analyzed using IBM Statistical Package for Social Sciences (SPSS). Descriptive statistics, including frequencies, percentages and means, standard deviations, variance applied to extract the challenges faced in RE and REPI processes.

Using purposive sampling technique, 100 company employees to participated as respondents. This purposive sampling used primarily targeted only personnel useful to the study given their knowledge and skills, work experience and roles in the organization. Field data collected is grouped and analyzed using thematic approach method to capture general and technical knowledge relating to software development problems.

3.2.1.1 Population

The research population study had a total range of between 5000-6000 people focusing on company branches (69 in total). Each company branch had at least 20 system users where eight were section managers. From the total population, the target groups included; system developers, section heads, database administrators, programmers, project managers, system database testers, system analysts and auditors. A total target population of 150 people was in focus. However, the accessible population was 100 people. Interviews and questionnaire (Structured and non-structured) method was used in data collection. The research considered eighty, (80) forms received from respondents which were statistically analyzed using IBM SPSS after eliminating those not fully filled to eradicate inconsistencies. (Poloudi, A., 2004), (Zawedde, et al., 2011), (Majiwaart, R., 2012) and (Pruyt, E., 2010, 2013)

3.2.1.2 Interviews

A face-to-face or direct communication for data collection where the interviewee/researcher holds a brief discussion with the interviewer, and have either an open or a closed interview. The method considered was most appropriate since the researcher was able to know about the existing system through a discussion with the system users and help save on cost. (Pohl, K. & Rupp, 2011) and (Yaniv Mordecai & Dov Dori, 2017)

3.2.1.3 Questionnaire

For FDGs data collection, the researcher prepared a set of questions and gave them to the 100 people to answer. The method was most appropriate since the researcher was able to reach a larger number of people and it was economically fit. (Williams, D., 2003a, 2003b), (Zawedde A.S.A. et al., 2011), (Krishnaveni, R. & Deeper Ranganath, 2011), (Kabaale, E, Manyoka, K.G. & Mbarika, I., 2014) and (Zawedde, A., 2016)

3.2.2 Research Design, Stock and Flow Diagrams (SFD)

After causal loops were developed, the researcher designed and synthesized the causal loop developed using Vensim software. Stock and flow diagrams (SFD) developed using Stella Software. To create stocks, flow, action converters, decision processes as well as graphs. For modelling and simulation, the Stella software tool used was to develop stock and flow diagrams. (Pruyt, E., 2010) and (Michael Mutingi, et al., 2017)

3.2.3 Simulation

After the stock and flow diagrams developed under different systems and sub-systems/sectors, simulations run were to capture systems behavior over time (BOT). The model's simulation results displayed were in form of graphs developed through a powerful graph tool within the software. Simulation graphs disclosed identified existing problems that applied in study of existing policies to reverse the occurrence of the problem in future. (Hekimoglu, M., Barlas, Y., 2010), (Sterman, 2000), (Sterman, C.D., 2003), (Sterman, J.D., Oliva, R. Linderman, K. & Bendoly, E., 2015), (Michael Mutingi, et al., 2017) and (Zawedde, A.S.A., et al., 2016)

3.3 Research Design

Research design began with identifying the key variables with aim to define the key research objectives of the study. Based on the objectives, a schedule is prepared to guide each focus discussion group. (D.W., Williams, 2000). Outlined in Dynamics System Modelling (DSM), SD stood superior in comparison to all other modelling approaches since it allowed case study development with simulation to give a deeper problem investigation. Incorporating SD simulations into the study allowed application of DSM methodology to strengthen case study. This combination enabled collection of data and current on-site systems products from its natural setting, system owners, user requirements as well as specifications needed to develop a unified and generic model. (Sterman, 2000), (Sterman, C.D., 2003), (Sterman, J.D. Oliva R. Linderman, K. & Bendoly, E., 2015)

3.4 Research Strategy Stages

Outlined below are six stages of the research study strategy namely:

3.4.1 Problem Statement (Stage 1)

A thorough digging into literature reviews helped define the key factors that historically influenced the success of software projects. Interviews conducted were through focus group discussions with personnel that handled system development, project planning, human resource and other resource planning, project management, quality control, system testing and users. The results obtained were from interviews to enrich the descriptive model representing behavior over time (BOT). (Pruyt, E., 2010, 2013) and (Zawedde, A.S.A., et al., 2016).

3.4.2 Field Studies (Stage 2)

Field discussion groups (FGD) results determine the existence of challenges facing software development projects. The interview's focus was to identify existence of key variables that greatly contribute to poor RE/REPI and overall software project failure.

3.4.3 Model Building (Stage 3)

Gathered information from stage two (Field studies) was used to develop the descriptive model in form of Causal Loop Diagrams (CLD) which were presented to stakeholders for enhancements. This formed part of qualitative research. (Michael, M.J. & Shipman, F.M., 2000), (Kotonya, G. & Summervile, I. (1998, 2006), (Krishnaveni, R. and Deepa Ranganath, 2011) and (Kamuni, S.K., 2015)

3.4.4 Case Study (Stage 4)

Empirical investigations conducted were using data collected from the case study to populate the model. (J., Starman, 2000), (Zawedde, A.S.A. et al., 2011), (Zawedde, A., 2016) and (Yaniv Mordecai and Dov Dori, 2017)

Table 3.1 REPI Model Key Variables, Definition and Source. (*Model* e*xpanded further later)*

No	REPI Key Variables	Definition	Source
1	Customer Requirements	Functional and non-functional needs.	-Field Discussion Groups
2	Software Quality Assurance	-Expected software standards as defined by user/Stakeholders needs.	-Field Discussion Groups
3	REs/Developers Productivity	-Effectiveness and accuracy of REs, designers, developers, quality assurance team and testers.	-Model
4	Resource/Budget	- Available project resources (allocated budget), time, hardware, software and people.	-Model - Field Discussion Groups
5	REs Experience	- Workforce knowledge level on subject area, e.g. design, system testing and development.	-Field Discussion Groups - Model
6	Project Schedule/Duration	-Agreed/proposed/projected project completion time and duration before hand-over of software to customer as per the contract document.	-Field Discussion Groups - Model

To measure product quality, productivity, errors and error detection, rework rates, staff communication levels, training conducted, staff levels, motivation, staff quit rate and replacements are included in the model. This formed part of the data used in quantitative research. (Putnam-Majarian, T. & Putman, D., 2015)

3.4.5 Model Simulation Experiments (Stage 5)

At this stage, scenario building, model testing, validation and simulation performed using the Stella software interface. Inputs adjusted are to give the model's diverse behavior output over time. (Zawedde, A.S.A. et al., 2011), (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

3.4.6 Policy Analysis (Stage 6)

Intervention measured, identified strategies and proposed policy intervention in the research design stage towards REPI assisted the researcher carry out informed policy analysis by evaluating obtained results, comparing them to literature reviews to give informed critique policies on the REPI process. (Zawedde, A. & Williams, D., 2013, 2014), and (Zawedde, A., 2016)

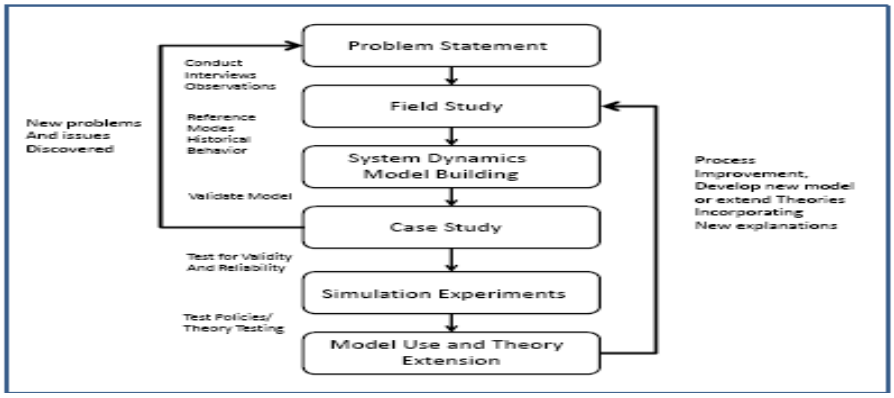

Figure 3.2: Dynamic Synthesis Methodology [Adapted from D.W., William, 2000]

The above figure represents the dynamic synthesis model towards requirements improvement process. (D.W., Williams, 2000)

3.5 System Dynamic Development Methodology

The RE improvement process guides software development to meet desired requirements on time and budget. The RE improvement process aligned is in three phases that include pre-project, project life cycle and post-project. (Hekimoglu, M. & Barlas, Y., 2010), (Gloria, P. et al., 2014)

3.5.1 Pre-Project

The first phase of SDM involves identifying the project and searching for project sponsors. Project managers ensure the project team is highly disciplined. (IEEE, 2012-2017), (D.W., Williams, 2000), (Gloria, P., 2014), (Majiwaart, R., 2012), (Pruyt, E., 2013, 2014), (Daneva, M., 2016) and (Yaniv Mordecai & Dov Dori, 2017)

3.5.2 Project Life-Cycle (PLC)

The second stage of SDM encompasses a number of activities namely:

- Performing the feasibility study
- Performing the business study
- Function model iteration
- Design and build iteration
- Implementation

3.5.3 Post-Project

The last stage of the DSM that involves system testing and maintenance. The stage checks the system functionality and achievement of software requirements. (D.W., Williams, 2000), (Cuellar, M., 2011), (Friker, S. & Glinz, M., 2010) (Wolstenholme, E.F., 2004), (Lang, M. & Duggan, J., 2012), (Pruyt, E., 2010, 2013), (Lech P., 2013) and (Michael Mutingi, et al., 2017)

Conclusion

Resources/budget and allocation, staff capacities (training & experience), workforce productivity, requirements, schedule and software quality are major variables that directly affect the RE and RE improvement process and quality software delivery. Improved loops for the model capture other variables that affect the RE, REPI and software quality that are included in the model expansions. (Zawedde, A.S.A. et al., 2011), (Mohapatra S. & Gupta, K., 2011), (Zawedde, A. & Williams, D., 2013, 2014), (Putnam-Majarian T. & Putman, D., 2015) and (Zawedde, A., 2016)

CHAPTER FOUR: THE MODEL AND RESULTS OF THE MODEL

4.1 Introduction

This chapter discusses the models structure and design. Field data results presented and conceptualized by using of the model. The section expounds on the second research strategy discussed in chapter 3 and forms a strong foundation of the third research strategy (Data Analysis). The model causal loops and simulation results shown are in this chapter.

Using Vensim software, key causal loops diagrams (CLD) are assembled. Using the Stella software further reconstructs the causal loop diagrams to include stocks and flow, converters, action connectors, sector frames and graphs to graphically present results and their associated CLDs. The results shown in this chapter are without discussion but discussed are in detail in the next chapter.

The dynamic relationship between sub-system and sub-sector variables in the conceptual model is that the original Abdel-Hamid model Abdel-Hamid, (1991) failed to establish and simulate. This chapter contains causal loop diagrams for the various sub-sectors/sub-systems and their dynamic relationships between them. The causal loop diagrams represent various interlinked sub-sectors in the overall system or model diagram. System dynamic approach methods provide valuable tools to depict the causal loop diagrams and stock and flow diagrams. (Wolstenholme E. F., 2004), (Gorschek, T., & Davis, A.M., 2007) and (Michael Mutingi, et al., 2017)

Quality assurance, rework and testing are central processes considered key towards REPI and software quality improvement. Staff productivity is major in achieving the three central processes. The client is important in testing the software product quality. The final model demonstrates the dynamic hypothesis for the proposed REPI enhancements through the stock and flow diagrams, depicting the major parts as well as the possible variables.(D.W., Williams , 2000) and (Williams, D., 2003a, 2003b), (Zawedde, A. &Williams, D., 2013, 2014), (Zawedde, A.S.A, et al, 2011) and (Zawedde, A., 2016)

4.2 Model Variables

Model variables can be endogenous while others exogenous. System dynamics behavior and patterns emerge from the endogenous variables. Some variables are included or excluded in the model. The core aim of the model is to identify the causal relationship of the production process from the project launch time to the testing phase. It is therefore clear that the model assumes planning and budgeting activities are pre-determined and initiated. The assumptions are therefore that the process to determine the

system's requirements is finalized, funds allocated as well as determination of the workforce capabilities. Hence considering the factors exogenous. The model excludes post-development activities, efforts, post-production, maintenance and system re-design. The model includes endogenous activities that affect production activities such as design, coding and quality assurance, re-work and testing. (Zawedde & Williams, 2013) and (Yaniv Mordecai & Dov Dori, 2017)

The table below gives examples of endogenous, exogenous and excluded variables.

Table 4.1: The REPI Model: Endogenous, Exogenous and the Excluded Processes. [*Adapted from Yaniv Mordecai and Dov Dori, (2017*)]

Endogenous	**Exogenous**	**Exempted activities**
Software design activities	Testing work force needed per activity.	Software maintenance
Software Coding	Coding effort required.	Consultancy/far staff
Quality Assurance and Rework	Testing effort overheads.	Requirements definition
Software Testing	Maximum tolerable staff exhaustion	Requirements definitions
Human Resource Management	Exhaustion	Clients pressure

The table above shows endogenous processes and excluded activities according to the identified boundaries. The goal is to identify the causal relationship and the resultant behavior pattern(s) generated between the actor(s) within the production cycle. This excludes the consultancy or the far staff needed. The preliminary planning activities and requirements defined excluding the project-budget since it lies outside the main software development activities. The model assumes that the client's demands are constant. The model itself excludes the clients. Hence, the development team un-subjected to the client's pressure and regarded to be agents outside the base model. These are the profound main boundaries of the base-model. (Abdel-Hamid, 1991: p.20) and (Michael Mutingi, et al., 2017)

Literature review reveals that software quality and "*customer satisfaction*" are related; hence, development acceptance, testing, end-user reviews and the client should be included as important factors in the model. These factors eventually lead to client satisfaction. To review the satisfaction levels, a model boundary is included to link the work force allocated by management to the work force allocated by the client to the client-reviewing process. The aim is to expand the model's boundary, to capture and enhance software quality as well as emerging software quality policies. The table below indicates new-model boundaries for the proposed enhanced base model.

Table 4.2: *Endogenous, Exogenous and Excluded Processes in the Enhanced Base-Model. [*Adapted from Cuellar, (2010)]

Endogenous	**Exogenous**	**Exempted activities**
Software Design	Client re-viewing work force.	Software maintenance
Software Coding	Client work force to review.	Consultancy/far staff
Quality Assurance and Rework	Client manpower to review	Requirements definitions
Client testing	Client work force productivity and efficiency.	Clients pressure
Human Resource Management		Changing clients demand

The process excludes clients changing demands and pressure. The key determinant variables are the client-testing side staff, experience not affected by the other factors therein.

4.3 System Model Boundary

The system's model boundary provides the scope and applicability to the phenomenon they mean to represent. Systems models are a representation of the real world processes and the relationship between identified variables and recreates them inclusively in the real world. The absence of a clear system boundary may result to variable's displacement and observed behavior details. According to Sterman, 2003), it is important to have an unambiguous boundary.

4.4 Time Scope

Time's horizon varies from project to project, project size, complexity as well as scope. From Abdel-Hamid's original model, (Abdel-Hamid, (1991) analysis, for medium projects the paper considers projects that run for about 430 days having an average project delay of 33%. Considering the previous research studies/projects did not resolve the "software crisis" to include all the project activities from start to the end the study considers 800 days (2.1 years) to cater for the other factors that are exogenous to affect the set time scope.

4.5 The System Model Structure for the USREPM System

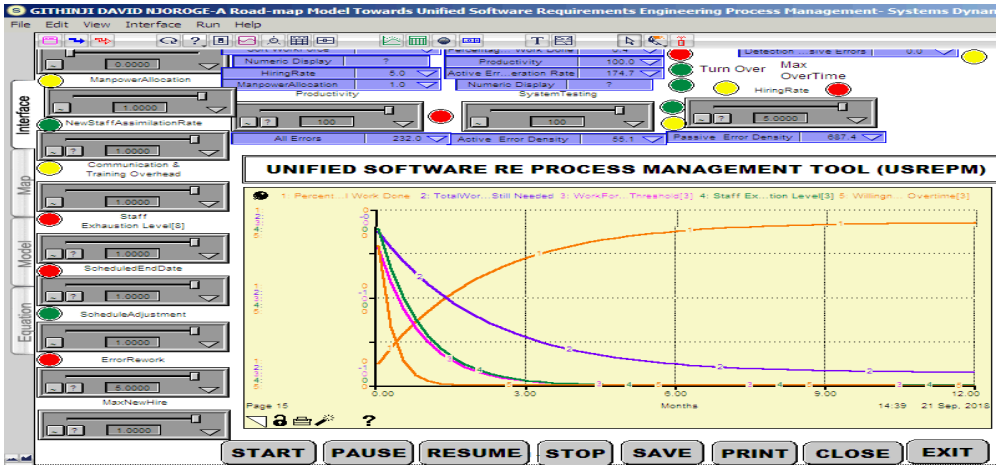

Figure 4.1: The USREPM System Model Structure

4.5.1 User Interface:

The USREP interface structured using Stella version 9.0.2, contains links to the model design map that contains the stock and flow diagrams (SFD), and the model equations.

