

**PREDICTORS FOR DYSFUNCTIONAL SOFTWARE:
A GLOBAL SURVEY**

Anneleen Naidoo

2017

A Naidoo **PREDICTORS FOR DYSFUNCTIONAL SOFTWARE: A GLOBAL SURVEY**

2017



**PREDICTORS FOR DYSFUNCTIONAL SOFTWARE:
A GLOBAL SURVEY**

Anneleen Naidoo

Student Number: 8492

**Dissertation submitted in partial fulfilment of the requirements