4.5.2 System Stock and Flow Diagrams (USREPM System/Subsectors)

4.5.2.1 Software Project Management Sub-System/Sector

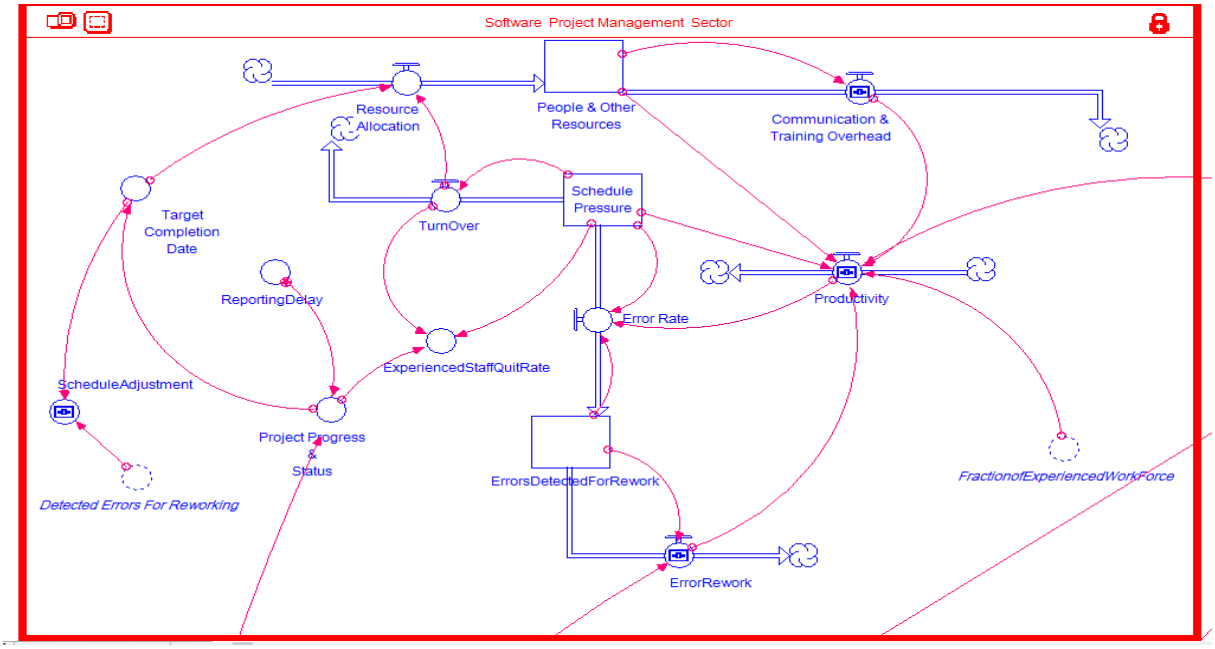

Figure 4.2: Software Project Management System/Sector

4.5.2.2 Human Resource Management System/Sector

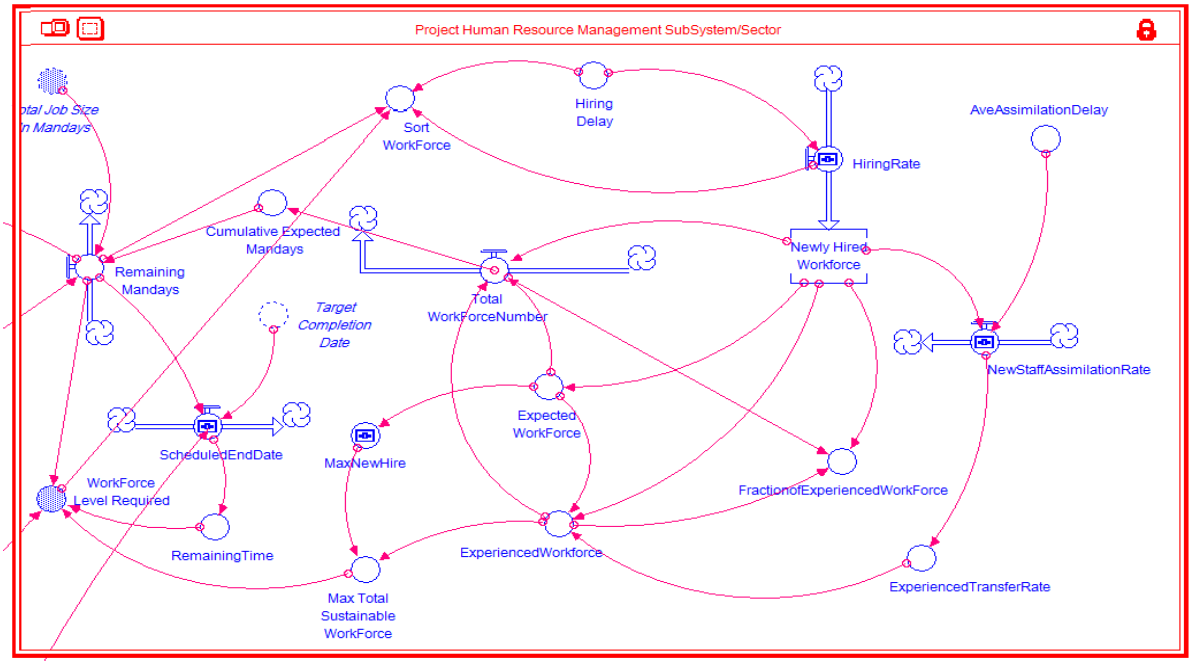

Figure 4.3: Human Resource Management System/Sector

4.5.2.3 Manpower Allocation Sub-System/Sector

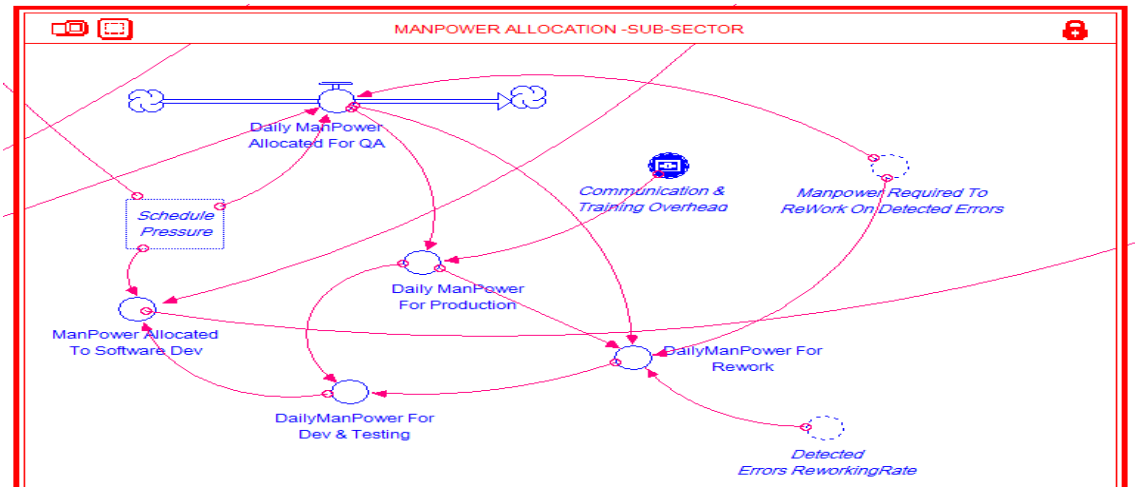

Figure 4.4: Manpower Allocation Sector

4.5.2.4 Development & Productivity Sub-System/Sector

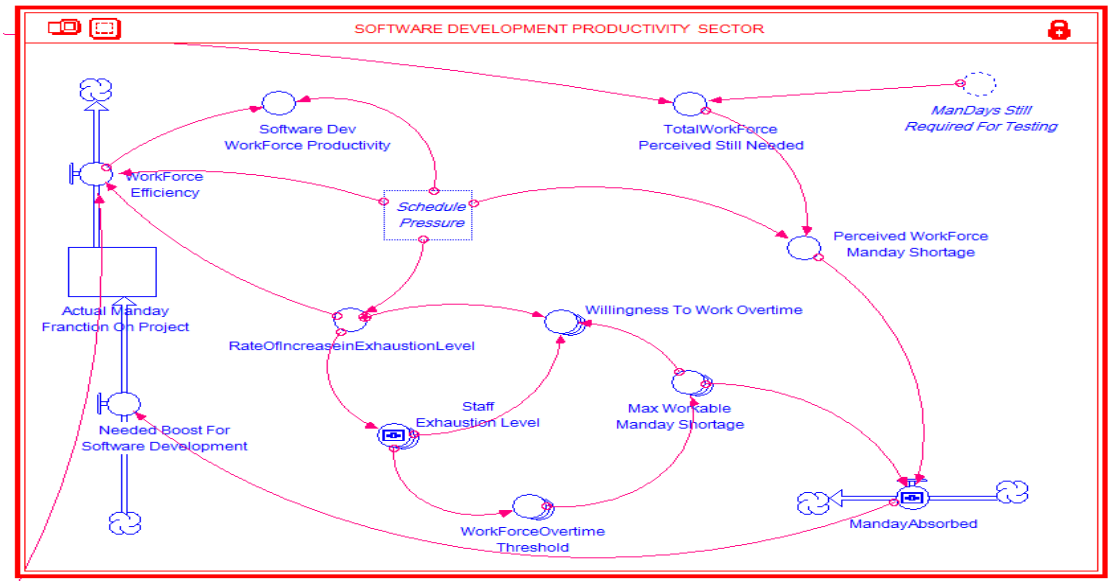

Figure 4.5: Software Development & Productivity Sub-System/Sector

4.5.2.5 Quality Assurance & Re-Work Sub-System/Sector

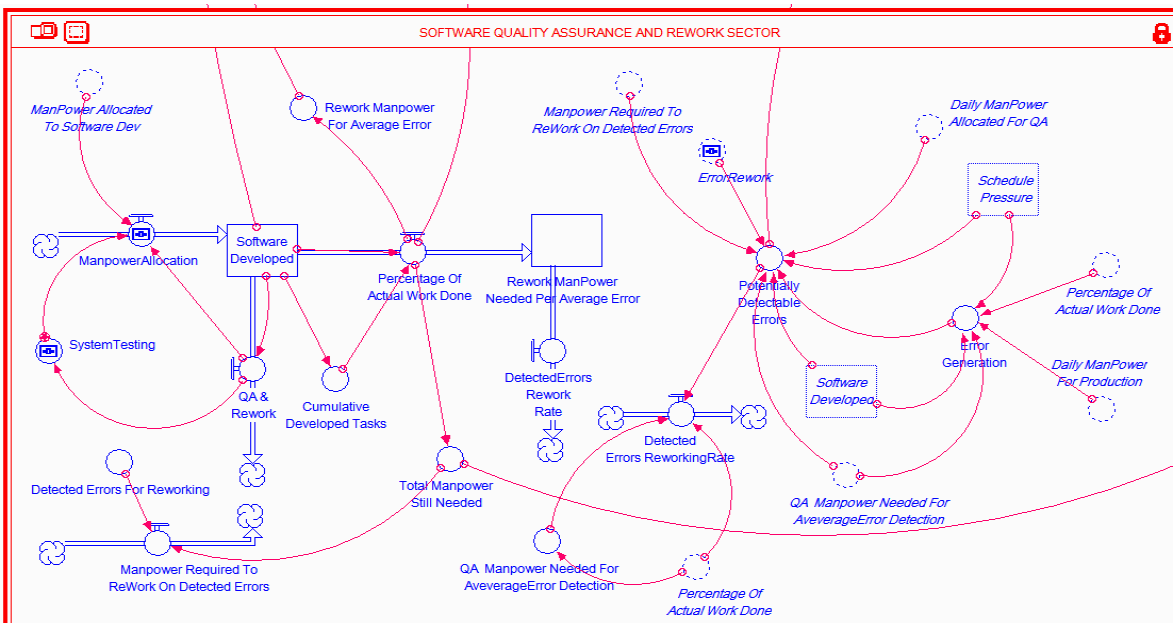

Figure 4.6: Quality Assurance & Re-Work Sub-System/Sector

4.5.2.6 System Testing Sub-System/Sector

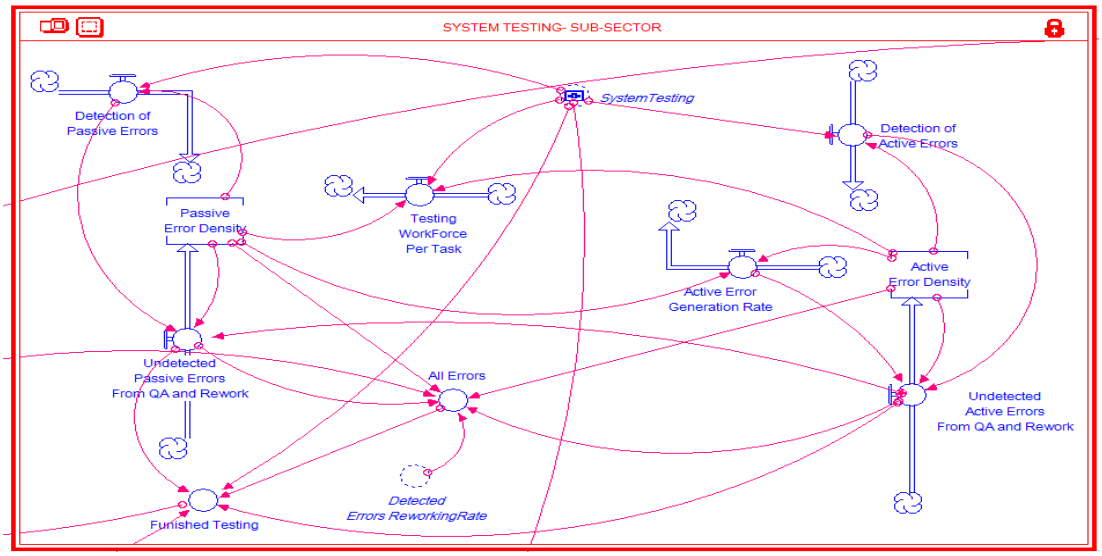

Figure 4.7: System Testing Sub-System/Sector

4.5.2.7 Controlling Sub-System/Sector

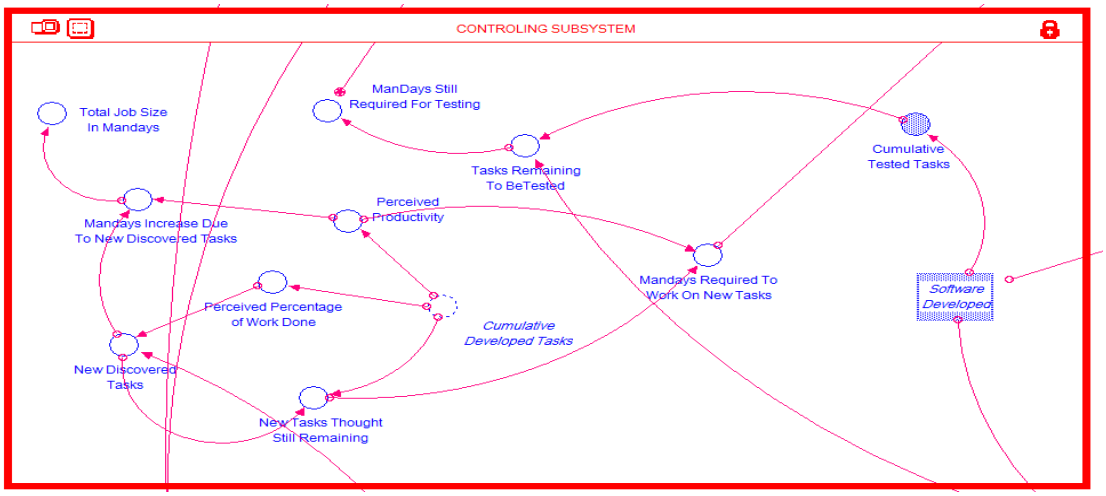

Figure 4.8: Controlling Sub-System/Sector

4.4 System Model Stock & Flows Diagram Relationships

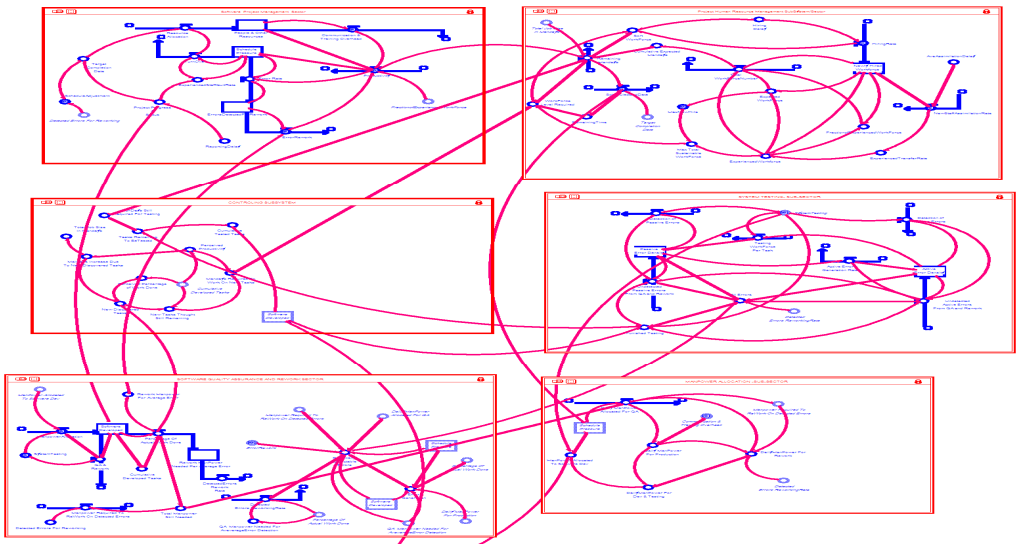

Figure 4.9: USREPM System SFD/ Relationship Diagram

4.5 Causal Loop Diagrams (CLD)

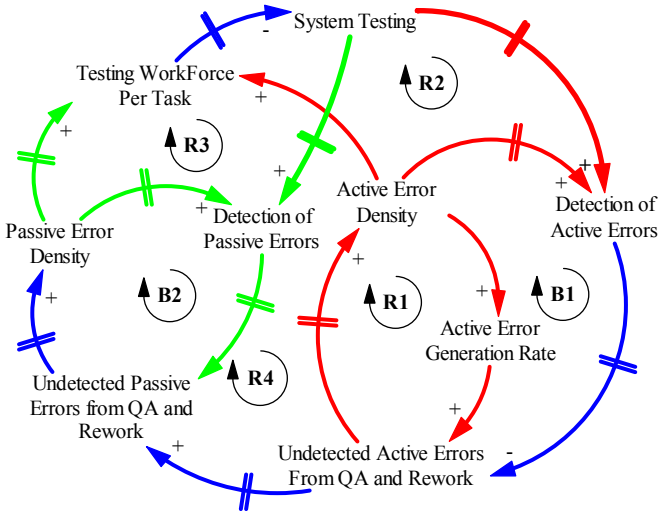

Figure 4.10: System Testing Causal Loops Diagram

The system-testing subsector is the final sector of the software production cycle.

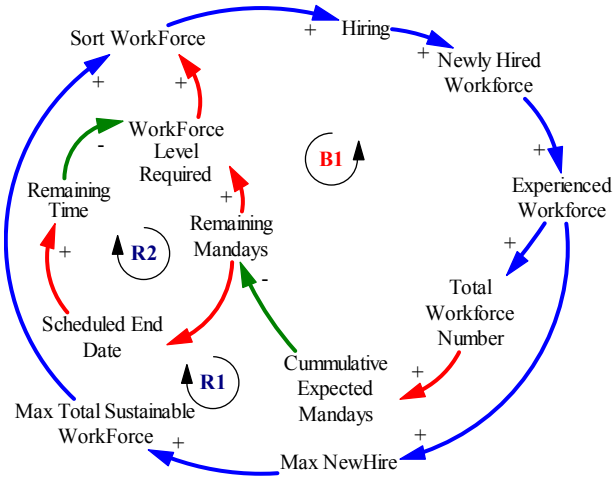

Figure 4.11: Human Resource Management Causal Loops Diagram

Experienced workforce positively increases the total workforce number as well as the accumulative expected mandays. (Chapter 5: Human Resource Management Subsystem), this subsystem manages and controls the hiring and management of the total project work force. (Zawedde, A. & Williams, D., 2013)

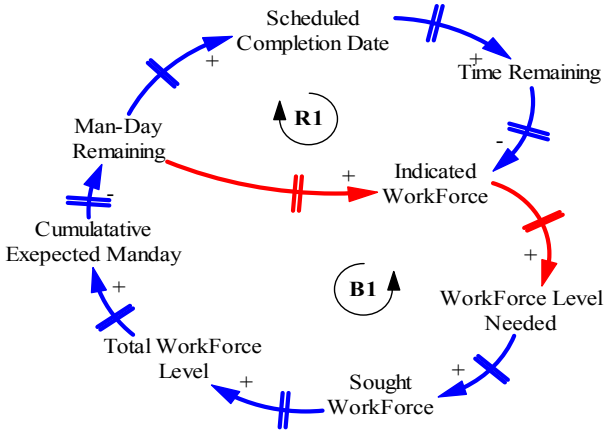

Figure 4.12: Planning Subsystem Causal Loops

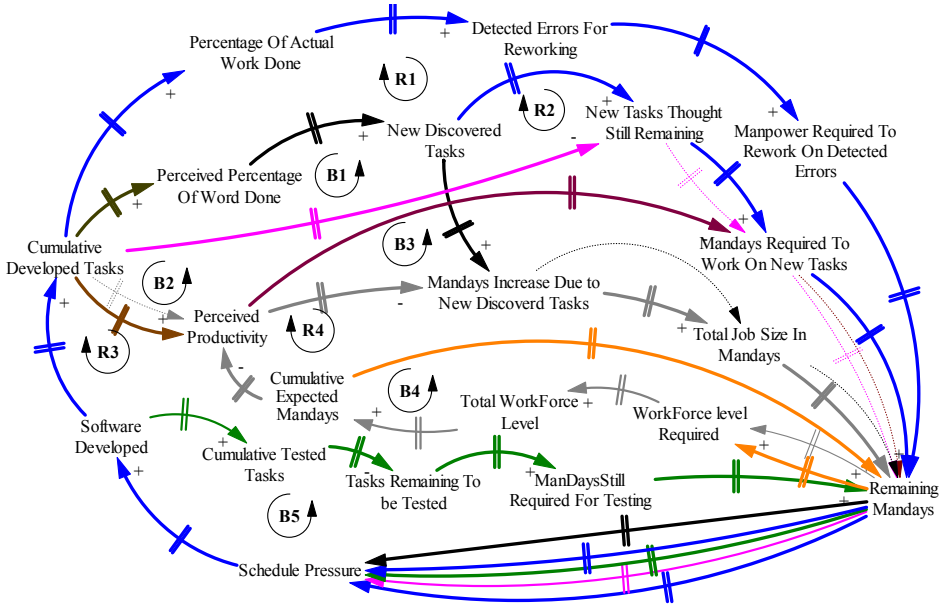

Figure 4.13: Controlling sub-system Causal Loops Diagram

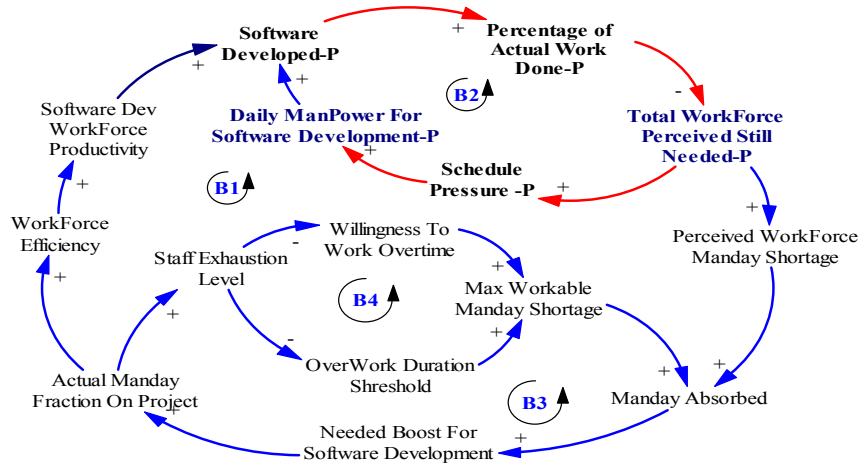

Figure 4.14: Software Development Productivity Sub-sector Causal Loops

The sub-sector's focus is on the software development phase. Management conceive and make predictions of the rate at which software needs to be developed. (See Chapter 5: -Software Development Productivity Subsectors)

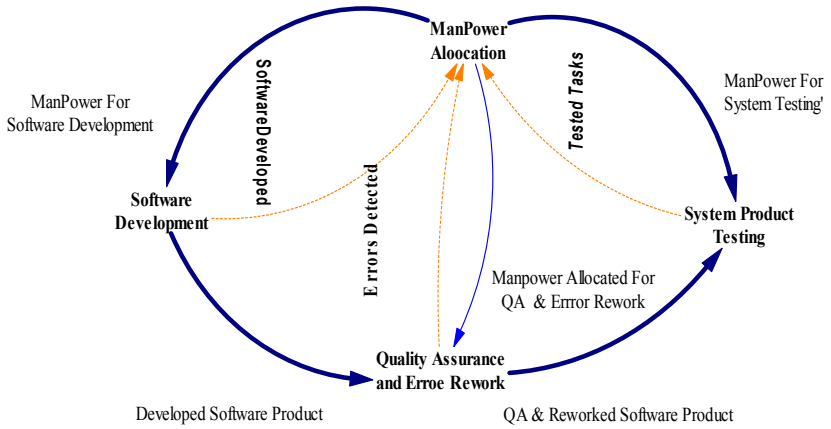

Figure 4.15: Software Production Sub-Systems Causal Loop Diagram

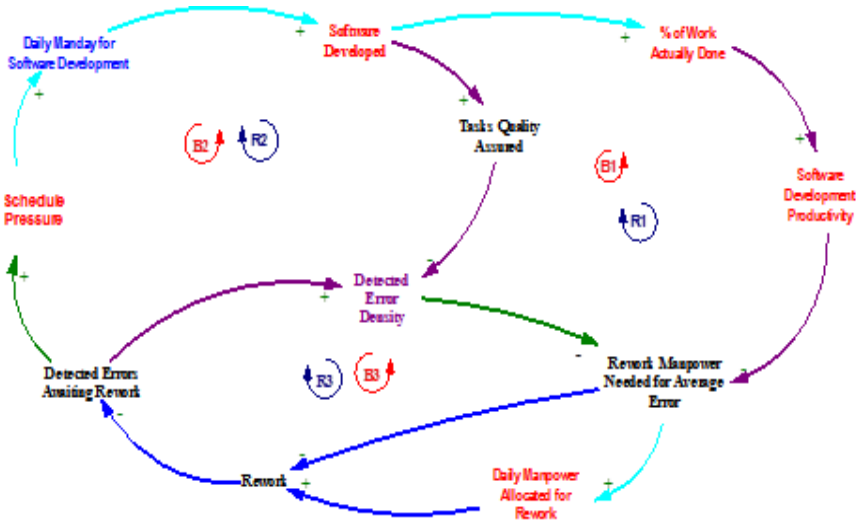

Figure 4.16: Quality Assurance & Re-Work Sub-Sector

The rework process involves staff drawn from the QA team who identify and fix errors (see Chapter 5: Rework Manpower Effort)

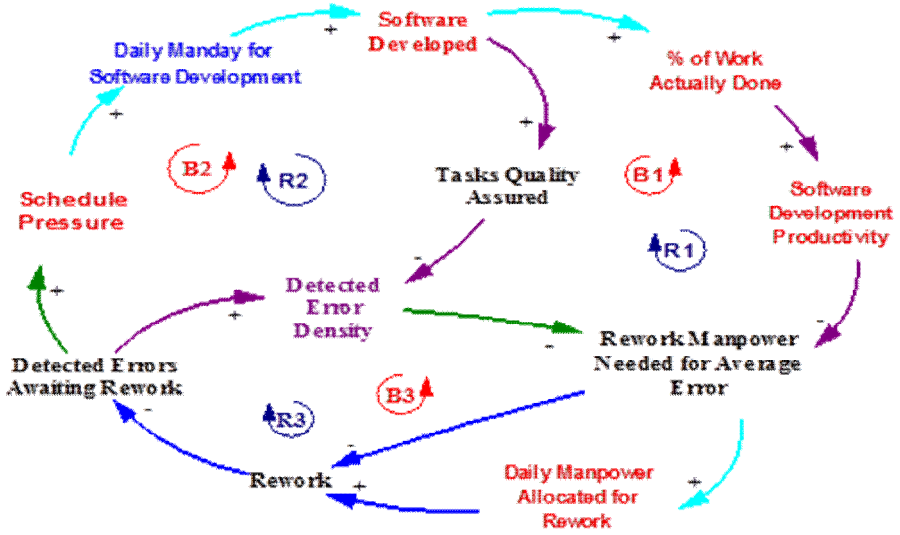

Figure 4.17: Enhancement of Effort for Re-Work Process. (Zawedde, A., 2016) (Chapter 5: Rework Manpower Effort)

Error Detection and Rework Detection Rate

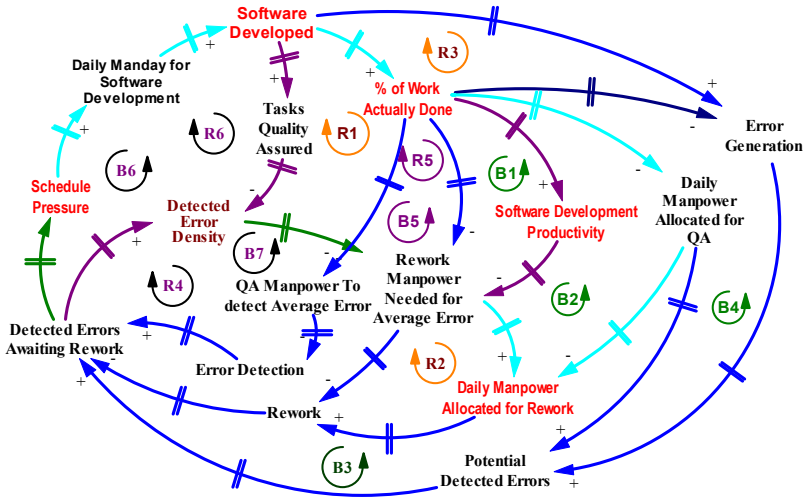

Figure: 4.18**:** Error Detection and Rework Detection Rate

Quality Assurance Sub-System & Rework /Sub-Sector

Quality Assurance Sub-systems and sub-sectors focus their attention to error detection (see Chapter 5: Software Errors Rework Process, Rework Manpower Effort, Error Detection Rework & Error detection rate).

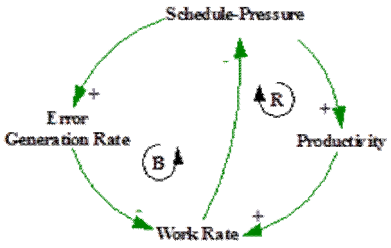

Figure 4.19: Effects of Error Generation Rate CLD)

Schedule pressure affects error generation, work rates and work force productivity (Chapter 5: Error Generation and Error Generation Rate). (Cooper et al., 2009), (Zawedde A & Williams D., 2013) and (Zawedde, A., 2016)

Error Densities

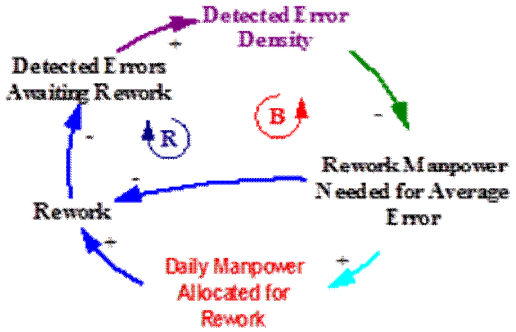

Figure 4.20: Effects of Error Densities on Re-Work and QA Staff Allocation (CLD)

Error detections in various software development stages increase error densities (see Error densities, error detection and rework detection rate in chapter five)

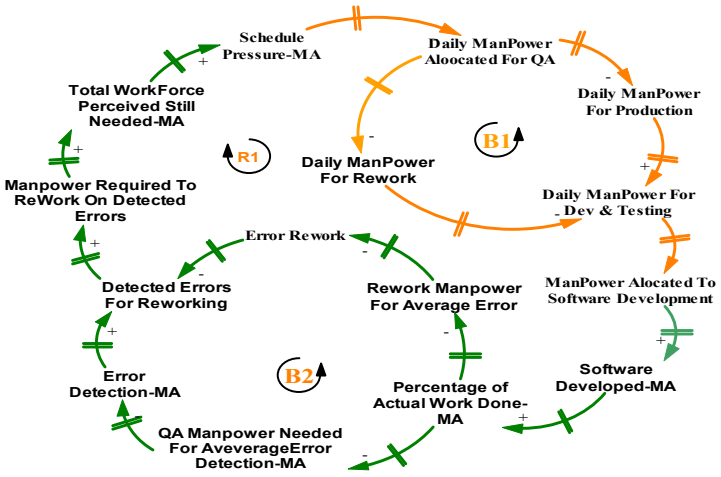

Figure 4.21: Manpower allocation Sub-system/sub-sector CLD.

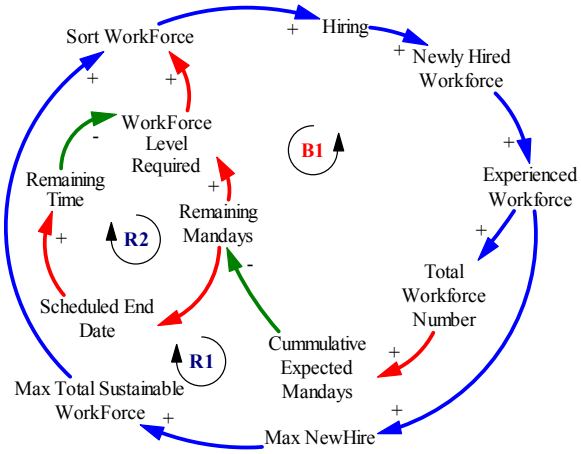

Figure 4.22: Human Resource Management Causal Loops Diagram

4.6 Model Simulation Graphs/Results

Software Errors Rework Process

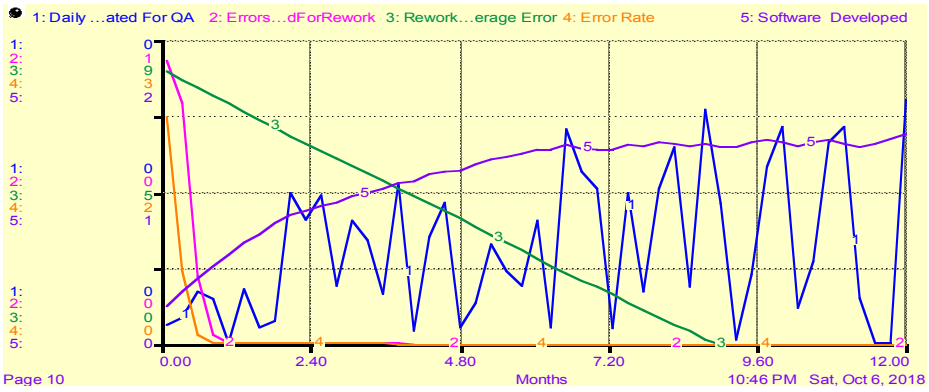

Figure 4.23: Effect of Error Rework Processes on software production

The error rework process focuses on the REPI and software quality. (*See Chapter 5: Software Error Reworks*) (Zawedde, A., et al... 2011, 2014, 2015) and (Zawedde, A., 2016)

Re-Work Manpower Effort

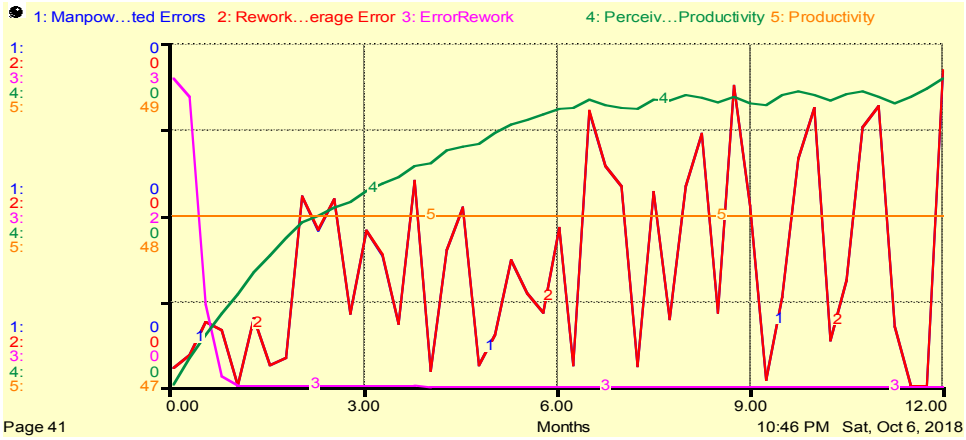

Figure 4.24: Effects of Rework Manpower Effort on Software Quality

Error density in software heavily affects its quality. (Zawedde, A. & D. Williams, 2013, 2014) and (Zawedde, A., 2016) (*See Chapter 5: Rework Manpower Effort*)

Error Rework Rate

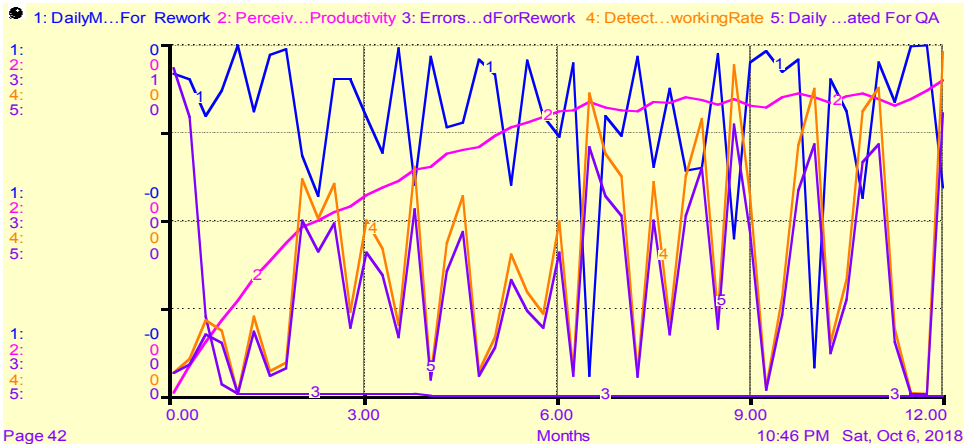

Figure 4.25: Effects of Error Rework Rate on the REPI process

Work force effort allocated rework process depends on the rework job and the number of rework errors awaiting rework and the perceived rework work force productivity. (See rework rate in chapter five)

Error Detection and Error Detection Rates

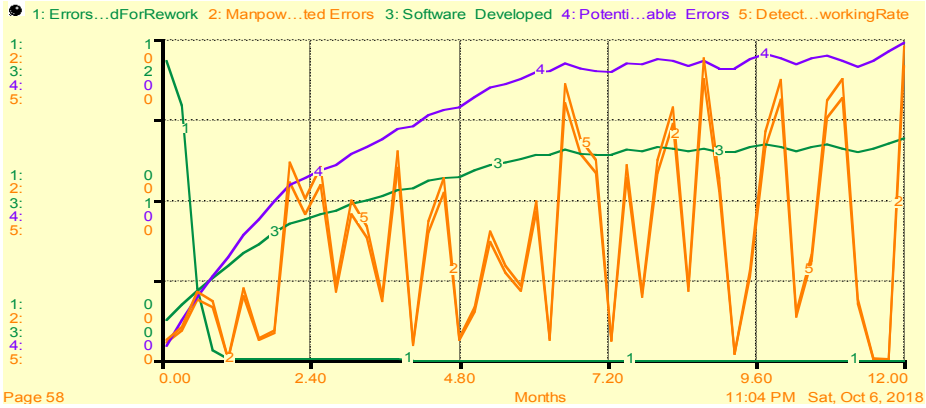

Figure 4.26: Effects of Error Detection and Detection Rate on REPI and Software Quality

Error remains potentially detectable but hidden until tasks review and testing. Some go undetected while others remain to the last stage (testing) and corrected through rework. (Chapter 5: Error Detection and Rework Detection Rate) (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

Quality Assurance Subsystem & Rework /Sub-Sector

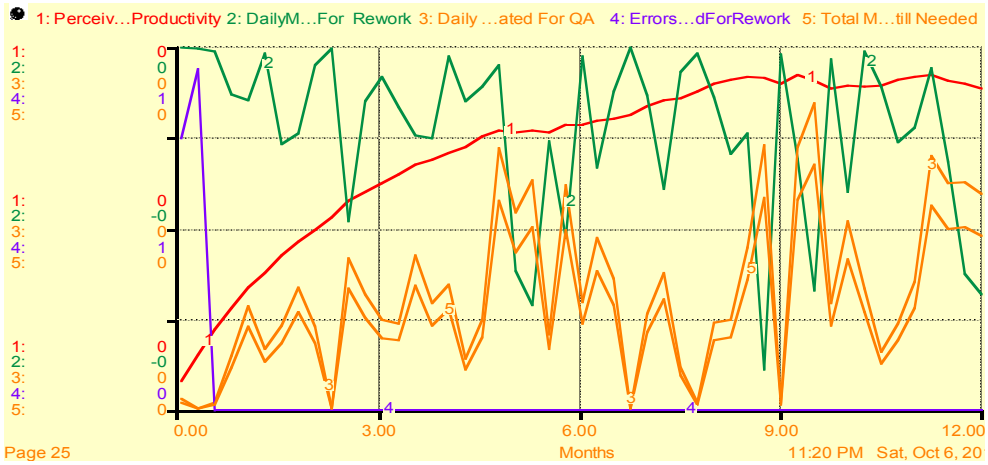

Figure 4.27: Effects of Error Detection Rate on Software Quality Assurance

Error detection as a QA function has a significant effect on the number of errors detected. (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016) (*See Chapter 5: Quality Assurance Sub-system and Rework Sub-sector*)

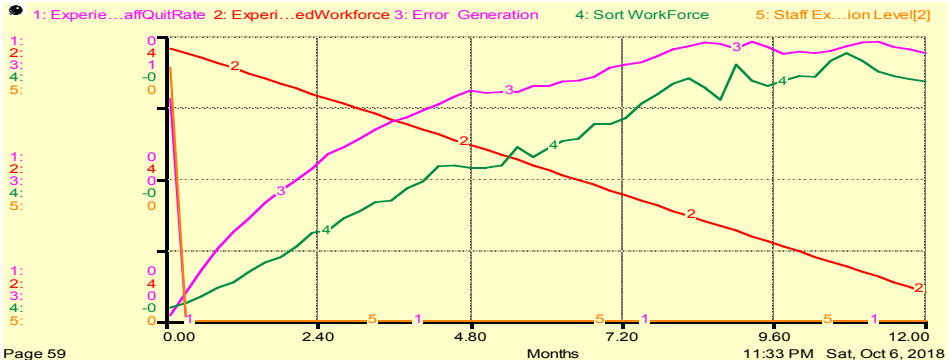

Figure 4.28: Effects of Error Generation Rate on Workforce Level

(*See Chapter 5: Quality assurance sub-system and rework Sub-sector*). (S.C., Davar & M., Parti, 2013), (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

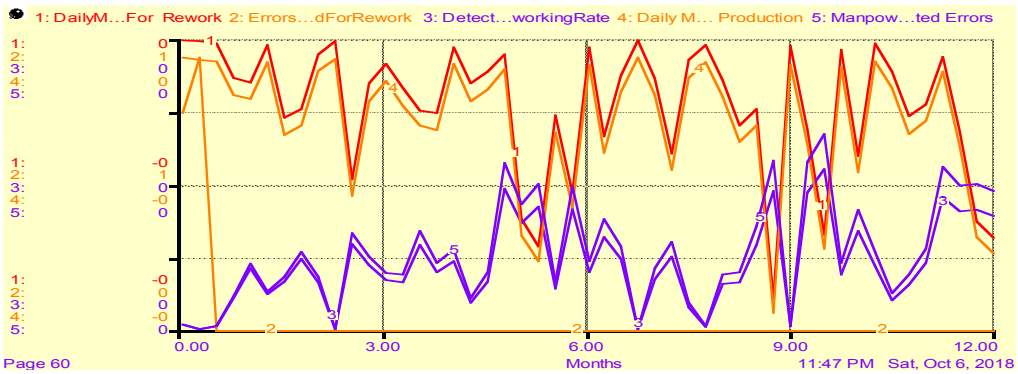

Figure 4.29: Effects of Error Densities on Re-Work and QA Staff Allocation (*See Chapter 5: Error Densities, Error Detection and Rework Detection Rate*)

Productivity, Rework Rate and Work Force Needed

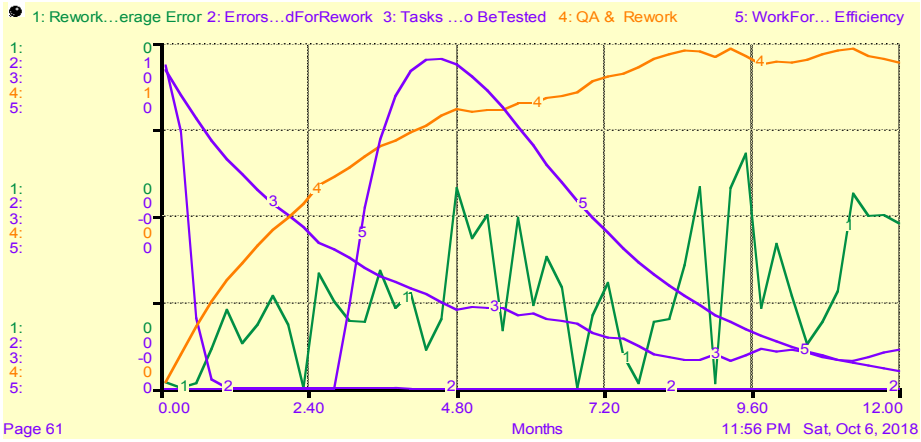

Figure 4.30: Productivity, Rework Rate and Workforce Needed for REPI

Peak productivity can be achieved when a staff or team of staff work at their peak efficiency. (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, A., 2016) and (S.C., Davar & M., Parti, 2013) (*See Chapter 5: Productivity, Rework Rate and Workforce Needed*)

Effects of Productivity, Re-Work Rate and Work Force Absorbed

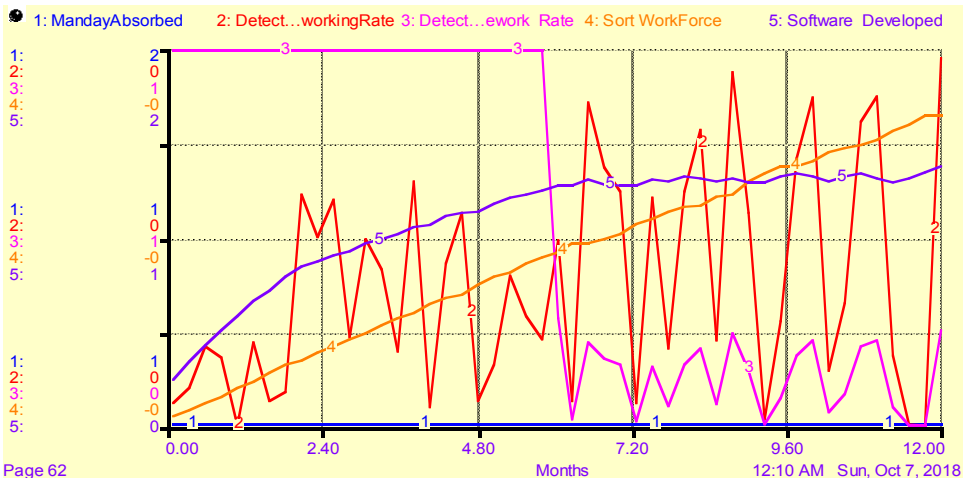

Figure 4.31: Effects of Productivity, Re-Work Rate and Workforce Absorbed. (*See Chapter 5: productivity, Rework Rate and Workforce Needed*) (S.C., Davar & M., Parti, 2013)

Effects of Experience and Learning on Staff Overall Productivity

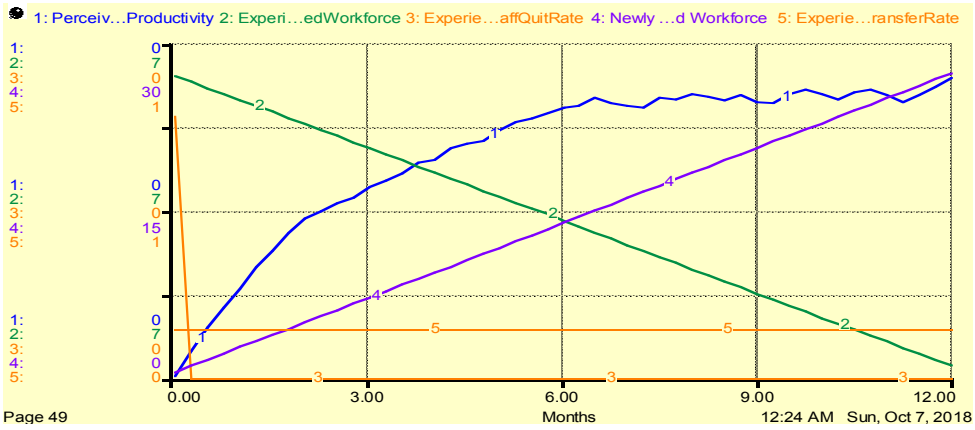

Figure 4.32: Effects of Experience & Learning on Staff Productivity

Experienced staff carry higher productivity and a bigger output potential than the newly hired. (Zawedde, A. & Williams, D., 2013, 2014), (S.C., Davar & M., Parti, 2013) and (Zawedde, A., 2016). (*See Chapter 5: Effects of Experience and Learning on Staff Overall Productivity*)

Effects of Communication on Staff Allocations

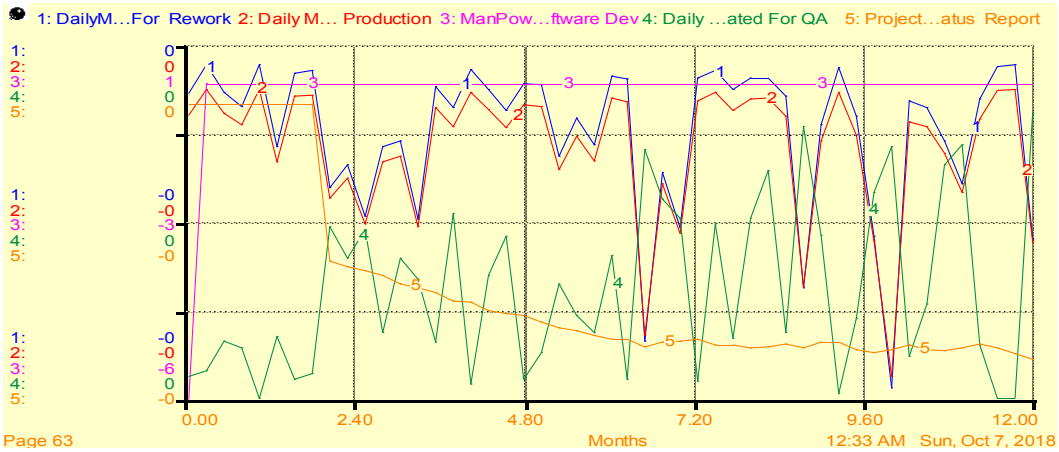

Figure 4.33: Effects of Communication and Briefing on Project Staff Allocations

Communication by nature in the project is an overhead. (D.W. Williams, 2000), (Williams, D., 2003a, 2003b), (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, A.S.A. et al., 2011) and (Zawedde, A., 2016)

Manpower Per Average Error against Efficiency

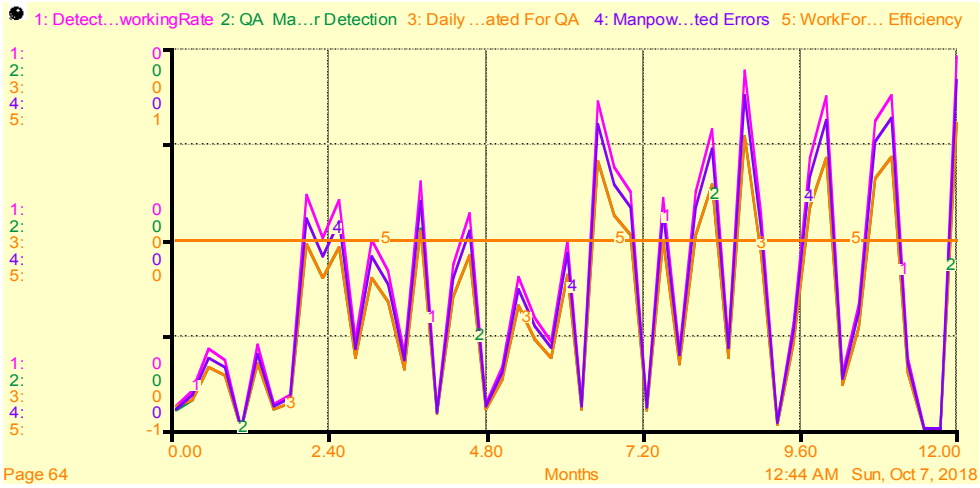

Figure 4.34: Manpower per Average Error against Efficiency

Wrong decisions on the average number of errors that each staff can handle per day may lead to overworking and loss of motivation. (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, et al., 2011), (S.C., Davar & M., Parti, 2013) and (Zawedde, A., 2016) (*See Chapter 5: Effects of Communication and Project briefing on Staff Allocation*)

Software Bugs Fixing

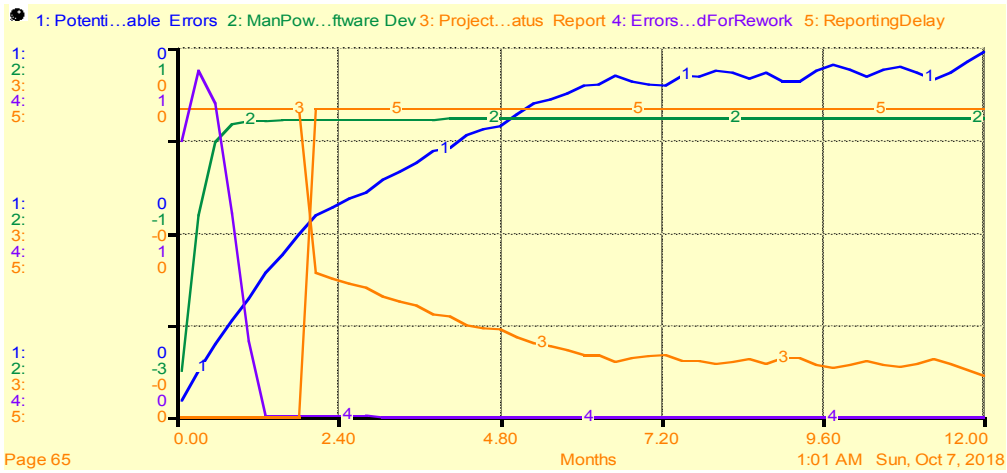

Figure 4.35: Effect of Early Error Detection and Fixing (Goal Seeking)

Error fixing in REPI is a development function. (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, A.S.A. et al., 2011), (Putnam-Majarian, T. & Putman, D., 2015) and (Yaniv Mordecai & Dov Dori, 2017) **(***See Chapter 5: Bug Fixing*)

Staff Motivation and Exhaustion

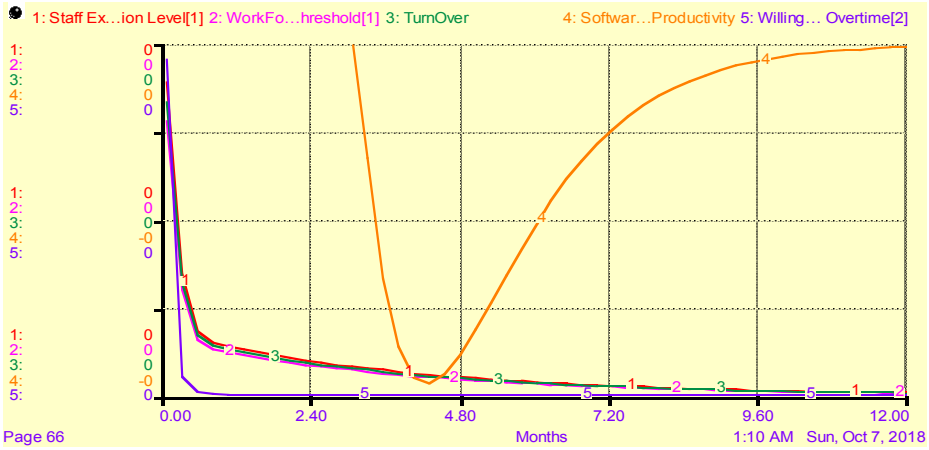

Figure 4.36: Effects of Over-Work and Staff Motivations on Productivity (S-Shaped)

Staff motivational factors may remain constant or change over time during software production process. (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, et al., 2011), (S.C., Davar & M. Parti, 2013) and (Zawedde, A., 2016) (*See Chapter 5: Staff Motivation*)

Effects of Workforce Exhaustion on Software Development Projects

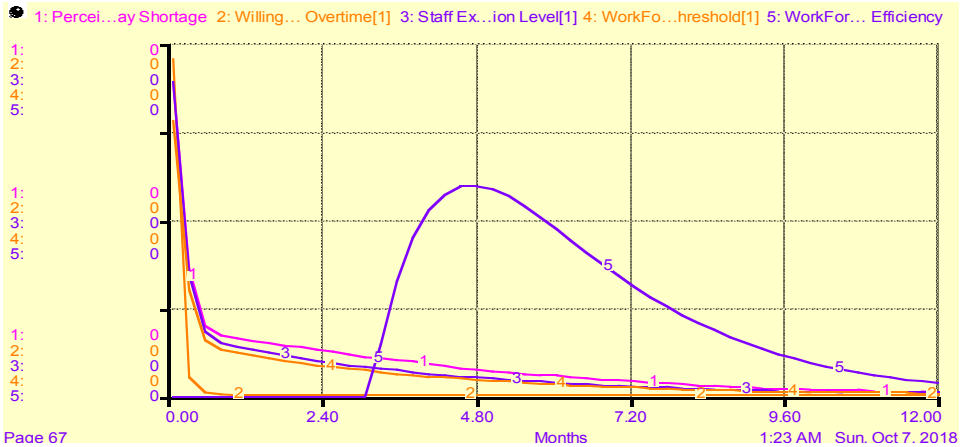

Figure 4.37: Effects of Workforce Exhaustion on Software Development

Staff exhaustion due to overwork pressure tends to have a negative effect on workforce productivity rate over time in a project. (Zawedde, & Williams, D., 2013, 2014), (Zawedde, A.S.A et al., 2011), (Zawedde, A., 2016), (Williams, D. (2003a, 2003b) and (Kamuni, S.K., 2015)

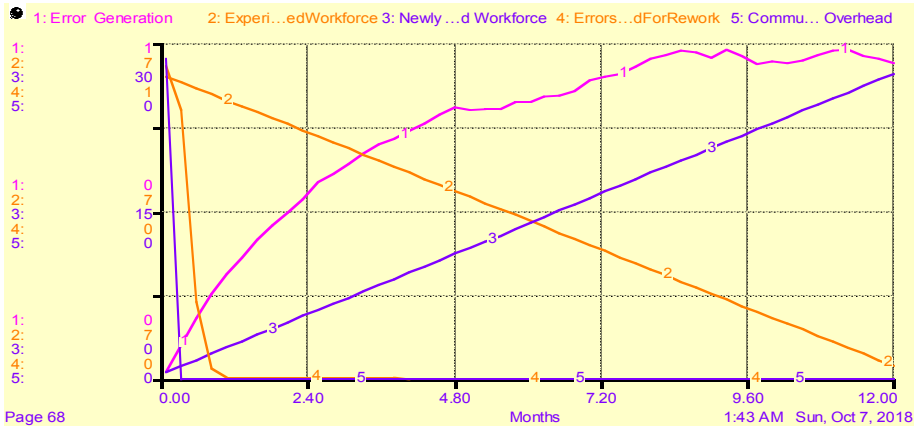

Figure 4.38: Effects of Staff Experience on Software Process

Experienced staff better detect and generate fewer errors, handle more tasks at a time and effectively rework on detected errors. (D.W. Williams, 2000), (Williams, D., 2003a, 2003b), (Kamuni, S.K., 2015) and (Zawedde, A., 2016)

Determination of Project Staff Levels in Software Projects

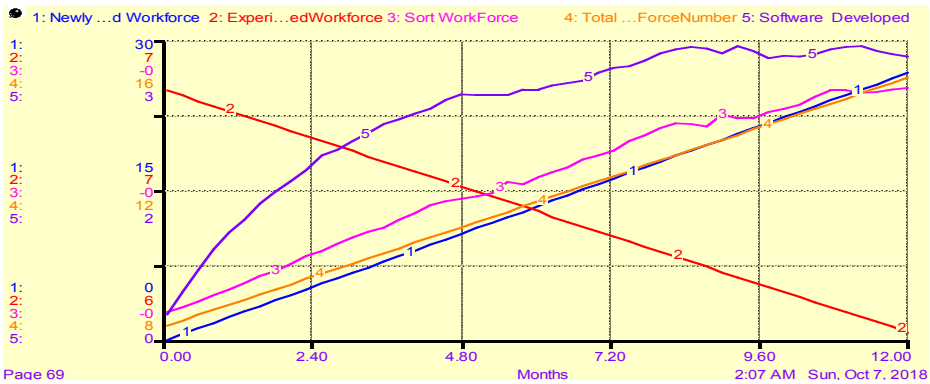

Figure 4.39: Decisions to Determine Project Work-Force Level

The management determines the necessary staff level to complete the project within the scheduled time based on the perceived remaining tasks. (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, A.S.A. et al. 2011, 2016), (D.W. Williams, 2000) and (Williams, D., 2003a, 2003b) (Chapter 5: Determination of Project Staff Levels)

Effects of Turn-Over on Projects

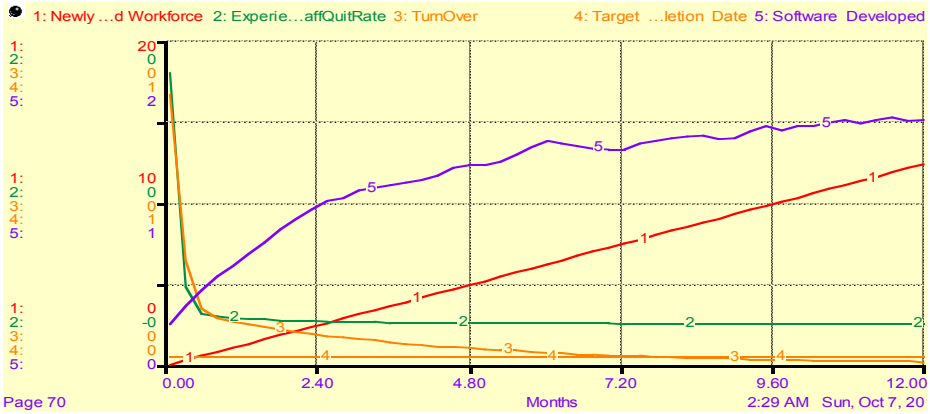

Figure 4.40: Effects of New Hire, Staff Assimilation and Turnover on Workforce Productivity

Turnover affects the workforce and expresses itself as the quit rate of the experienced workers. (D., Bator, 2014) (Zawedde, A. & Williams, D., 2013, 2014), (D.W. Williams, 2000), (Williams, D., 2003a, 2003b), (Kamuni, S.K., 2015), (Zawedde, A.S.A. et al., 2011) and (Zawedde, A., 2016), (*See Chapter 5: Turnover*)

Cost of Errors on Projects

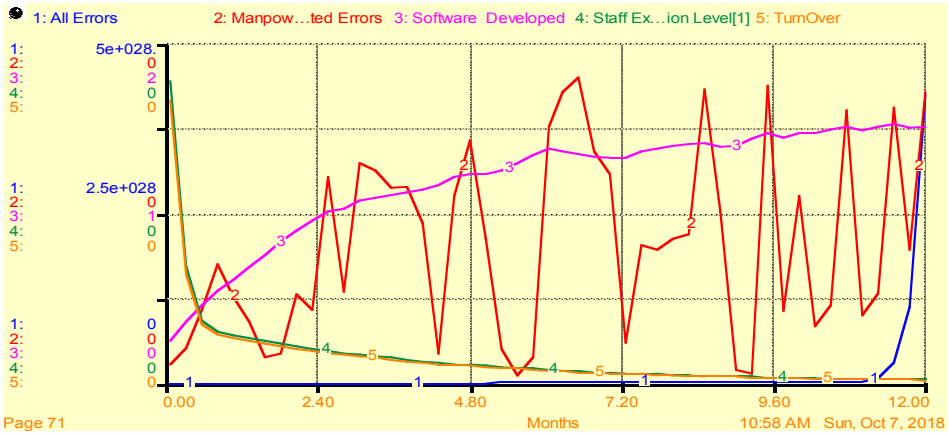

Figure 4.41: Cost of Errors on Software Projects

The QA policy has a significant impact on the total project cost. (Williams, D., 2003a, 2003b), (Svahnberg M.T., Gorscheck, R., Torkar, S., Saleem, B. & Shafique, M.U., 2010), (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde A, 2016) and (Kamuni, S.K., 2015) (*See Chapter 5: Cost of Errors*)

CHAPTER FIVE: DISCUSSION OF RESULTS

5.1 Introduction

Chapter five forms a detailed discussion of results obtained in the fourth research strategy (research design) and a strong foundation for quantitative research founded from the fifth research strategy. In the chapter, the researcher discusses the model's stimulation results (*Stella graphs*) based on associated causal loops in chapter 4. The simulation's behavior is compared with past findings and accounts for the researcher's opinions for and against the existing REPI models such as Abdel-Hamid's base model and the current REPI process, methods, procedures and the causes of software failure. At the end of the chapter are equations arrived at to achieve results of the CLD variable relationships. (Abdel-Hamid, T.K., 1991), (Hekimoglu, M. & Barlas, Y., 2010), (Zawedde, A.S.A et al., 2011), (Annet Reilly, 2017), (Zawedde, A. (2016) and (Michael Mutingi, et al., 2017),

After the model's structural development, system simulations follow to achieve the goal of validating the problem statement and literature review finding. (Ferraira, et al., 2009) and (Friker, S. & Glinz, M, 2010)

From literature's review, Abdel-Hamid's base model provided empirical data to offer a range of between 6 to 10 percent with a constant industrial average of 7.5 % of successful software projects. Recent literature reviews gave an average of 7%. This is however contrary to the true project report of bad fixes, which stands at 2%. Very unsuccessful projects give as high as 25% of bad error fixes generation. (Jones & Bonsignour, 2012), (IEEE, 2017), (Eveleens, L. & Verhoef, C., 2010), (Daneva, M., 2016) and (Hastie, S., 2015)

The other area of focus is the rate at which the rework process generates bad fixes. The error density, schedule pressure and the levels of experienced work force influences the rate of bad error fix generation. Bad-fixes are an influencing factor or source of increased re-work and constitute to a big percentage of errors occurring in the system (Jones, 2007) (Glinz, M. & Fricker S., 2013) and (Zawedde, A., 2016)

Causal loop diagrams work as important tools used for identifying the possible logical links between various variables inside a dynamic system. The resultant causal loop diagrams that identify all variables and factors that influence the error density on the rework process, form part of the larger overall model

connection. All the sub-systems will be enhanced further to form part of the larger system in the overall model design. (Williams, D., 2003a, 2003b)

5.2 Software Errors and Rework Process

The rework process identifies the REPI process as one of the main focal points of software quality improvement. In the original model Abdel-Hamid, T.K., (1991), the rework process missed out two important concepts that greatly influence software product quality.

One of the key areas of REPI is the effort required during the rework process. In the original model, the rate of rework effort greatly influenced is by the effort needed per average error during the rework process. The determinant factor is the rework work force needed to re-work on a specific type of error and the effects of communication and overhead losses. These two factors affect the rework rate. Error density and staff productivity influence the rework process needed per average error. (Mijwaat, R., 2012), (Lech, P., 2013), (Pruyt, E., 2010, 2013), (Pranjali, K. & Dhananjay, S., 2014), (Putnam-Majarian, T. & Putman D., 2015) and (Zawedde, A., 2016)

5.3 Re-Work Manpower Effort

Abdel-Hamid, T.K., (1991), Putnam-Majarian, T. & Putman, D. (2015), Williams D. (2003a, 2003b), Zawedde, A.S.A. et al., (2011) and Zawedde, A., 2016) put forward assumptions that error densities heavily influence the REPI and software quality assurance.

As software development shifts from the design to the coding phase in the design, the effort required to detect average error is higher than in the coding phase. The effect relates to the fact that design errors are artful and hence harder to detect than coding errors. The errors are more in the coding phase. This implies that the error density increases from design to coding. This increase in density provides more information on hidden errors. (Abdel-Hamid, 1991: p.105), (Solomon, B., Shahibuddin, S. and Ghai, A. 2009) and (Yaniv Mordecai & Dov Dori, 2017)

The rework process involves staff members from the production team. The team identifies the errors and fixes them or gather errors awaiting rework. The QA team gathers coding errors reveals design errors and hands them over to the rework team. Design errors are harder to rework hence the average effort required per error is greater than for coding errors. Design errors are not passive and they themselves generate code-errors with a higher error rate and increases the error density. The Quality Assurance and Rework Sub-Sector with new loops shown in chapter 4 provides improvements enhancing the effort for

the rework process. (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, A., 2016) and (Pruyt, E., 2010, 2013)

Re-Work Manpower Effort Balancing Loop (B1):

This loop relates to the percentage of work done on the project, development productivity, and the rework effort process. Percentages of work-done increase as production proceeds, learning continues and the team gains a clear picture of the software develop and how it works. The learning, experience and knowledge sharing (S.C., Davar & M. Parti, 2013) process allows for an accelerated production rate and in a more efficient manner increases the rate of error detection, error correction and rework. Eventually, the required effort per average error decreases and outputs a negative relationship between productivity and the effort per average error. (Zawedde, A.S.A. et al., 2011)

Re-Work Manpower Effort Balancing Loop (B2):

A decrease in the workforce needed per average error has direct influence on the daily work force required/allocated for rework. Decrease in the effort to correct errors lowers the daily work force allocated for rework. A lower daily work force allocated for the rework effort decreases the rework rate, increases detected errors awaiting rework. Increase in the schedule pressure prompts management to make drastic decisions to increase the rate of software development. The adjustments undertaken eventually lead to an increased development and production rate. (Williams, D., 2003a, 2003b) (Zawedde, A. & Williams, D., 2013).

Re-Work Manpower Effort Balancing Loop (B3):

This loop concerns itself with the efforts to improve software production, error rework and efforts triggered by the detected error density, quality assurance efforts towards REPI and high quality software. A lower error detection density triggers the emergence of a harder rework process performance and influence in the rework work force required to work on an average error. On the contrary, when error density is high, an error pattern learning process (S.C., Davar and M., Parti, 2013) trickles in on how well to handle the existing as well as emerging errors. This effort results into a decrease in the rework effort required to resolve an average error and reduction of errors awaiting rework, decreasing the error density. (Philip Morris International, 2015), (Zawedde, A. and Williams, D., 2013), (Zawedde, A, 2016)

Re-Work Manpower Effort Reinforcing Loop (R3):

The final reinforcing loop in the system concerns itself with the relationship between the number of errors detected (Error density) and the rework effort needed and the process itself.

An increase in the error density as software development as it moves from design to coding phase, translates into an increase in coding errors and the generated error pattern makes it easier to resolve them. The error pattern learning, though expectations are that an increase in the error density leads to a decrease in the work force needed to per error. With the observed decrease in the workforce needed per error, downward adjustments made are to the work force allocated to rework on errors. With a reduction in the work force allocated for rework, the rework-rate decreases tremendously as well. This leads to an increase in the number of detected errors awaiting rework. The number of errors detected increases because of a decreased detection and rework rate. (Williams, D. 2003a, 2003b), (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

5.4 Software Error Rework Rate

Detected errors throughout the quality assurance activities management step in and assign staff to correct the identified errors. (Sterman, J.D., Oliva, R. & Linderman, K. & Bendoly, 2015)

The work force effort allocated to the rework process depends on the rework job, rework errors awaiting rework and the perceived rework staff productivity. (S.C. Davar & M. Parti, 2013). The desire is to set goals for predefined amount of error rework per day (Perceived Error Correction Rate). Delays occur since the production process does not correct all errors generated. When a workforce assigned to error corrections diverted is from production, bulk workforce allocated assigned is for quality assurance. (Williams, D. 2003a, 2003b), (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

5.5 Error Detection and Rework Detection Rate

Errors remain potentially detectable and hidden until task reviewing and testing. During this phase, some errors go undetected and corrected are through rework while some may remain undetected only at the late stages. Error detection is a critical essence of quality assurance through activities such as reviews, walk-through, code reading and inspection. QA therefore focuses its attention to error detection. The backlog may initially be zero when all identified are errors.

The major problem with QA is the uncertainty underlying defining when the QA effort has been fully successful and all errors have been detected, sent back for correction and should be finished because

some errors may slip undetected through the QA process. This is because errors may be active and passive in nature.

Demonstrated in the production sub-system, motivation loss, communication and training overheads influence the potential error detection rate. When staff are demotivated, and suffer communication losses, more staff needed are to locate to errors than would be under normal circumstances. If management at an early stage does not correct the problem, errors slip through the QA effort only to detect them at the late stages of production. (S.C., Davar & M., Parti, 2013)

The number of staff required to detect an error may be determined by the error type, (Not included in the model) work efficiency and the error density. A decrease in the number of staff allocated for QA leads to decrease in detectable errors. The potential error detection rate is often lower than the actual one. As a result, some errors may go undetected at the QA stage and end-up undetected at the next stage of software production cycle or in the final product itself as active or passive errors.

The error detection rate has a significant effect on the number of errors detected. The higher the error detection rate, the higher the number of detected errors. The error detection rate is a QA & REPI function. The error detection rate is subject to the number of QA staff allocated to detect errors. The higher the number of staff allocated to QA the higher the rate. Though this is a significant factor, over-allocation of staff to QA may result to reduction in production staff that may lead to increased production tasks per staff, initiates fatigue, loss of staff motivation and increase in the number of detected errors. (Williams, D. 2003a, 2003b), (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

5.6 Error Generation and Error Generation Rate

Several factors influence the error generation rate in the software development phase. A specified estimation may represent the effect of varied organizations and projects. (Kamuni, S.K., 2015) However, schedule pressure and a workforce mix, play a major dynamic role during the software development phase.

Workforce fall in two major groups namely: newly hired and experienced workforce. Newly hired staff deemed are less productive and error prone than experienced staff. Where the workforce consists of more experienced staff, the error rates are almost at their nominal rate. Experienced staff avoid making errors. New staff assumed are twice as accident prone than experienced ones. A mix of the two blends of

staff therefore influences the errors generated. If majority of staff are new the error generation rate increases. (Zawedde, A. & Williams, D., 2014), (Zawedde, A., 2016) and (S.C. Davar & M., Parti, 2013)

A number of other factors influence the error generation rate. These include organizational factors such as structured techniques, quality of staff and those of a particular project with variations from project to project and remain invariant during a single project. The cumulative effects of such factors captured are as real numbers of errors committed per task and computed as a product of the software development rate and the number of errors committed per task. The error type generated varies over time depending on the level of system development in the SDLC cycle. (Williams, D., 2003a, 2003b), (Zawedde, A.S.A. et al., 2011), (Zawedde A. &Williams, D., 2013, 2014) and (Zawedde, A., 2016)

Schedule pressure bears a hand in errors generation. When schedule pressure is on the increase (+ve), staff concentrate on their effort and put in more hours. However, exhaustion and stress increase the error rate. When the production rate is higher, tasks tackled per hour consequently increase. This may generate more errors per hour. (Williams, D., 2003a, 2003b), (Zawedde A.S.A., et al., 2011), (Zawedde, A. & Williams D., 2013, 2014), (Zawedde, A., 2016), (Van Oorehot K., Langerak, F. & Ngupta, K.S., 2011), (Putnam-Majarian, T. & Putman, D., 2015) and (Trecentis, 2018)

5.7 Software Error Densities

Error densities are because of the error detection during various stages of software development. Increased error densities affect the allocation of staff to rework on errors. High error densities relate to the quality of staff, their overall productivity, the injection of the quality assurance and testing functions. Lack or late injection of QA and testing results into an increase in the error density. From earlier discussions, a high introduction of new staff results into an increase in the error density. (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A., 2016)

5.8 Workforce Productivity, Rework Rate and Workforce Needed

When the potential productivity is achieved staff, use resources to meet the tasks assigned. The peak productivity can only be achieved when a staff or team of staff work at their peak efficiency. When variables or factors such as loss of motivation and existence of faulty processes, the production rate cannot achieve its full potential. Normal productivity measured by the amount of tasks they perform per day and tied to a constant or fixed factor varies from project to project depending on the number of tasks

involved. (S.C., Davar & M., Parti, 2013), (Putnam-Majarian, T. & Putman, D. 2015) and (Solomon, B., Shahibuddin, S., & Ghai, 2009)

5.9 Effects of Staff Experience & Learning on the Overall Productivity

The model simulation graphs demonstrates that distinguishing productivity exist between the newly hired and experienced workforce. Experienced staff carries a higher productivity and a larger output potential than the later. The implications therefore are that the ratio between new and experienced staff allocation to software production influences the potential productivity rate. Staff learning though with delay, increases the potential staff productivity rate since the new staff learns to handle their assigned tasks more efficiently. The learning curve therefore demonstrates that learning staff over time increase their productivity since they become more familiar with the tasks required and improve their problem solving skills. (Williams, D. 2003a, 2003b), (Zawedde, A. and Williams, D., 2013, 2014), (S.C., Davar and M., Parti, 2013) and (Putnam-Majarian, T., Putman D., 2015)

Actual productivity on the other hand is the level of staff or team productivity attained when they work at their optimal rate given the best possible use of their inherent resources. However, several variables such as multiple losses due to either communication or motivation influence the realization of the individual or team's full potential productivity level. If eliminated, influencing variables that create potential gaps in real environment situations and ideal situations are achieved then potential and actual productivity would be equal. (S.C., Davar & M., Parti, 2013)

5.10 Effects of Communication on Staff Allocations and Production

Communication influences software production since individual staff and teams must communicate to each other in order to progress well. Communication by nature in the project is an overhead. Time spent in communication between individuals and teams lead to a decrease in the production rate. This results to an overall drop in the team member's nominative productivity.

Workforce per average error in a management function determines the number of errors that each team member handles per day. Wrong decisions on the average number of errors staff can handle per day lead to overworking, loss of motivation, increase in the number of errors, experienced staff quit rate, hiring rate that have a ripple effect on staff productivity and schedule. (Williams, D. 2003a, 2003b), (Zawedde, A. & Williams, D., 2013, 2014), (S.C. Davar & M., Parti, 2013), (Putnam-Majarian, T. & Putman, D. 2015) and (Zawedde A. (2016)

5.11 Software Bug fixing.

Error fixing is a REPI and development function. After the QA, and testing, teams identified errors are direct to production team for rework. Early error detection and fixing have significant effect on production team productivity since the team have more time to concentrate on development other than bug fixing. (Zawedde, & Williams, D., 2013, 2014), (Kabaale, E. Mayoka, K.G. & Mbarika, I., 2014), (Putnam-Majarian, T., Putman, D., 2015)

5.12 Staff Motivation

Staff motivation link to personal feelings and observations and the goals they intend to achieve during the software production processes. (Williams D., 2003a & 2003b), (Zawedde, A. & Williams, D., 2013, 2014), (Zawedde, et al. (2011), (Mohapatra, S. & Gupta, K., 2011), (Morrison B.J., 2012) & (Zawedde, A., 2016)

Motivational factors such as salary, responsibility, and self-realization among others may be characteristic in overall organizations climate. The factors may remain constant or change over time during software production process. Slack time or breaks are an example of motivational mechanism that changes over time in accordance with the schedule pressures. Time lost to non-project activities such as tea breaks, telephone, email reading as well as time used for personal activities. Quantification of slack time is losses in terms of person-hours. To achieve actual production the slack influence staff not, pushing it away from the potential productivity. When productivity and falls behind schedule and not achieved, the workforce, subjected to increased schedule pressure. Because of the increased pressure, slack time reduce and concentration now shifts to tasks ahead. The workforce also tends to increase their person-hours in order to recover the lost time within the schedule. The outcome for the reaction which is for a limited period, the actual productivity increase as staff work harder and overtime hours to close the lost time gap. At that particular reaction time, actual productivity may exceed the potential productivity rate as this excluded overtime person-hours.

In limited time, positive effects realized in productivity is adversely affected as the above normal reaction results into strain to the workers with the increased schedule pressure, overall since there is a limit to overwork and exhaustion levels, the two variables limit the expected overall boost in productivity. If there is a boost the overtime and increased labor intensity that fall within the limits, the

boost is possible to bring the project back to the schedule time. (Van Oorchat K. Langerak F. Ngupta, K.S., (2011)

5.13 Effects of Workforce Exhaustion on Software Development

Workforce exhaustion due to schedule pressure and workload overtime have negative effect on staff productivity, efficiency and overall REPI efforts. When the overtime and exhaustion levels are beyond their limits, schedule time is re-adjusted. Due to exhaustion's realized during the period of increased schedule pressure, a slack time is necessary for staff to regain and stabilize at normal pressure levels. The slack work has a psychological healing factor hence staffs allowed breaks to remove tension of the workload and enjoy themselves. Due to increased workload, pressure and lack of slack time, worker's willingness to tolerate further levels of increased work pressure lowers. This scenario tends to pose a challenge to management as a misguided decision to these factors may eventually lead to an increased quit rate and eventual loss of experienced staff. When staff is unwilling to overwork, putting extra man-hours will have no effect and will be equal to zero.

In instances where opposite phenomenon occurs and the project is ahead of schedule and creates a negative schedule pressure, there exists an equal threshold for under-work as for the overwork. In such a situation, staff will tend to increase their slack time in order to fill their days. However, if this situation persists for long, staff work overtime, loose motivation, and initiate a demand for the reduction of the schedule with intention to create positive pressure. (Williams, D., 2003a & 2003b), (Pruyt E., 2010, 2013), (Paranali, K. Dhananjay, S., 2014), (Zawedde, A. & Williams, D., 2013, 2014)

5.14 Effects of Staff Experience on Productivity

The project workforce capacity has weighty effect on productivity, errors generated, rework and testing rate. Experienced staff better detects errors, generate less errors, handle more tasks at a time, work under pressure as well and effectively and efficiently rework on detected errors. If a project has high number of new staff, productivity decrease, and schedule pressure may creep in. (S.C., Davar and M., Parti, 2013)

5.15 Determining of Project Staff Levels

A number of factors influence determination of the projects total workforce level. One of the major and key factors is the current schedule completion date. (Van Oorchot, K., Langerak, F. and Ngupta, K.S., 2011) In planning phase, management determines staff level to complete the project within schedule

time based on the perceived remaining tasks, evaluate and defines the workforce stability. Before new hires, management determines and estimate the period when the new staff are required. It costs the project additional time and resources to hire a new staff because of overheads for training and assimilation before operations resumes normal peak efficiency. From the human resource management point of view, workforce level needed does not automatically translate into the hiring goal.

The projects ability to absorb new staff is pegged on the rate at which the hired workforce is assimilated and the training resource that is offered by the experienced workforce that are now assigned to training new recruits. A ceiling on new hires ("Max New Hire") as expressed in the model is equal to "Experienced Workforce" multiplied by the number of "New Workforce". A full time staff is expected to train effectively, i.e. (New Workforce*Experienced Workforce). Management has to determine the effectiveness of new hires to the project versus the loss in efficiency of their experienced workforce plus loss of workforce stability. These factors affect management decision on the determination of "Sort Workforce", Schedule completion ("ScheduledCompletionDate", work force stability and training needs. The "Sort Workforce" and the "TotalWorkForceNumber" then expresses the desired and actual workforce level. If the gap between the two, member will be hired or fired so that to equalize the difference. (Williams, D., 2003a, 2003b), (Pruyt, E., 2010, 2013), (Paranali, K. Dhananjay, S., 2014), (Zawedde, A. & Williams, D., 2013, 2014)

5.16 Effects of Turn-Over/Quit-Rate & Schedule on Projects

Staff turnover and expresses itself as the quit-rate of the experienced workers before project completion time. There are fluctuations in the over-all quit rate and replacement must be sort through hire. The new hires require training and assimilation into the project and if poorly managed, this may result into a significant delay. (William, D., 2003a & 2003b), (D., Bator, 2014), (Zawedde A. and D., Williams, 2013, 2014), (Zawedde, A. (2016)

5.17 Product Quality Assurance, Continuous Testing and Error Rework

Workforce productivity at all stages of software development determine software quality assurance (**QA**) and REPI success. When production team are efficient and effective, fewer errors generated are, errors await reworking decreased, while product-testing process is more effective. Increased levels of team's communication and effectiveness management decisions on resources allocations boost product quality assurance. REPI and quality assurance is effective where effective policy and feedback

mechanism are established and validation and quality analysis applied to all phases of software development. The analysis re-evaluates the effectiveness in the REPI process. The overall goal is effectively & efficiently carry out all requirements inspection. (Svahnberg, M.T., et al., 2010), (IEEE, 2014), (Zawedde A., 2016)

5.18 Cost of Errors in Projects

The RE pr0cess improvement and the quality assurance policy has significant impact on projects total cost. At low levels of QA, expenditure increase due to high cost injected into the testing phase as errors go undetected and slip through. Excessive REPI and QA expenditures result into increased cost. Increased errors demand that more staff be allocated to error detection and rework tasks, slowing down development, increased schedule pressure, loss of staff motivation, increased turn-over and increased demand for new hires that have implications of costs of training and assimilation. (Williams D. 2003a, 2003b), (Zawedde A.S.A., et al., 2011), (Zawedde A. and D. Williams, 2013, 2014), (Pruyt, E., 2010, 2013) & (Zawedde A., 2016)

5.2 Software Project Sub-systems and Sub-sectors

a) Planning Sub-System

Planning activity represents the initial stage for any software project. Before production and control launch, project size, scope, budgets are set discussed. Planning try to establish reasonable project completion date (man-days required estimates) and the initial workforce required. Management plays a key role on project development by determining the working hours per day, total workforce size required, and set projects completion date in advance. This implies that planning cut-across several other factors represented in human resource and project schedule planning in the planning subsystem. (IEEE, 2016, p. 1-73), (Zawedde, A., 2016, p. 109), and (IEEE, 2017).

Planning Subsystem/Sub-Sector Reinforcing Loop (R1): The initial step of any project-planning phase is to formulate and set a project completion schedule (Scheduled Completion Date). When the projects Scheduled Completion Date increases, Time Remaining increases, decreasing the Work Force Level Needed to complete project and the Sought workforce (Human resource management system) and decrease the Total workforce Level (Human resource management) hence a reducing new staff hire rate. Decrease in Total workforce Level required provides reduces the Cumulative Expected Man day (Workforce allocation) and increase in project Man-Day Remaining (Control) and effects the Scheduled

Completion Date. The loop terminates when an increase in Man-Day Remaining leads a need to increase the Scheduled Completion Date. (Lech, P. (2013) and Kamuni, S.K., (2015).

Planning Subsystem/Sub-Sector Balancing Loop (B1): The loop is concerned with the Man-Day remaining and workforce Level Needed. An increase in Indicated work-force, increases in the workforce Level Needed, Sought workforce and the Total workforce Level. An increase in the Total workforce level, decreases the Cumulative Expected Man day, Man-Day Remaining and the Indicated Workforce. The Man-Day Remaining closes the loop. . (Lech, P., (2013), and (Williams, D., 2003a, &2003b)

b) Controlling Sub-System

The controlling process in software development project encompasses three major elements. These include measurement of elements to be controlled, evaluating significant activities, variables controlled, and monitoring deviation of the system from the expected standards. The third element communication, reporting measurements and behavior of system development altered are to shift to expected standards. (Zawedde A. and D., Williams, 2013, 2014), (Zawedde, A., 2016)

The number of resources consumed, tasks completed or both measures project progress. Such measurements are used to calculate man-days still required to complete the project on time. In the process of this establishment, effects of production time, development quality and REPI and QA tasks to rework errors and system testing in comparison with the actual man-days remaining before project end time/deadline are observed.

The existence of remaining man-days in comparison to the actual man-days leads to conclusion that the project is behind schedule. This information is forwarded to management to enable them make decision to motivate their workforce urging staff to raise their energy to bring the project back on timetable. Other tactics may apply to extend the project schedule.

The key major challenge to management is how to measure progress for software management due to dynamic relationship between workforce size and software production. (Adbel-Hamid, 1991 pp. 117-119), (Pandey, D. & Ramani, 2009), (McLeod, Laurie, Stephene G. & MacDonnell, 2011), (Morrison B.J., 2012), (S.C., Davar and M., Parti, 2013) & (Zawedde, A., 2016)

Controlling Sub-System Reinforcing Loop (R1): The first reinforcing loop in the Controlling Sub-system focus on cumulative tasks developed, error detection man-days perceived needed and the schedule pressure.

An increase in the Cumulative Developed Tasks leads to a higher Percentage of Actual Work Done pushing the software development from design to coding phase. This increase Detected Errors for Reworking and Manpower Required to Rework on Detected Errors, increase Remaining Man-days leading, Schedule Pressure, and slack time prompting staff to boost their production effort which boost, software development. The loop closes when Cumulative Developed Tasks increase with an increase with Software Developed. (D. Bator, (2014), (Cuellar M., 2011), (McLeod, Laurie, Stephene, G. & MacDonnell, 2011), (Barbara Gladysz, et al., 2015) and (Zawedde, A. 2016)

Controlling Sub-System Reinforcing Loop (R2): The second reinforcing loop identify the relationship between developed and new discovered tasks. The loop tries to explain the problem where management makes underestimations in their decision. As the existing and new staffs improves their experience with improved productivity, error detection, resolving skills, and the Cumulative Developed Tasks increase, enabling management unveil more New Discovered Tasks. The New Discovered Tasks lead to New Tasks Thought Still Remaining with a time delay, increase in demand for Man-days Required Working on New Tasks, increased Remaining Man-days and schedule pressure. An increase in the Schedule Pressure leads to increased Software Developed. The loop closes when Cumulative Developed Tasks increase. (Mohapatra, S. and Gupta, K., 2012) (Williams, D., 2003a, 2003b), (McLeod, Laurie, Stephene G. & MacDonnell, 2011), (Zawedde, A., 2016)

Controlling Sub-System Reinforcing Loop (R3): The third reinforcing loop in the controlling subsector is concerned and linked to Cumulative Developed Tasks, Perceived Productivity, and Total Job Size (Man-days), Remaining Man-days and Schedule Pressure.

Like in the discuses above loop (B3), management translates an increase to Cumulative Developed Tasks to eventual increase in Cumulative Developed Tasks, Perceived Productivity. Increase in Perceived Productivity translated to a decrease in Man-days Increase Due to New Discovered Tasks. A reduction in a reduction in the workforce required to work on new discoveries (Man-days Increase Due to New Discovered Tasks) lead to a decrease in Total Job Size in Man-days. The drop causes a decrease in Remaining Man-days that increase the Schedule pressure as the project proceeds to its stop time. In

Response to reduced schedule pressure, staff requested are or forced to work hard to bring the project back to schedule. The effects of decrease in schedule pressure here is a boot to software development (but only for a short time), eventually Cumulative Developed Tasks levels increase and the loop close here.

Controlling Sub-System Reinforcing Loop (R4): The fourth reinforcing loop is concerned with the links between software to the total man-days needed for the newly discovered tasks, the total job size and the total workforce level.

When Cumulative Developed Tasks increases, Perceived Productivity increase as well leading to decrease in Man-days Increase Due to New Discovered Tasks. With reduced number of Man-days Increase Due to New Discovered Tasks, the Total Job Size in Man-days decrease as well which translates to a decrease in Remaining Man-days. A decrease in Remaining Man-days causes a decline in workforce Level required as the project move towards completion. With reduced workforce Level required, management decided to reduce the Total workforce Level that decrease the Cumulative Expected Man-days and negatively affects the Perceived Productivity. The loop closes. (Williams, D. 2003a, 2003b), (Putma-Majarian, T. & Putman, D., 2015), (S.C., Davar & M. Parti, 2013), (Zawedde, A.S.A. et al. 2011), (Zawedde, A., 2016)

Controlling Sub-System Balancing Loop (B1): The first balancing loop identifies the relationships between discovered tasks, increase in development man-days and the total job size (man-days). As Cumulative Developed Tasks increase, New Tasks Thought Still Remaining also increase as more tasks are unveiled and increase in Man-days Increase Due to New Discovered Tasks. As a control measure by management to reverse the trend of increased Man-days and Increase Due to New Discovered Tasks, that has the effect of decreasing remaining Man-days. A decrease in Remaining Man-days leads to a decrease in Schedule, Software Developed and subsequently a decline in the Cumulative Developed Tasks when the loop eventually closes. (Zawedde, A., 2016), D.W., Williams, (2003), (S. C., Davar and M., Parti, 2013)

Controlling Sub-System Balancing Loop (B2): The second balancing loop in the Controlling system focus on the relationship between Cumulative Developed Tasks, perceived tasks remaining, total person-days perceived still needed and schedule pressure. When Cumulative Developed Tasks

increases, New Tasks Thought Still Remaining decreases. With less New Tasks Thought Still Remaining, Man-days Required to Work on New Tasks decrease as well and results to a decrease in the perceived Remaining Man-days. This reduces the tension causing a decrease in the Schedule Pressure and Software Developed that eventually results into a decrease in the Cumulative Developed Tasks when the loop closes. (Williams, D., 2003a, 2003b), (Zawedde A. & Williams, D., 2013 & 2014), and (Morrison, B. J., 2012)

Controlling Sub-System Balancing Loop (B3): The third balancing loop concerns itself with the perceived software development productivity. This is an important variable for controlling. Linked to the Perceived Productivity are variables; Man-days Required to Work on New Tasks, Remaining Man-days, Schedule Pressure and the software developed and Cumulative Developed Tasks. (Zawedde, et al., 2013, 2014)

When Cumulative Developed Tasks increases, the Perceived Productivity increase as well. For this reason, the main concern link to the relationship between cumulative tasks and Productivity. As project move from design to coding phase in software development, the number of tasks accomplished increase fast. Increasing staff experience and on job, learning contributes to this increase. The experienced and well-versed staff can carry out tasks and coding fast and efficiently. This is the management view that when Cumulative Developed Tasks, Perceived Productivity increase as well (Abdel-Hamid, 1991: pp.117-120), (S.C. Davar and M. Parti, 2013) & (Morrison, B. J., 2012)

The perceptions that increase in Cumulative Developed Tasks lead to eventual increase in Perceived Productivity and decrease the Man-days Required to Work on New Tasks. As Man-days Required to Work on New Tasks decrease also decrease Remaining Man-days follow suit. A decrease in Remaining Man-days provides for a decreased Schedule Pressure provide for a decline in both the Schedule Pressure and Cumulative Developed Tasks where the loop is closed.

Controlling Sub-System balancing Loop (B4): The fourth balancing loop is linked with the cumulative man-days (Cumulative Expected Man-days) expended with the remaining man-days (Remaining Man-days) for the project. An increase in the Cumulative Expected Man-days leads to a decrease in the Remaining man-days. With this decrease management have to re-calculate the workforce levels required by reducing the workforce Level Required. A reduction in the workforce level required translates to a

decrease in Total workforce Level. A decrease in Total workforce Level eventually translates into a decrease in Cumulative Expected Man-days decreasing. The loop is closes at that point. (Zawedde, A., 2016), (Williams, D. 2003a, 2003b), (Putman–Majarian, T. & Putman, D., 2015), (S.C. Davar & M. Parti, 2013), (Zawedde, A.S.A. et al. (2011), (Zawedde A., 2016)

Controlling Sub-System Balancing Loop (B5): The last and final balancing loop in the controlling Sub-System links software development to the person-days perceived still required for testing and schedule pressure. The loop efforts in the testing phase control. An increase in Software Developed translates into an increase in Cumulative Tested Tasks when more tasks go to testing. An increase in the Cumulative Tested Tasks, decrease the volume of Tasks Remaining to be tested. This observed decrease lead to ManDaysStill Required for Testing reducing the Remaining man-days still required. A reduction in perceived Remaining man-days leads to decrease in Schedule Pressure. The loop close when slack times increase which results to less Software Developed. (Zawedde, A., & William, D., 2013, 2014)

c) Software Production Sub-System

Software production activities involve system design and coding, quality assurance (QA), error rework and system testing The subsystem has been broken down and mimic the base model by Abdel-Hamid, 1991, p.69) subsectors i.e. work force allocation, software development productivity, quality assurance and rework and the system testing. (Zawedde, et al., 2011), (Zawedde, A. & Williams D., 2013, 2014), (Zawedde, A., 2016)

Workforce allocation done is on to three different efforts in software development namely software development, quality assurance (QA), error rework and system product testing.

Management undertakes the role of planning and resources allocation as per the processes involved.

The development team initiates processes by designing and coding the software then send it to the Quality Assurance and Rework team. Quality assurance effort goes hand in hand with the production. The software is subject to quality testing methods and techniques such as code reading and testing, walkthroughs, reviews, inspection and integration testing. Rework team receive any identified errors are sent to the rework team. Staff allocation is dependent on the number of errors identified. (Williams D., 2003a & 2003b) (D.W., Williams, 2013) (Zawedde, A. & Williams, D., 2013, 2014) and (Zawedde, A.S.A. et al., 2011)

Testing is a quality check function to ascertain all tasks are fully developed. Testing team receive rework errors for scrutiny while some may be allocated back to the production team. At the onset of the project, as the development proceeds, some staffs shifted to testing and quality assurance teams. In the end, the entire team ends-up to in the testing section. The model represented in the chapter four demonstrate man-power allocation with feedback on their progress giving its present status report and man-power allocated.

d) Manpower Allocation Subsystem/Subsector

The work force allocation sub-sector in chapter 4 demonstrates how management allocates different resources between production, QA, Controlling and testing sub-sectors, (Zawedde, A. & D. Williams 2013, 2014) and (Zawedde, A., 2016)

Man-power Allocation Subsystem Balancing Loop (B1): The first balancing loop identified the central role of management in the project work force allocation. Manpower distribution is between production, rework and quality assurance with focus on processes undertaken and schedule pressure levels.

An increase staff allocating to QA, decrease production man-power leaving fewer workforce for production and testing resulting to a general reduction in software development productivity. When productivity decreases, percentage of the actual work done decreases (Control). An increase in required rework effort on the average error results to a decrease in the error rework in the QA assurance and rework subsectors. A decrease in the error rework causes an increase in the errors waiting for rework in the QA and Control sub-sectors. With an observed increase in the number of errors waiting rework, management has to undertake a decision (*Control*) to focus on the appropriated increase the numbers of person-days they think is still required to carry out the error rework duties on the already increased rework detected errors. An upward trend decision to increase work force is undertaken. (Williams D. 2003a & 2003b), (Morrison, B.J. 2012), (Zawedde, A. & D. Williams, 2013, 2014), (Zawedde, A., 2016)

An increase in the number of rework work force creases an increase in the schedule pressure (Control). The loop closes when work force allocation to QA decreases because of an increased schedule pressure.

In resource allocations allocates efforts to software development, QA and rework of the resources during the first phase of software development. As demonstrated in the figure above (simulation graph)

one of the major challenges faced with project management is to set the allocation ratios to utilize work force more efficiently. (Abdel-Hamid, 1991, pp. 69-75) and (Zawedde, A., 2016)

Man-power Allocation Subsystem Reinforcing Loop (R1): The first reinforcing loop in the work force allocation subsector demonstrates the relationship between QA manpower allocations the rework team allocation, rework effort and process and schedule pressure. As demonstrated in the simulation graph above, when more manpower is allocated to QA less manpower will be available for rework effort. Interchangeably, when less manpower is allocated to rework, more manpower will be available for development and testing. This trend follows the same trend as for the B1 effort. (Zawedde, A. & Williams, D., 2013 & 2014), (Zawedde, A., 2016)

Allocation of more workforce or development, results to high software development. The more software developed the higher percentage of actual work done. As the project shifts from design to coding, more workforce is required to rework when average error level decreases. In the design, errors are much more difficult to detect and rework on. The implications are that the average work force required to rework design errors is greater than that needed to rework code errors. Indicated in the simulation there is a negative relationship between the percentages of work done and work force required reworking on average errors. (Damian, D., & Chisan, J. 2006) and (Daneva, M., 2016)

Allocation of more workforce to for QA, percentage of work increases. When the Detected Error Rework Rate, Rework-Manpower per Average, Detected-Error-Reworking Rate increase. This has a negative effect on the number of the Perceived-workforce-man-day-Shortage and Total-workforce-Still-Needed. Reworked on errors decrease errors awaiting rework and the Workforce perceived still needed. The decrease in schedule pressure reacts to these changes. The loop closes when work force for quality assurance increase because of an increase to the schedule pressure. (S. C. Davar & M. Parti, 2013), (Tricentis, 2018), (Zawedde, A. & D. Williams, 2013, 2014) and (Zawedde, A., 2016)

Man-power Allocation Subsystem Balancing Loop (B2): The second balancing loop of the man-power allocation subsystem is related to the relationship between manpower for quality assurance, allocation of manpower for rework, manpower allocation for error detection and process and schedule pressure.

When management make a decision to increase manpower for quality assurance, man-power available for rework decrease accordingly. Rework effort work force is drawn from the production team hence

when production increase, rework allocation decrease hence a negative relationship between the two variables. A reduction in the rework work force lead an increase in the number (Williams, D., 2003a, 2003b), (Zawedde, A., 2016) of staff allocated to production and testing teams. An increase of work force allocated to production subsequently lead to an increase at the rate of software development. An increase in software development increases the percentage of actual work done which has an effect to the quality of product produced. The realized increase in the percentage of work done as a negative effect on the number of staff required to detect average error. However, the number of detected errors increases due to improved efficiency. An increase errors increases errors waiting reworking. The increase in the number of errors awaiting rework brings in need to increase the number of mandays to rework of the errors. This signals an increase in the number of perceived person-days still required for project completion. The loop close when an increase in schedule pressure causes a decrease in manpower allocated for quality assurance. (Zawedde, A. & D. Williams, 2013, 2014), (Zawedde, A., 2016) (Kamuni, S. K., 2015),

Man-power Allocation Subsystem/Subsector Reinforcing Loop (R2): The reinforcing loop have the same influence as the balancing loop B2. However, the link relates the relationship between work force allocation for quality assurance, software production, error detection and the schedule pressure.

Allocation of workforce to quality assurance effort implies that there will be less staff available for software production and testing. A low allocation of staff to production leads to a decrease in software development that have a direct effect on the percentage of work actually done. When the later happen the quality assurance work force required to detect the average errors increases. The effect also leads to a sharp decline on the amount of errors detected and the detected errors awaiting rework. A decline in the amounts of errors wait for rework results into a decline in the number of work force perceived still required for rework and the total mandays perceived still required. Because of these dynamic, schedule pressure reduces. The loop closes when more workforce allocated are to the quality assurance effort. (Zawedde, A. & Williams, D. 2014) & (Zawedde, A., 2016)

e) Software Development Productivity Sub-Sector

The subsector and subsystem shown in chapter 4 focus on the development phase. In this phase management have a duty to conceive and make predictions the rate at which software need to be

developed. Multiple factors affect the rate of software productivity alongside work force allocation. (IEEE, 2017), (Zawedde, A. and D. Williams, 2013, 2014) & (Zawedde, A. 2016)

The productivity subsector evaluates the project workforce overall productivity affected by factors such as workloads and overtime, motivational factors, well as schedule pressure. Workforce productivity may be sub-divided into two; potential and actual productivity. Potential productivity minus all faults due to faulty processes expressed are as actual staff productivity (Abdel-Hamid, 1991: pp.77-94). The faulty interrupting considerations include training and communication overheads, personal and team motivation and schedule pressure. (S.C. Davar and M. Parti, 2013), (Morrison, B.J., 2012), (S. C. Davar, M. Parti, 2013) & (Putnam-Majarian T. & Putman, D., 2015)

Software Development Productivity Sub-Sector Balancing Loop (B1): The loop considers the amount of software Developed-P, influences of remaining tasks, and the perceived Workforce Man-Day Shortage for efficient software production. Software developed influence the percentage of the total software development tasks done (Percentage of Actual Work Done-p). The percentage of actual work done determines and reduces the Perceived Work-Force Man-days Shortage (Control) as per the schedule pressure. Manday Absorbed reacts in a positive relationship with the Perceived WorkForce ManDay Shortage. If no shortage exists, no increase in the mandays needed to absorb the shortage. The tasks allocation to software development is dependent on the Perceived WorkForce manday Shortage. If no shortage exists, the project schedule is on course hence little or no adjustments are to the rate of development. If a shortage exists, to close the existing/possible gap, an upwards boost to the production per staff (Needed Boost for Software Development) is necessary. (Daneva, M., 2016), & (Gloria, P. et al., 2014)

The Actual manday on Project depends on how the workforce efficiently work and spend their time on the project. In real life, environment staff can never work to their maximum productivity and efficiency because some time which does not positively contribute to software production. Staff go for tea and lunch breaks or take to times perform their personal activities such as calling, check on emails or take offs. The slack time affects the overall staff productivity since, it calculated in lost man-hours and affects production man-hours. (IEEE, 2017), (Mohapatra, S. & Gupta, K., 2011) & (Gorschek, T. and David, A. M., 2007)

To cover for the lost time, management result into increasing pressure on staff by cutting the time slack to boost software development to desired levels. A cut on the workforce slack time and a subsequent increase in the man-hours tend to increase the development levels (Increased Productivity) to bring the project within the schedule frame. However, the increase is only for a short time before fatigues and loss of motivation sets in. (S.C. Davar & M. Parti, 2013), (Kabaale, E. Mayoka, K.G. and Mbariaka, I., 2014)

A positive relationship emerges between software development and the slack time. When development decreases slack time increase by decreasing the actual man-day spent on development (Actual man-day Fraction on Project) and vice versa. The loop (B1) closes when software development decreases and the percentages of actual work done (Percentage of Actual Work Done-P) respectively decrease. (Mijiwaart, R., 2012), (Lech, P., 2013), (Kamuni, S.K., 2015) & (Zawedde, A., 2016)

Software Development Productivity Subsector Balancing Loop (B2): The loop focus on the relationship between software development, development workforce allocation and schedule pressure. If the percentage of job done increase, due to increased software development, the Total Workforce Perceived Still Needed-P and schedule pressure (Control) and the workforce allocated are for production due to a reduction in required effort decrease. Less allocation of workforce to production eventually lead to a decrease in the software developed and the percentage of actual work done (Percentage of Actual Work Done-P) and the loop closes at that point. (Cuella, M., 2011), (Zawedde, A.S.A et al., 2011) Zawedde, A. (2016) and (Barbara Gladysz, et al., 2015)

Software Development Productivity Subsector Balancing Loop (B3): The loop as shown in the software development productivity sub-sector causal loop, concern itself with pressure at work and on actual man-day fraction on the project (Actual Man-day Fraction on Project).

When maximum, workable man-day shortage (Max Workable Man-day Shortage) decreases, the Man-day absorbed increase. The effects of slack time earlier discussed. If the project is running late, workers work hard to catch-up with the lost time and increasing the man-day absorbed to boosts the desired software development rate and fraction of man-days spent on work. Seen in my earlier discussion, this boost in software development productivity is short term after which it produces negative results thereafter. (Mohapatra, S. and Gupta, K., 2011) (Kamuni, S.K. 2015), (Putnam-Majarian, T. & Putman D., 2015)

Work pressure increase management as a response to rising schedule pressure and affects workforce total productivity (S.C. Davar and M. Patri, 2013), (Zawedde, A., 2016). Due to the existence of workload and exhaustion thresholds, that limits further boost in the Actual man-day Fraction on Project. The scenario is only possible when overtime increase labor intensities and the Max Workable man-day Shortage are within limits. Due to the limits, the Actual man-day Fraction on Project increase staff exhaustion levels. Exhaustion leads to negative effects on productivity. (S.C. Davar & M. Parti, (2013), Zawedde, A. et al., 2013, 2014) and (D. Bator, 2014)

Software Development Productivity Subsector Balancing Loop (B4): The loop shown in the software development productivity sub-sector, (See chapter 4) staff willingness to work overtime and the exhaustion levels, when the project is behind schedule, decision to increase the number of man-day Absorbed discussed earlier creates pressure that increase Staff Exhaustion Levels, decrease staff willingness to work overtime and decrease the Max Workable man-day Shortage. The loop is closed as the man-day Absorbed decrease. (Williams D. 2000, 2003a & 2003b) (Mohapatra, S. & Gupta, K. 2011), (Van Oorchat, K., langerak, F. & Ngupta, S.K., 2011) (Zawedde, A.S.A et al., 2011), (Morrison, B.J., 2012), (Zawedde, A. & Wiliams, D., 2013 & 2014) (Kamuni, S.K., 2015) and (Zawedde, A., 2016)

Quality Assurance and Rework Subsystem

Achievement of software quality fall in collaboration with the software production processes. As production proceeds, errors occur because the process involves human interaction bound to generate errors in the design, and coding or errors that generate from poor RE and REPI specifications. The finished software undergo testing to ascertain that functional and non-functional standards has been captured, achieved to meet the client and developer's satisfaction. Reinforcing (R1, R2 & R3) and balancing loops (B1, B2, B3 and B4) represents the Quality Assurance (QA) subsector.

QA and Rework Sector Balancing Loop (B1): The first balancing loop in this sector (See Chapter 4) concerns itself with software development progress, rework effort and process.

When software production increases, percentage of actual work done also increase. As project development, progress from system design to coding a reduction in Rework Manpower Needed per Average Error is evident. Increase in reworked errors (Rework) decreases errors waiting reworking (Detected Errors Waiting for Rework), schedule pressure, effort to carry out tasks and the daily man-day required for software development (Daily Manpower for Software Development). The loop terminates

when Daily Manpower for Software Development leads to less software developed. (Nurmulian, N., Zowghi, D., & Fowell, S., 2004)

QA and Re-work Sector Balancing Loop (B2): The second balancing loop is concerned with the allocation of workforce to the QA effort and subsequent effects on the daily manpower to the rework Process. (See Figure 4.23)

Expounded in the previous loops, increase development leads to increase in percentage of work done. As development moves from design to coding, the process provides passive errors more that are easily detectable, hence decreasing the workforce allocated quality assurance. Response to this decrease, there will be less Daily ManPower Allocated for Reworks still required which in turn lead to an increase to the resources available for the rework process. When Daily Man-Power Allocated for Rework is low, an increase observed is in the re-work duties. The increase rework duties results into decrease in Detected Errors Waiting for Rework, schedule pressure and the Daily Manpower for Software Development. The loop closed when Daily Manpower for Software Development decrease. (Zawedde A. & D. Williams, 2013, 2014) and (Zawedde, A., 2016)

QA and Rework Sector Balancing Loop (B3): The third balancing loop of the QA and rework system (Figure: 4.29) puts down the mechanism behind manpower allocation for QA and the potential to detect errors.

Discussed earlier in previous loops, an increase in software developed lead to an increase in the percentage of actual work done. As software development shifts from design toward coding phase, an increase in the Percentage of Actual Work Done decreases the Daily Manpower Allocated for QA translating into a decrease in the Potential Detectable Errors and Detected Errors Waiting for Rework, schedule pressure and Daily ManPower for Software Development as well. Less Daily ManPower for Software Development eventually translates into less Software Developed where the loop is closed. The loop demonstrates that lower QA effort eventually lead to increase in undetected errors. (Williams D., 2003a, 2003b) & (Zawedde, A. 2016) & (Zawedde, A.S.A. et al., 2011)

QA and Rework Sector Balancing Loop (B4): The fourth and final balancing loop in the quality assurance (QA) and rework subsystem demonstrates a relationship between the Percentage of Actual Work Done and Error Generation. When software developed increases, so is Percentage of Actual Work

Done. As system, development shifts from design to coding phase, Error Generation a decrease. Since coding tasks are easier to accomplish, avoid, and detect possible errors than in design phase, lower Error Generation leads to less Potential Detectable Errors. This in turn results into a decrease in Detected Errors Waiting for Rework and Schedule Pressure. A decrease in the Schedule Pressure leads to a decrease in Daily Manpower for Software Development and effort. (Williams D., (2003a, 2003b) Loop close as Daily Manpower for Software Development results to a reduction in Software Developed. (Sterman, C. D. 2003), (J. Sterman, 2000), (S.C. Davar & M. Parti, 2013), (Sterman J.D., Oliva R., Linderman, K. & Bendoly, E., 2015), (William D., 2003a, 2003b) and (Zawedde, A., (2016)

QA and Rework Sector Reinforcing Loop (R1): The first reinforcing loop is concerned with progress of software development, the QA effort and process. An increase in software development results into an increase the Percentage of Actual Work Done. As development move from the design to coding stage, the quality assurance work force is required to detect errors, decreasing the QA Manpower Needed for AveverageError Detection depending on the type of errors, staff experience level and assimilation delay. Easy errors to detect require a lower level of QA Manpower Needed for AveverageError Detection than would be with hard errors to detect. When amounts of errors waiting to be reworked (Detected Errors Waiting for Rework) increase, schedule pressure increase increasing the effort required to carry out tasks and Daily Manpower for Software Development. The loop close this lever the effects of Daily Manpower for Software Development results into an increase in software developed. (Morrison, B, J., 2012)

QA and Rework Sector Reinforcing Loop (R2): The second reinforcing loop in the Quality Assurance and Rework Sub-sector is concerned with software development progress; the errors rework effort and process as well as the overall work force allocation to rework activities. The rationale in the loop is that when software development increases, the overall Percentage of Actual Work Done. As the project progresses from design to coding, rework of coding errors becomes easier than design errors.

QA and Rework Sector Reinforcing Loop (R3): The third reinforcing loop in the quality assurance (QA) and rework subsystem efforts on the relationship between software production and error generation.

The loop reveals that an increase in software development lead to an increase in error generation. Potential Detectable Errors, Detected Errors Waiting for Rework and schedule pressure. Schedule pressure increase increases the Daily Manpower for Software Development. The loop closes as software development increases. (Williams D. 2003a, 2003b), (S.C., Davar & M. Parti, 2013), (Zawedde, A.S.A. et al., 2011) and (Zawedde, A. 2016)

f) The System Testing Sub-system

The system-testing subsector is the final sector of the software production cycle (figure: 4.8, 4.12). If errors escape the QA effort and go detected, they tend to remain in the software product until the testing phase. Earlier mentioned in the paper, errors may be passive or active in nature. The two type of errors influence the testing phase. The main effort of the testing team is to first if possible to capture the active errors and later search for the passive errors. The figure below demonstrates the testing subsector causal loop diagram (Solomon, B. Shahibuddin S. & Ghai, A., 2009), (Kabaale, E. Mayoka, K. G., & Mbarika, I., 2014). (IEEE, 2013, p.1-68), (IEEE, 2014), (IEEE, 2015, p. 1-149), (IEEE, 2017) and (Kamuni, S. K., 2015)

System Testing Sub-system Reinforcing Loop (R1): When the undetected errors arrive from the from QA, Active Error Density and rework effort increases. An increase in the Active Error Density leads to a subsequent increase in the Active Error Generation Rate, which leads to increase in Undetected Active Errors from QA and Rework process and closes the loop. (Williams, D. 2003a, and 2003b), (Kartika Rai, Lokesh Madan & Kislay Anand, 2014)

System Testing Sub-system Balancing Loop (B1): The loop concentrates on the process of active error detection. When the Undetected Active Errors from QA and Rework effort increases, likewise the Active Error Density increases. If the number of errors of Active Error Density increases, likelihood is that, the Detection of Active Errors increases accordingly. An increase in Detection of Active Errors eventually reduces the volumes of Undetected Active Errors from QA, Rework, and the loop close. (Joosen, D. Basten, D. & Mellis, W., 2011)

System Testing Sub-system Balancing Loop (B2): The second balancing loop concerns the detection of passive errors just like the loop **B1** than deal with active errors. The number of active errors increase sporadically for a while and become passive errors after a while generation of errors. (Joosen, D. Basten D., & Mellis, W., (2011), (Williams, D., (2003a, 2003b). Increase in the number of Undetected Passive

Errors from QA and Rework leads to an increase in Passive Error Density. An increase in Passive Error Density triggers the increase in Detection of Passive Errors. This loop closes when Detection of Passive Errors decreases the Undetected Passive Errors from QA and Rework process. (Jones, C. & Bonsignour, O., (2012)

System Testing Sub-system Reinforcing Loop (R3): The third reinforcing loop is concerned with the Passive Error Density and the System Testing process. When Undetected Passive Errors from QA and Rework are fending into the system and the Active Error Density retires the Passive Error Density increase as well. This increase in Passive Error Density leads to higher demand for the Testing WorkForce per Task that eventually slows down System Testing process because detected errors are fewer. This loop close when quality assurance and rework process increase due to a higher Passive Error Density and less system testing, is undertaking. (Zawedde, A.S.A. et al., 2011), (Zawedde, A., 2016), (Williams, D., 2003a, 2003b), (Zawedde A., & Williams, D., 2013 & 2014)

System Testing Sub-system Reinforcing Loop (R4): The last and final reinforcing loop (See Figure: 4.8, 4.12) is concerned with the detection of passive and Active errors in the subsector and system. (See figure 4.20, 4.28). When active errors go undetected to the testing phase from the Undetected Active Errors, QA and Rework rate of undetected errors increase. With a delay, active errors are retired and become passive instead increasing the Passive Error Density. Likewise, Testing WorkForce per Task increase leading to less System Testing and more active errors slipping into the system (Tricentis, (2018). As Detection of Active Errors increase, Undetected Active Errors From QA and Rework increase as the loop close. (Zawedde A., and Williams, D., (2013, 2014), (IEEE, 2013, 2014) & (Gloria, P. et al., (2014)

Human Resource Management Subsystem/Sub-Sector

This subsystem manages and controls the hiring and management of the total work force (S.C., Davar & M. Parti, 2013), (Zawedde A. & D., Williams, 2013, 2014), (Gloria, P. et al., 2014) and (Zawedde, A., 2016),

HRM Subsystem Balancing Loop (B1): Causal loop diagram focus on the project workforce hire. Management hires new people into the project to fill available vacancies. The hiring exercise increases the workforce. The newly hired staff are trained and assimilated into the project with delay due to trainings that the new staff under go before they are qualified to be considered for assimilation. After

training and gaining experience S.C., Davar and M. Parti, (2013), the new staffs gain experience to undertake tasks as experienced staff. Increase in cumulative expected man-days, decrease remaining man-days (Control). A decrease in man-days remaining call for need to boost the workforce through the workforce level required (Planning) with a call for an increased hiring. (Williams D., 2003a & 2003b), (D., Bator 2014), (Pruyt E., 2013), & (Putnam-Majarian, T. & Putman, D., 2015)

HRM Subsystem Reinforcing Loop (R1): The focus of this reinforcing loop is on the hiring capacity and the maximum number of new hires. As projects kickoff, hired workers undergo induction and training to gain experience. The experienced staff undertakes the new staff training and induction duties. As training proceed, new staff gain experience, and assimilated to the project until maximum limits hit. An increase in experienced staff lead to increase in maximum staff, ceiling number of new hires, and the maximum total number of sustainable workforce. This eventually results to an increase to the number of sort workforce by management to finish the project on time. As a result, management decides to hire more staff (Hiring). The loop ends at that point. (Williams, D., 2003a & 2003b), (S.C., Davar & M. Parti, 2013) and (Zawedde, A., 2016)

HRM Subsystem Reinforcing Loop (R2): The second reinforcing loops relate to staff hiring and the schedule considerations. From explanations of the earlier loop (R1), as hiring is undertaken, the amount of the number of man-days remaining decrease with increase in the workforce. This is a controlling function. The increase has an effect to the schedule completion date (Planning) and the project remaining time. Based on the amount of time remaining for the project and the workforce level required (Planning) to complete the project is calculated and determined as indicated in the simulation graph above. Management has to seek for workforce (Sort Workforce) through hiring (Adjustment of the hiring policy). (Svahnberg, M.T., Gorschek, R. Torkar S., Saleem, B. & Shafique, M.U., 2010), (Zawedde A., et al, 2011), (Gloria, P., 2014) & (Zawedde A. & Williams, D., 2013)

5.3 Field Discussion Groups Research Findings

5.3.1 Validity and Reliability Statistics

For validity tests IBM SPSS version 20 was used to establish how well the research instrument (Questionnaire) used measured to the intended concept, reliability tests was conducted to evaluate the consistencies and stability of the research instrument. (Pranjali, K. & Dhananjay, S. , 2014)

Cronbach's alpha (Cronbach, 1951) is a measure of reliability i.e. lower bound for true reliability of the survey defined as the proportion of variability in the responses to the survey that is the result of differences in the respondents. This mean that answers to a reliability survey differ because respondents have different opinions not because the survey id confusing or contains multiple interpretations. Comparison base on the number of items on the survey and the ration of average inter-item covariance to the average item variance.

Table 5.1 Case Processing Summary

		N	%
Cases	Valid	64	55.2
	Excluded	52	44.8
	Total	116	100.0

Table 5.1 indicated statistical analysis of case processing weighted by the variable Years Worked.

Table 5.2 Field Study Item Summary Statistics

	Mean	Minimum	Maximum	Range	Maximum / Minimum	Variance
Item Means	3.295	1.969	4.234	2.266	2.151	.348
Item Variances	1.075	.316	1.853	1.536	5.855	.180
Inter-Item Covariances	.119	-.632	1.214	1.846	-1.921	.111
Inter-Item Correlations	.084	-.656	.857	1.513	-1.307	.080

Table 5.3 Scale Statistics

Mean	Variance	Std. Deviation	N of Items
95.55	127.585	11.295	100

The tables 5.2 & 5.3 give a field study item summary statistics.

Table 5.4: Validity and Reliability Statistics

Cronbach's Alpha	**Content validity index**	**No. of Items**
.783	0.762	100

Reliability and validity results in table 5.4 Show that the field study questionnaire was both reliable and valid since both variables scored a Cronbach Alpha Coefficient and Content Validity index greater than 0.7 (Krishnaveni, R. & Deepa Ranganath , 2011) and (Williams, D. , 2003b)

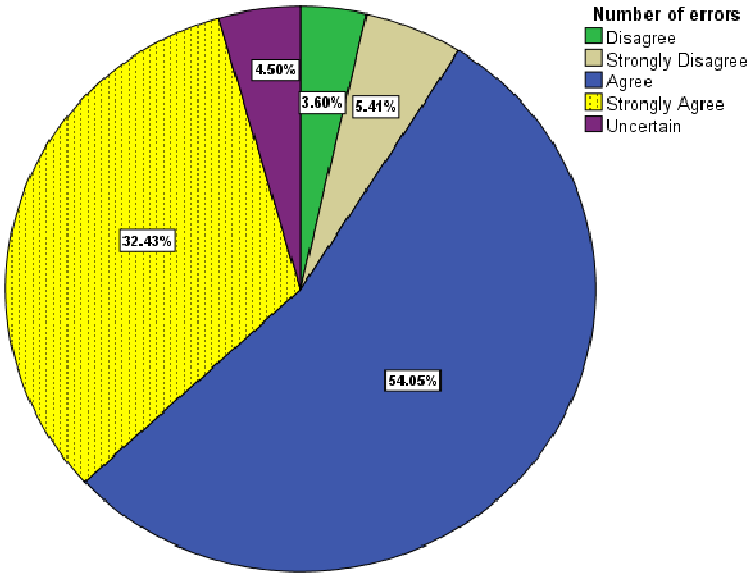

Figure 5.1 Number of Errors as a Major Cause of Software Failure

Figure 5.1 revealed findings that existing number or error are agreed as biggest challenge (54.05%) faced by the respondents, followed by 32.43% who strongly agreed, Only a small percentage of 9% , (5.4% - Strongly Disagreed plus 3.6% - Agree). A 4.5% was uncertain whether errors are the real cause of software and REPI failures. (Mohapatra, S. & Gupta, K., 2011), (Lech, P. ,2013), (Michael, M.J. & Shipman, F., M., 2000) and (Hastie, S., 2015)

Figure 5.2: Effects of Staff Productivity on REPI and Software Product Quality

Indicated in the figure 5.2, 45% of the respondents agreed and 8% strongly believed that, staff productivity affects final software product and the REPI process while less than 30% felt that productivity was not the root cause for software failures (Mohapatra, S. & Gupta, K., (2011). However though slightly above 10% were not sure whether staff productivity affected software product no of the respondents (0%) strongly agreed with the hypothesis. (Krisshnaveni, R. & Deeper Ranganath, 2011) and (Hastie, S., 2015)

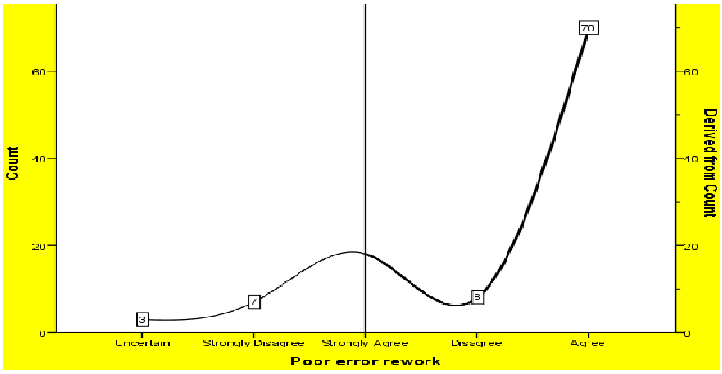

Figure 5.3: Poor Error-Rwork Cause Software Failures

Indicated in figure 5.3, most respondents agreed (70%) and strongly agreed (18%) that indeed poor error rework contribute to "software crisis" (Mohapatra, S. & Gupta, K., (2011) & (Kartik Rai, Lokesh Madan & Kislay Anand, 2014). However, 8% and 7% respectively felt they disagreed and strongly disagreed with the preposition that poor error rework were the root cause for failed software. (Kamuni, S. K., 2015), (Hastie, S., 2015) & (Putman-Majarian, T. & Putman, D., 2015)

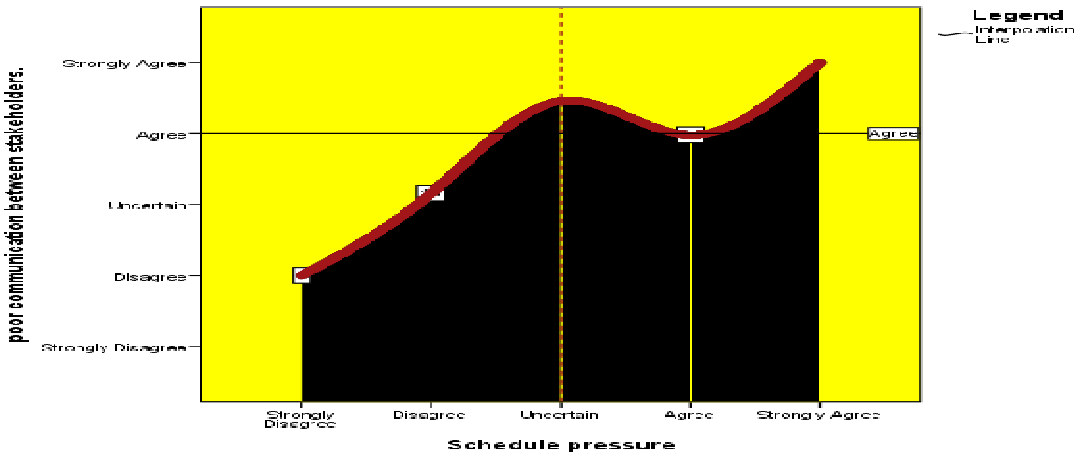

Figure 5.4 Effects of Poor Communication and Schedule Presure on Software Product

From the figure above , results indicates that poor communication between stakeholders and schedule pressure affects the REPI aprocess and software product delivery (Putman-Majarian, T. & Putman, D., (2015). However results indicate that stakeholders in big numbers do not understand or are not certain of the main cause of software crisis . This study therefore stands as an eye openner to stakeholders of the causeof software failuress. (Gloria, P. et al, 2014), (Hastie, S. 2015), (Annet, Reily, 2017) and (Yaniv Mordecai & Dov Dori, 2017)

5.4 Conclusion

The model was developed in consultation with stakeholders who included requirements engineers, process improvement experts (team leads), quality requirements managers, project managers and customers. Stakeholders presented with tabular and graphical model outputs, checked correctness and

insight generated by the model results. Model simulation results confirmed that a several factors from any of the six systems/Sectors listed below contributes to poor software quality (Zawedde, A., & Williams, D., 2013 & 2014), (Williams, D. 2003a, and 2003b), (Zawedde, A., 2016) and (Tricentis, 2018) namely:

a) Planning system
b) Human resource management
c) Software development
d) Software Production
e) Quality control
f) Testing
g) Controlling

Results from the discussion groups and simulations has indications that software crisis still exists and the causes highlighted in early research studies remain the same hence need to undertake and integrated approach to resolve RE, REPI and software development processes problems. A holistic dynamic model approach remain the hope to arrest software crisis. (Zawedde, A. 2016), (Yaniv Mordecai & Dov Dori, 2017)

5.5 Models Sub-Sectors and Associated Formulas and Calculations

The Unified Software Requirements Engineering Process Management (USREPM) model is a system made of sectors and sub-sectors demonstrated in chapter 4. The sectors are generated relationships as shown in the Sub-system/sub-sectors equations. (Williams, D., 2003a, 2003b), (Danian, D. & Chisan, J., 2006) and (Zawedde, A., 2016)

Controlling Sub-System/Sub-Sector

Cumulative__Tested_Tasks = +DT*Software__Developed
Mandays_Increase_Due__To_New_Discovered_Tasks =
+DT*New_Discovered_Tasks*+DT*Perceived__Productivity
Mandays_Required_To_Work_On_New_Tasks =
+DT*New_Tasks_Thought_Still_Remaining/+DT*Perceived__Productivity
ManDays_Still_Required_For_Testing = SUM (+DT*Tasks_Remaining__To_BeTested)
New_Discovered_Tasks = +DT*Percentage_Of__Actual_Work_Done-
+DT*Perceived_Percentage_of_Work_Done
New_Tasks_Thought_Still_Remaining = +DT*Cumulative__Developed_Tasks-
+DT*New_Discovered_Tasks
Perceived_Percentage_of_Work_Done = 100 %-(+DT*Cumulative__Developed_Tasks)
Perceived__Productivity = +DT*Cumulative__Developed_Tasks
Tasks_Remaining__To_BeTested = +dt*(Funished_Testing-+dt*Cumulative__Tested_Tasks)

Total_Job_Size___In_Mandays = +DT*Mandays_Increase_Due__To_New_Discovered_Tasks

Man-Power Allocation System/Sub-Sector

Daily_ManPower__Allocated_For_QA =
+DT*Total_Manpower__Still_Needed++DT*Schedule__Pressure*+DT*Manpower_Required_To_ReWork_On_Detected_Errors

DailyManPower_For_Dev_&_Testing = if 100 --DT*DailyManPower_For__Rework+-
DT*Daily_ManPower_For_Production then 1 else -DT*Daily_ManPower_For_Production--
DT*DailyManPower_For__Rework or DailyManPower_For__Rework--
DT*Daily_ManPower_For_Production

DailyManPower_For__Rework = +DT*Daily_ManPower_For_Production-
+DT*Daily_ManPower__Allocated_For_QA*+DT*Manpower_Required_To_ReWork_On_Detected_Errors/Detected__Errors_ReworkingRate

Daily_ManPower_For_Production = 100 %-(
+DT*Communication_&_Training_Overhead++DT*Daily_ManPower__Allocated_For_QA)

ManPower_Allocated_To_Software_Dev = +DT*DailyManPower_For_Dev_&_Testing-
+DT*SystemTesting*+DT*Schedule__Pressure

Project Human Resource Management Sub-System/Sub-Sector

Newly_Hired_Workforce (t) = Newly_Hired_Workforce (t - dt) + (HiringRate) * dtINIT

Newly_Hired_Workforce = +dt*HiringRate*+dt*Hiring__Delay

Inflows:

HiringRate = DELAY (1, +DT*Hiring__Delay, 1)

NewStaffAssimilationRate = +DT*Newly_Hired_Workforce*+DT*AveAssimilationDelay

Remaining__Mandays = (DT*Total_Job_Size___In_Mandays-+DT*Cumulative_Expected__Mandays)-
+DT*ManDays_Still_Required_For_Testing

ScheduledEndDate = +DT*Target__Completion__Date--
DT*Remaining__Mandays*+DT*Schedule__Pressure

Total__WorkForceNumber = +DT*Expected__WorkForce+ (+DT*ExperiencedWorkforce+
(+DT*Newly_Hired_Workforce))

AveAssimilationDelay = DELAY (1, 1, 0)

Cumulative_Expected__Mandays = SUM (+DT*Total__WorkForceNumber)

Expected__WorkForce = ENDVAL (Newly_Hired_Workforce, 0)

ExperiencedTransferRate = (+DT*NewStaffAssimilationRate)

ExperiencedWorkforce = +DT*Expected__WorkForce-
(+DT*Newly_Hired_Workforce*+DT*ExperiencedTransferRate)

FractionofExperiencedWorkForce = +DT*Total__WorkForceNumber-
+DT*Newly_Hired_Workforce/+DT*ExperiencedWorkforce

Hiring__Delay = TREND (1, 1)

MaxNewHire = FORCST (+DT*Expected__WorkForce, 1, 1, 0)

Max_Total_Sustainable__WorkForce = MAX (+DT*ExperiencedWorkforce++dt*MaxNewHire)

RemainingTime = -DT*STOPTIME-+DT*ScheduledEndDate

Sort_WorkForce = (+DT*Remaining__Mandays*+DT*Hiring__Delay)-
(+DT*HiringRate*+DT*WorkForce_Level_Required)

WorkForce_Level_Required = +DT*Max_Total_Sustainable__WorkForce-
+DT*Mandays_Required_To_Work_On_New_Tasks-+DT*Remaining__Mandays*-DT*RemainingTime

Software Project Management Sub-System/Sub-Sector

ErrorsDetectedForRework (t) = ErrorsDetectedForRework (t - dt) + (Error_Rate - ErrorRework) * dtINIT

ErrorsDetectedForRework = +DT*Active__Error_Density-
+dt*Detected_Errors_For_Reworking*+dt*Perceived__Productivity

Inflows:

Error_Rate = +DT*ErrorsDetectedForRework*+DT*+DT*Schedule__Pressure/+DT*Productivity

Outflows:

ErrorRework = +DT*(ErrorsDetectedForRework/+DT*Rework_Manpower_For_Average_Error)

People_&_Other__Resources (t) = People_&_Other__Resources (t - dt) + (Resource__Allocation -
Communication_&_Training_Overhead) * dtINIT People_&_Other__Resources =
Total_Job_Size___In_Mandays

Inflows:

Resource__Allocation = +DT*TurnOver*+DT*Target__Completion__Date

Outflows:

Communication_&_Training_Overhead = 2*+dt*(People_&_Other__Resources)

Schedule__Pressure (t) = Schedule__Pressure (t - dt) + (- TurnOver - Error_Rate) * dtINIT

Schedule__Pressure = if +dt*STOPTIME>+dt*Target__Completion__Date then 1 else IF
+dt*STOPTIME= +dt*Target__Completion__Date THEN 0 else 1

Outflows:

TurnOver = +DT*Schedule__Pressure

Error_Rate = +DT*ErrorsDetectedForRework*+DT*+DT*Schedule__Pressure/+DT*Productivity

Productivity = +DT*People_&_Other__Resources-
+DT*FractionofExperiencedWorkForce*+DT*Remaining__Mandays-
+DT*Communication_&_Training_Overhead-+DT*ErrorRework*-DT*+DT*Schedule__Pressure

ExperiencedStaffQuitRate = 2*+DT*TurnOver* (+dt*Schedule__Pressure-
+dt*Project_Progress__&__Status__Report)

Project_Progress__&__Status__Report = (+DT*Software__Developed)*+DT*ReportingDelay

ReportingDelay = +DT*DELAY (1, 2, 1)

ScheduleAdjustment = If (-DT*stoptime >+DT*Target__Completion__Date) then 1 ELSE
+DT*Detected_Errors_For_Reworking=0

Target__Completion__Date = ENDVAL (Project_Progress__&__Status__Report,
+DT*Project_Progress__&__Status__Report)

Software Development Productivity Sector

Actual_Manday_Franction_On_Project (t) = Actual_Manday_Franction_On_Project(t - dt) +
(Needed_Boost_For__Software_Development - WorkForce__Efficiency) * dtINIT

Actual_Manday_Franction_On_Project = dt*Total__WorkForceNumber/+dt*(ManpowerAllocation)

Inflows:

Needed_Boost_For__Software_Development = +DT*MandayAbsorbed

Outflows:

WorkForce__Efficiency =
+DT*ManPower_Allocated_To_Software_Dev*+DT*RateOfIncreaseinExhaustionLevel*+DT*Schedule
__Pressure

MandayAbsorbed = +DT*Max_Workable__Manday_Shortage
[1]/+DT*Perceived_WorkForce_Manday_Shortage

Max_Workable__Manday_Shortage[Exhaustion_&_willing_to_work] =
+dt*WorkForceOvertime_Threshold [Exhaustion_&_willing_to_work]

Perceived_WorkForce_Manday_Shortage = +DT*Schedule__Pressure/-
dt*TotalWorkForce__Perceived_Still_Needed

RateOfIncreaseinExhaustionLevel = +DT*Schedule__Pressure

Software_Dev_WorkForce_Productivity = +DT*WorkForce__Efficiency*-DT*Schedule__Pressure

Staff_Exhaustion_Level[Exhaustion_&_willing_to_work] = +dt*RateOfIncreaseinExhaustionLevel

TotalWorkForce__Perceived_Still_Needed = -DT*Software__Developed-
+DT*ManDays_Still_Required_For_Testing

Willingness_To_Work_Overtime [Exhaustion_&_willing_to_work] =
+DT*Max_Workable__Manday_Shortage
[1]*(+DT*RateOfIncreaseinExhaustionLevel*+DT*Staff_Exhaustion_Level [1])

WorkForceOvertime_Threshold[Exhaustion_&_willing_to_work] =
+dt*Staff_Exhaustion_Level[Exhaustion_&_willing_to_work]

Software Quality Assurance and Rework Sub-System/Sector

Rework_ManPower_Needed_Per_Average_Error (t) = Rework_ManPower_Needed_Per_Average_Error
(t - dt) + (Percentage_Of__Actual_Work_Done - DetectedErrors_Rework__Rate) * dtINIT

Rework_ManPower_Needed_Per_Average_Error = +dt*Active__Error_Density/+dt*ErrorRework

Inflows:

Percentage_Of__Actual_Work_Done = 100 %-(+DT*Cumulative__Developed_Tasks-
+DT*Software__Developed)

Outflows:

DetectedErrors_Rework__Rate = 1

Software__Developed (t) = Software__Developed (t - dt) + (ManpowerAllocation - QA_&__Rework - Percentage_Of__Actual_Work_Done) * dtINIT Software__Developed =
+dt*Detected_Errors_For_Reworking

Inflows:

ManpowerAllocation =
+DT*QA_&__Rework++DT*ManPower_Allocated_To_Software_Dev++DT*+DT*SystemTesting

Outflows:

QA_&__Rework = +DT*Software__Developed

Percentage_Of__Actual_Work_Done = 100 %-(+DT*Cumulative__Developed_Tasks-
+DT*Software__Developed)

Detected__Errors_ReworkingRate = +DT*Percentage_Of__Actual_Work_Done-
+DT*Potentially__Detectable__Errors/+DT*QA__Manpower_Needed_For_AveverageError_Detection

Manpower_Required_To_ReWork_On_Detected_Errors =
+DT*Detected_Errors_For_Reworking/+DT*Total_Manpower__Still_Needed

Cumulative__Developed_Tasks = SUM (+DT*Software__Developed)

Detected_Errors_For_Reworking = IF +DT*STOPTIME > +DT*STARTTIME THEN 1 ELSE 0

Error__Generation = +DT*Software__Developed-
+DT*Percentage_Of__Actual_Work_Done*+DT*Daily_ManPower_For_Production*+DT*QA__Manpower_Needed_For_AveverageError_Detection/+DT*Schedule__Pressure

Potentially__Detectable__Errors = +dt*Software__Developed-
(+dt*Daily_ManPower__Allocated_For_QA*+dt*ErrorRework*+dt*Manpower_Required_To_ReWork_On_Detected_Errors)*(+dt*QA__Manpower_Needed_For_AveverageError_Detection*+dt*Error__Generation)/+dt*Schedule__Pressure

QA__Manpower_Needed_For_AveverageError_Detection = +DT*MIN
(+DT*Percentage_Of__Actual_Work_Done)

Rework_Manpower_For_Average_Error = +DT*Percentage_Of__Actual_Work_Done

SystemTesting = 100%- (+DT*QA_&__Rework)

Total_Manpower__Still_Needed = REWORK (+DT*Percentage_Of__Actual_Work_Done)

System Testing- Sub-Sector

Active__Error_Density(t) = Active__Error_Density(t - dt) +
(Undetected__Active_Errors__From_QA_and_Rework) * dtINIT Active__Error_Density =
+dt*STOPTIME-STARTTIME *+dt*SystemTesting

Inflows:

Undetected__Active_Errors__From_QA_and_Rework = +DT*Active__Error_Density-
+DT*Detection_of__Active_Errors*+DT*+DT*Active_Error__Generation_Rate

Passive__Error_Density(t) = Passive__Error_Density(t - dt) +
(Undetected__Passive_Errors__From_QA_and_Rework) * dtINIT Passive__Error_Density =
+dt*SystemTesting-+dt*Active__Error_Density++dt*ErrorsDetectedForRework

Inflows:

Undetected__Passive_Errors__From_QA_and_Rework = +DT*Detection_of__Passive_Errors+DT*-DT*Undetected__Active_Errors__From_QA_and_Rework++DT*Passive__Error_Density

Active_Error__Generation_Rate = +DT*Active__Error_Density/+dt* Passive__Error_Density

Detection_of__Active_Errors = +DT*Active__Error_Density/+DT*SystemTesting

Detection_of__Passive_Errors = +DT*Passive__Error_Density/+DT*SystemTesting

Testing__WorkForce__Per_Task = +DT*SystemTesting-
+DT*Active__Error_Density+DT*Passive__Error_Density

All Errors =
+dt*Active__Error_Density++dt*Passive__Error_Density++dt*Detected__Errors_ReworkingRate*+dt+Undetected__Active_Errors__From_QA_and_Rework++dt*Potentially__Detectable__Errors*+dt*Undetected__Passive_Errors__From_QA_and_Rework

Funished_Testing = if (+DT*System Testing) = (All_Errors-
(+DT*Undetected__Passive_Errors__From_QA_and_Rework-
+DT*Undetected__Active_Errors__From_QA_and_Rework)) = 0 Then +DT*System Testing =
+DT*Software__Developed else 1

CHAPTER SIX: CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK

6 Introduction

The Chapter concludes the research strategy; summarizes the REPI research study with proposed policy improvements and recommendations. The section summarize the problem statement, literature review, field study and discussion groups, research data analysis, research design and model construction and simulation (Pruyt, E. 2010, 2013) and models simulation validations in relation to the problem statement and literature review.

The chapter form a discussion of the existing policy analysis and new policy formulation (Michael Mutingi, et al., 2017). The chapter finally offer recommendations areas, outlined as the focal point for future research.

The study dedicates to the answer and legacy questions of software quality and the application of SD to resolve SDLC problems. (IEEE, (2017) & (Pruyt, E. (2013). The study enhances our understanding of the RE, REPI dynamic nature and software development when holistically seen as an integrated system. The researcher develops an integrative SD model of software development, its structure, behavior and use it to study and demonstrate the dynamic implications of REPI process management.

The research however left out the exhaustive model behavior validation in real life environment due to time and resources constraints and the failure to find an ongoing project at the time to further the research. Model validation according to (Pranjali, K. & Dhananjay, S., 2014) remains the future research question and area of study. The model describe the dynamic behavior generated by the REPI, software development process interactions and project management including project the scope, requirements engineering, and staffing management, communication, cost and schedule. (Williams, D. 2003a, 2003b), (Jones, C. & Bonsignour, O., 2012) and (Zawedde, A, 2016)

The research contribute to the development of a new research tool that aid to understand and examine the impact of RE the development and REPI practices. (Pruyt, E., 2013) The model help investigate the key RE, REPI and software development processes: human resource management, change management, communication, planning, control and system testing), project monitoring and other practices necessary for successful REPI in software development. (Zawedde, A. & Williams, D. 2013, 2014), (Zawedde, et al. 2011), (Zawedde, A., 2016) and (Sterman, J. D., Oliva, R., Linderman, K., & Bendoly, E., 2015)

This study widely contributes to literature on RE improvement and software development, in reference to research work. The study establishes that RE, REPI and software development processes are dynamic than static practices that exists in isolation. The model in its SD thinking provides multiple building blocks to be used and applied in future research (Williams D., 2003a, 2003b), (Mwangi, H., Williams, D. Timothy, W. and Zipporah, N., 2015) & (Zawedde, A. , 2016). The results of from this study could offers significant interest to information systems organizations and clients, and stakeholders in software development. The study provides all stakeholders with a simulation environment to examine the impacts of their participation, practices and policies in software projects. (Michael Mutingi, et al., 2017).

The study exhibits important feedback structures in software development. The models simulation behavior demonstrates key feedback structures that determine key behaviors in software development. Intertwined interaction, between elements in software development, identified are, and clearly demonstrated in the model. The model therefore, considered could be a theory of RE, REPI and software development that can guide future research in areas of software development projects The model be if used by organizations could be used to study as well as carry out a holistic analysis on their design, specific practices and management policies on software projects. (Dahistedt, A. G., Natt, O. D., Ragnell, B. & Persson, A., 2007) and (Michael Mutingi, et al. (2017).

The model runs simulations demonstrates that management decisive changes do influence on the costs of reworking errors to the system (Hekimoglu, M. & Barlas, Y., 2010) and (Sterman, J.D., Oliva R., Linderman, K. & Bendoly, E. ,2015). The costs change sporadically and tend to upsurge over time (Putnam-Majarian, T. and Putman D., 2015). Observed behavior patter rearranging behavior as critical to success in software projects. The system behavior displays a pattern where projects with fixed resources budgets (Morrison, B.J., 2012), schedule and delays, re-factoring lead to higher project costs, poor production and staff motivation. Management decisions impacts on productivity, error rework, resource adjustments and unit testing. Uncoordinated resources allocation and policies decisions have negative impact on software projects. (Williams, D. 2003a, 2003b), (Glinz, M. & Fricker, S., 2013), (Jones, C. & Bonsignour, O., 2012) & (Zawedde, A, 2016)

6.1 Achievements of the Research Study

The model can be used by practitioners improve their software development practices because it facilitates the understanding of software project dynamics. The model can be used investigate impacts of

specific practices such as the cross team communications, software quality assurance and involvement of developers. According to Morrison, B. J., (2012), the research applied SD modeling to understand project progress when resources are limited staff poorly trained, and project planning, coordination, controlling and monitoring fail. In the process of model development, various limitations encountered during the research. One major limitation was unreliable data. (Hall, T. Beecham, S. & Rainer, A., 2002)

The model can have used as a guide to build, design and analyze software project management policies. The model can be used, examine the sensitivity of software projects to varied endogenous and exogenous factors such as team size, length and levels of interaction as well as requirements volatility. In summary, the study provides researchers a tool to study software projects, assist practitioners make tangible decisions and formulate appropriate software project and REPI process. (Yaniv Mordecai & Dov Dori, 2017)

6.2 Limitations of the Research Study

a) One of the key limitations of the model development was failure to accurately to predict when the staff were fatigued, individual developers capacities and levels of staff on job experience because there were no reliable data.

b) Varied data from multiple sources such as project managers, developers, quality control team and the testing team.

c) Some data existed using graphical relationships.

d) Limited resources/ research budget for effective data collection from a larger population.

e) Limited modeling tools: the only readily available modeling and simulation tools were limited to Vensim and Stellar that have their specific limitations. Obtaining licenses for globally available tools was not possible with the research budget available. Despite the listed limitations, the model analysis provides base guidelines for understanding and eliciting knowledge about REPI and software development problems.

6.3 Future Research

Software project failure is an old and widespread deadlock that hamper software projects year after year. The REPI model is my great contribution to improve software quality and in a nutshell the software development process. The model represents qualitative and quantitative data based from research carried out as highlighted in literature review and focused on the dynamics of the RE and REPI processes.

(Liuguo, S. Shijing Z. Jianbai, H, 2012), (Daneva, M., 2016), (Kamuni, S.K, 2015), (Lech, P., 2013), (Putman-Majarian, T., & Putman, D., 2015) and (Michael Mutingi, et al., 2017)

The model extended can be in scope and incorporate other factors such as tools used in capture user requirements, system development as well as consider firm sizes. The model, based on original Abdel-Hamid's model (Abdel-Hamid, T.K, 1991). Though not supported by standardized empirical and technical tests, there are opportunities for future research further improving my model. Future research work for could focus on the various types of testing that are undertaken during software development e.g. design and code testing as well as varied efforts required for each test type.

Future research must explore a broad ranges, boundaries, functions to achieve different model formulations and explore better ways to deal with the REPI and "software crisis" using a holistic dynamic approach, better analysis methods, data mining techniques, control techniques, current and formal modelling methods that would take a more multi-dimensional visualization.

However, the entire system constructed was to cover various sub systems/sectors behavior in totality. Future research could detail the analysis for each sub-system and sub-sector to unearth more policy hiccups and resolve the RE, REPI and Software gaps. (Putnam-Majarian, T. & D. Putman, 2015)

This research study did not consider critical factors that affect development such as: the domain, environment, software project size, industry size as well as the geographical locations. Future research should consider these factors. (Bjorner, D., 2006).

To obtain better software quality, research detailed ways to improve the current and future REPI, software development and project management. From the onset of software project, quality checks must be apply. The research recommends the adoption of the PCI Trilogy (Plan- Control and Improve) to project approach in resolve current existing RE problems. (Joseph Juran, & A., Blanton Godfrey, 2009)

However, this research scope did not cover the software implementation stage of software project. Future improved models should include sectors that simulate software implementation as part of improved structure. Conceptual diagram show key processes, the flow of information within causal loop and provides a broad and integrated view of the system by stakeholders when applied to the project related policies. The application of causal loop diagrams provides a broad view to understand the system with different that requires to be developed and improve the REPI management.

For successful software projects, all the core the sectors; human resource allocation, production, project management, quality assurance and rework control, planning, controlling and testing should be expanded and capture other factors that affect software quality in the dynamic world. (Lech, P., 2013), (Kamuni, S.K., 2015)

In future REPI and software projects, must embrace Dr. Juran's Trilogy of RE Quality Wave (Quality, Quality Control, Quality Assurance, Quality Management, Quality Management System and Total Quality Management). (Joseph Juran, 2009), (Cuellar, M., 2011) and (Sterman, J.D., Olivia, R., Lindeman, K, & Bendoly, E., 2015)

6.4 Advantages of Using the USREPM Model

The advantage of using the model as a tool of software project management is that system dynamics have widely been applied in the software projects planning in the developed world. SD as a methodology based on system feedbacks remains as an effective approach that holds the future to understand complex systems other than software development systems focused in the thesis (Williams, D. and Kennedy, M. , 2000), (Ferraira, et al., 2009) and (Mwangi, H. et al., 2017).

SD methodology and the model if well applied be used can to study complex software systems and assist elucidate knowledge on factors that greatly and contentiously affects software development projects.

(J. Starman, 2000), (Sterman, C. D., 2003), (Zawedde, A.S.A., et al., (2011, 2013, and 2014) & (Pruyt, E., 2010, 2013), (Firesmith, D. 2003, 2007), (Williams, D. 2003a, 2003b), (Berand, 2010), (Mijwaat, R., 2012) and (Zawedde, A., 2016)

References:

1. Abdel-Hamid, T. K., (1991). Software Project Dynamics; An integrated Approach, Prentice-Hall, Upper Saddle River, USA.
2. (Annet Reilly, (2017). New or Improved, Software Engineering Standards for Quality, ISO/IEC 25022 (2016), American Society for Quality. IEEE Computer Society
3. Barbara Gladysz, et al., (2015). Factors Influencing Keeping with Project Budget in IT projects, Zeszwty Maulebone, Uniwersytetu Scczecinskiego nr 855 France Rynki Finansome, Ubezpiecsenia" nr 74, t1 Wydawnictwo Naukowe Universytetu, Szczecon S. 511-529. www.wne12.ph/Frfu
4. Barlas, Y., 1996: Formal Aspects of Model Validity and Validation in Systems Dynamics, Systems Dynamics Review, 12 (3): pp.183-210.
5. Berry, D. M., Czarneeki, K., Antkiewiez, M., & Abdelrazik, M., (2010). Requirements Determination is Unstoppable Conference. IEEE Requirements Engineering (p. pp 311). IEEE Computer Society.
6. Bjarnason, E., Wnuk, K., & Regnell, B., (2011). Requirements are slipping through the Gaps. Sweden: Department of Computer Science.
7. Bjorner, D., (2006). Domains, Requirements, and Software Design. Berlin: Springer-verlag.
8. Cheng, B., & Atlee J., (2007). Research Direction in Requirements Engineering. Washington DC: Future of Software Engineering.
9. Chung, L., Yu, E., Mylopoulos, J., & Nixon, B., (2000). Non-Functional Requirements in Software Engineering. Boston: Kluwer Academic Publishers.
10. Cohene, T., & Easterbrook S., (2005). Contextual Analysis for Interview Design. Department of Computer Science (pp. 95-104). Toronto Canada: IEEE Publications.
11. Cooper, et al., (2009). Requirements Engineering Visualization: A survey of the State-of-Art. In: Fourth International Workshop on Requirements Engineering Visualization (rev'09)
12. Cuellar, M., (2011). Assessing project success: Moving beyond the triple constraint. International Research Workshop on IT Project Management 2010, paper 13.
13. D. Bator, (2014). Why Employees turnover hurts customer satisfaction. URL: Http://www.tembostatus.com/blog/how-employee-turnover-also-huts-customer-satisfaction

14. Dahistedt, A. G., Natt, O. D., Regnell, B., & Persson, A., (2007). Requirements Engineering Challenges in Market Driven Software Development. An interview Study with Practitioners. Information and Soft Technology 49, pp.588-604.
15. Damian, D., & Chisan, J., (2006). An Empirical Study of the Complex Relationship between Requirements Engineering Process and Other Processes that Lead to Payoffs in Productivity, Quality and Risk Management. IEEE Transactions on Software Engineering, 32 (7), 33-453.
16. Daneva, M., (2016). Requirements Engineering Foundation for Software Quality, 22nd International Working Conference (p. 256). Gothenburg: Springer.
17. Dolores, R. W., & Roger, U. F., (1989). Software Verification and Validation on Overview. Journal in IEEE software, 6 (3), 10-17.
18. Elbert, C., & Dumke R., (2012). Global Software and IT. New York: Springer.
19. Eveleens, L., & Verhoef, C., (2010). The Rise and fall of the Chaos Report Figures. IEEE software, 27 (1), 30-36.
20. Ferraira, et al., (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. Journal of Systems and Software 82(10) 1568-1577
21. Firesmith, D., (2007). Common Requirements problems, Their Negative Consequences and the Industry Best Practices to Help Solve Them. Journal of Object Technology, 6 (1), 17-33.
22. Firesmith, D., (2003). Specifying Good Requirements. Journal of Object Technology, 2 (4), 77-87.
23. Friker, S., & Glinz, M., (2010). Comparison of Requirements Hand Off, Analysis and Negotiation, 18th IEEE International Requirements Engineering Conference. Sydney.
24. Forester, J.W., (1968). Principles of Systems. Wright-Allen Press, Cambridge, MA.
25. Glinz, M., & Fricker, S. (2013). On shared Understanding in Software Engineering. Aachen, Germany: Software Engineering 2012.
26. Gloria, P., et al., (2014). Software Requirements Development: A path of Improving Software Quality. In Barafort B., O'Connor R.V. Poth A., Messnarcz R. (eds) Systems, Software and Service Process Improvement. EuroSPI, 2014. Communications in Computer and Information Science, vol 425. Springer, Berlin, Heidelberg.

27. Gorschek, T. & Davis, A. M., (2007). Requirements Engineering: In Search of the Dependent Variables. Information and Software Technology, 50 (1), 67-75.
28. Gorschek, T., & Wohlin, C., (2006). Requirements Abstraction Model. Requirement Engineering, 11 (1), 79-101.
29. Gotteddiener, E., (2001). Collaborate For Quality: Using Collaborative Workshop to Determine Requirements. Software Testing and Quality Engineering, 3 (2), 1-12.
30. Graham, I., (1991). Structured Prototyping for Requirements Specification in Expert Systems and Conventional I.T Project. IEEE, 2 (2), 82-89.
31. Hall, T., Beecham, S., & Rainer, A., (2002). Requirements Problem in Twelve Companies An Empirical Analysis. IEEE Proceedings Software, 149 (5), 153-160.
32. Hassenzahl, M. Beu, A. & Burmesster, M., (2001). Engineering Joy. IEEE Software, 18 (1), 70-76.
33. Hastie, S. (2015). Standish Group (2015). Chaos Report –Q&A with Jennifer Lynch. Retrieved September 19 , 2018,from htttps://www.infoq.com/articles/standish-chaos-2015
34. Hekimoglu, M. Barlas, Y., (2010). Sensitivity Analysis of Systems Dynamics Models by Behavior Pattern Measures. In: Proceedings of the 28th Systems Dynamics Conference (2010)
35. Hove, S.E., (2005). Experience from Conducting Semi-structured Interviews in Empirical Software Engineering Research. Software Metrics. IEEE International Symposium, 1 (1), 19-22.
36. IEEE, (2011). ISO/IEC/IEEE International Standard- Systems and Software Engineering – Life Cycle Processes- Requirements engineering. IEEE 29148-2011.
37. IEEE, (2013). Software and System engineering Software Testing part 2: Test Processes. ISO/IEC/IEEE 29119-2:2013 (E), p. 1-68.
38. IEEE, (2014). IEEE Standard for Software Quality Assurance Processes, IEEE Std 730-2014 (Revision of IEEE std. 730-2002), p 1-138
39. IEEE, (2016). IEEE Draft Standard for System Software and Hardware Verification and Validation-Corrigendum 1, 1012-2016/Cor 1-2017.
40. IEEE, (2016). ISO/IEC/IEEE International Standard for Systems and Software Engineering- Life Cycle Management- Part 4: Systems Engineering Planning.

41. IEEE, (2017). ISO/IEC/IEEE International Standard – Systems and software Engineering – Software life Cycles. IEEE 12207-2017.
42. IEEE, (2017). Systems and Software Engineering – Life Cycle- Management – Part-3: Guidelines for the Application of ISO/IEC/IEEE/ 12207 (Software Life Cycle Processes), P24748-3-2017.
43. IEEE, (2017). ISO/IEC/IEEE International Standard –Systems and Software Engineering- Life Cycle Management-Part 5: Software Development Planning. IEEE 24748-5-2017.
44. J. Starman, (2000): Business Dynamics: System Thinking and Modeling for a complex world. Irwin Professional Publications, New York, USA.
45. Jalote, P., (1997). An Integrated approach to Software engineering. Springer.
46. Jones, C. & Bonsignour, O., (2012). The Economics of Software Quality, Addison Wesley, USA.
47. Joosten, D., Basten, D. & Mellis, W., (2011). Measurement of Information System Project Success in Organizations -What researchers can learn from Practice' *ECIS 2011 Proceedings Paper* 177.
48. Joseph Juran, A. & Blanton Godfrey, (2009). Juran's Quality Handbook, McGraw-Hill International Editions: industrial Engineering Series, fifth Edition.
49. Juristo, J., Moreno A. M., & Silva A., (2002). Is the European Industry Moving Towards Requirement Engineering Problems? IEEE Software, 77-177.
50. K. Saced (1999). Defining a problem or constructing a reference mode. Technical report, Worcester polytechnic Institute.
51. Kabaale, E. Mayoka, K. G. & Mbarika, I., (2014). Requirements Engineering Process Improvement Challenges faced by Software SME's in Uganda. In: International Journal Conference of Computer Applications (0975-8887) Volume 88-No. 5 February 2014
52. Kamuni, S. K., (2015). Study of Factors that Induce Software Project Overrun Time, in Department of Mechanical and Manufacturing Engineering. St. Cloud State University.
53. Kartik Rai, Lokesh Madan & Kislay Anand, (2014). Software Crisis, International Journal of Innovative Research in Technology, Vol. 1 Issue 11.
54. Katonya, G., & Summerville, I., (1998). Requirements Engineering Process and Technique, UK: John Wiley & Sons.
55. Kotonya, & Somerville, I., (2006). Software Engineering. (9. Edition, Ed.), USA: Pearson.

56. Kraut, R. E., & Streeter, L. (1995). Coordination in Software Development in. Communication of the ACM, 38 (3), pp. 69-81.
57. Krishnaveni R. and Deepa Ranganath, (2011). Development and validation of an instrument for measuring the emotional intelligence of individuals in the work environment – In the Indian Context. The International journal of Educational and psychological assessment.
58. Langanath, M., & Duggan, J., (2012). A tool to Support Collaborative Software Requirements Management. *Requirements Engineering Journal,* 6 (3), pp.161-172.
59. Lech, P. (2013). Time, Budget and Functionality? I.T Project Success Criteria Revised. Information Systems Management, 30 (3), 263-275.
60. Liuguo, S. Shijing, Z. Jianbai, H., (2012) Pricing Simulation Platform Based on System Dynamics. Systems Engineering Procedia 5: 445-453.
61. McLeod, Laurie, Stephene, G. & MacDonnell, (2011). Factors that Affect Software Development Project Outcomes: A survey of research , ACM Computing Survey, Vol 43, No.4, Article 24
62. Michael Mutingi, et al., (2017). Systems Dynamic Approaches to Energy Policy Modelling and Simulation, *4th International Conference on Power and Energy Systems*, CPESE 2017, 25-29 September 2017, Berlin, Germany.
63. Michael, M. J. & Shipman, F. M., (2000). A Comparison of Questionnaire-Based and GUI-Based Requirements Gathering. IEEE Publication, (pp. 35-43).
64. Mijwaart, R., (2012). A Requirement Engineering Process Model for Software Development and Requirements Management, Foswiki Publishers, 1-13.
65. Mohapatra, S. and Gupta, K., (2011). Finding Factors Impacting Productivity in Software Development Project Using Structured Equation Modeling. *International Journal of Information Processing and Management 2* (1) (January 2011) 90-100
66. Morrison, B.J., (2012). Process Improvement Dynamics under Constrained Resources: Managing the Work Harder Versus Work Smarter Balance. System Dynamics Review, 28(4): 329-350, October-December.

67. Mwangi. H., Williams D., Timothy W., and Zipporah N., (2015). "Using System Dynamic to understand the role of cofactors Tb and malaria in the progression of HIV", International Journal of System Dynamics Applications, Vol. 16. 72-81, 2015. (P.3-33).
68. Nurmuliani, N., Zowghi, D., & Fowell, S., (2004). Analysis of Requirements Volatility during Software Development Life Cycle. 2004 Australian Software Engineering Conference, (p. P.28). Washington.
69. Pandey, D., Suman, U., & Ramani, A. K., (2010). An Effective Requirement Engineering Process Model for Software Development and Requirements Management. IEEE, 287-291.
70. Pandley, D., & Ramani, A. K., (2009). Social Organization Participation Difficulties in Requirement Pngineering Process A study. National Conference on Emerging Trends in Software Engineering and information Technology. Gwalior: Gwalior Engineering College.
71. Parviainen, P., Hulkko, H., Kaariainen, J., Takalo, J., & Tihinen, M., (2003). Requirements Engineering, Inventory of Technology. VTT Publication.
72. Philip, A., Laplate, (2017). Requirements Engineering for Software and Systems. 3rd Edition, Auerbatch Publications, New York. America.
73. Philip Morris International, (2015). Software Quality Assurance, https://www.pmi.com/resources/docs/default-source/legal/software-quality-assurance.pdf
74. Pohl, K., & Rupp, C., (2011). Requirements Engineering Fundamentals: A study Guide for the Certified Professional for Requirements Engineering Exam. Rocky Nook Computing.
75. Poloudi, A., (2004). Stakeholder Identification in Inter-Organizational Systems: Gaining Insight for Drug Use Management Systems. European Journal of Information Systems, 6, 1-14.
76. Pranjali, K., & Dhananjay, S., (2014). The Role of Verification and Validation in system Development Life cycle. *International Journal of Research in Advent Technology*, 2 (2), 1-3.
77. Pruyt, E., (2010). Using Small Models for Big Issues: Exploratory System Dynamics Modelling and Analysis for Insightful Crisis Management, *Proceedings of 18th International Conference of systems Dynamics Society, 25-29 July 2010*, Seoul, Korea.
78. Pruyt, E., (2013). Small systems Dynamic Model for big Issues: Triple Jump Towards real-world Complexity, Delft Library, Netherlands

79. Putnam-Majarian, T. and Putman, D., (2015). The Most Common Reasons Why Software Projects Fail. Facilitating the spread of knowledge and innovation in professional software development

80. S. C. Davar and M. Parti, (2013). Does training affect productivity of employees? Two methods of meta-analysis. *Indian Journal of Industrial relations*, 48 (4), 2013

81. Sabaliauskaite, G., Loconsole, A., Engstrom, E., Underkalmsteiner, M., Regnell, B., Runeson, P., et al. (2010). Challenges in Aligning Requirements Engineering and Verification in Large-Scale Industrial Context. Foundation for Software Quality, 128-142.

82. Solomon, B., Shahibuddin, S., & Ghai, A., (2009). Redefining the Requirements Engineering Process Improvement Model. Asia-Pacific Engineering Conference (pp. 87-89). Malaysia: Penang.

83. Sterman, C.D., (2003). Business Dynamics: Systems Thinking and Modeling for a Complex World, Irwin/McGraw-Hill. Chicago, USA.

84. Sterman, J. D., Oliva, R., Linderman, K., & Bendoly, E., (2015). System Dynamic Perspective and Modelling opportunities for research in Operations Management. Journal of Operations Management, 39-40 (November), 1-5.

85. Stevens, R., Brook, P., Jackson, K., & Arnold, S., (1998). Systems Engineering-Coping with Complexity. London: Prentice Hall.

86. Svahnberg, M. T., Gorschek, R., Torkar, S., Saleem, B. & Shafique, M. U., (2010). A Systematic Review on Strategic Release Planning Models, Information and Software Technology. Information and Software Technology, 52 (3), pp. 237-248.

87. Tricentis: (Accessed 23-9-2018). Real life Examples of Software Development failures, Software Testing 101, https://tricentis.com

88. Tveito, A., & Hasvold, P., (2002). Requirements in the Medical Domain. Experience and Prescriptions, pp. 66-69.

89. Van Oorchot, K. Langerak, F. and Ngupta, K.S., (2011). Escalation, De-escalation, or Reformation: Effective Interventions in Delayed NPD Projects, *Journal of Product Innovation Management* 28: 848.

90. Woodbridge, S., (2003). Bayesian Belief Networks. Technical report, CSIRO Center for Complex Systems Science.

91. Williams, D., 2003a). Integrating Systems Dynamics to Deliver Projects: faster, better and cheaper. In: Proceedings of 21st *International Conference of the Systems Dynamics Society*.
92. Williams, D., (2003b). Integrating Systems Dynamics Modeling and Case Study Research method: A Theoretical framework for process improvement. In Systems Dynamics Society.
93. Williams, D. & Kennedy, M., (2000) Towards a Model of Decision Making for Systems Requirements Engineering Process Management. In: The 18th International Conference of The Systems Dynamics Society, Bergen, Norway
94. Williams D.W., (2000). "Dynamic Synthesis: "A theoretical framework for research in requirements engineering process management," Operations Research Society, ISBN: 0903440202, 2000.
95. Wolstenholme, E. F., (2004). Using generic systems archetypes to support thinking and modelling, Systems Dynamic Review 20 (4): 341-356
96. Yaniv Mordecai & Dov Dori, (2017). Model-based requirements engineering: Architecting for systems with stakeholders in mind. , IEEE 2017, Viena, Austria.
97. Zawedde, A. (2016). Modelling the dynamics of requirements process improvement Eindhoven: Technische Universiteit Eindhoven.
98. Zawedde, A., & Williams, D., (2013). Dynamics of Software Systems projects during the requirements Process Improvement, In Press: *International journal of Simulation and Process Modeling*
99. Zawedde, A. & Williams, D., (2014). Dynamics of Software Systems Projects during the Requirements Process Improvement. *International journal of Simulation and Process Modelling*, Vol. IX (4)pp.206-221
100. Zawedde, A. & Williams, D., (2013). Determinants of Requirements Process Improvement Success. In: Proceedings of the 31st International Systems Dynamics Conference.
101. Zawedde, A.S.A., Klabbers M.D.M., Williams, D.D., van den Brand M.G.J.M., (2011). Understanding the Dynamics of Requirements Process Improvement: A New Approach. In : Calvano D., Oivo M. Baldassare M.T., Visaggio, G. (eds) Product-Focused Software Process Improvement, PROFES 2011, Lecture notes in Computer Science, vol. 6759, Springer, Berlin, Heidelberg.

Appendix 1: Research Questionnaire

a) Please indicate your gender

Male { }

Female { }

b) Age category in years

20 – 30 years { }

31– 40 years { }

41 – 50 years { }

Above 51 years { }

c) Kindly state the highest level of education attained

Secondary { }

Certificate { }

Diploma { }

Degree { }

Post Degree { }

d) Please indicate the number of years you have worked in the IT department

Between 1-5 { }

6 – 10 { }

Above 10 { }

e) Kindly indicate the level held

i. Top management { }

Middle Managerial { }

ii. Business systems manager { }

Support Staff (user) { }

Challenges Facing Alignment of Requirement Engineering Process

Code	Condition	Strongly Disagree	Disagree	Uncertain	Agree	Strongly Agree
RE	Software has been changed in the last 5 years					
	There are weakness in the system that needed to be changed					
RE	You were involved in system require engineering rectifying existing software problems					
RE	Users were involved in the system building process					
RE	You were involved in the verification and validation, of the systems?					
RE	There are significant changes needed					
RE	There was constant communication between Requirement engineers and the stakeholders.					
RE	Were you involved in the decision process, during the engineering of the new software?					

Requirement Engineering Management Process

Code	Condition	Strongly Disagree	Disagree	Uncertain	Agree	Strongly Agree
RE	The software was delivered within the agreed time frame					
RE	The system meet the capacity needed					
RE	The software performs as expected					
RE	Requirements planned on at the beginning of the engineering process were all included in the final software					
RE	Were user's needs sufficiently meet?					
RE	The company got its value for money					

Requirement Engineering Process

Code	Condition	Strongly Disagree	Disagree	Uncertain	Agree	Strongly Agree
RE	Were you involved in the choice of software Requirements?					
RE	Were you involved in the analysis of software requirements?					
RE	There was a discussion between you, software requirement engineer and the software developer					
RE	There were verification of all user needs in the system					
RE	You agreed changes and recommendations not system					

Appendix 3: Research Budget

NO	PARTICULARS	COST
1	Typing and photocopying research proposal	10,000.00
2	Typing and photocopying of questioners	10,000.00
3	Pretest	10,000.00
4	Traveling and subsistence	30,000.00
5	Typing and photocopying the final project	30,000.00
6	Final project binding	10,000.00
	Total cost	**100,000.00**

Appendix 2: Implementation Schedule

Activity	Description	Precedence	Duration
A	Pretest questioners		2 days
B	Receive feedback from the pretest	A	2 days
C	Upgrade the questioner from the feedback received	A	2 day
D	Administer the questioner to the respondents	C	2 days
E	Collect the questioners from the respondents	D	4 days
F	Data preparation	E	2 days
G	Data analysis using SPSS	F	4 days
H	Compiling the final project and presentation	G	14 days
Total days			**32 days**
Total number of weeks			**1month 3days**

Appendix 4: Implementation Schedule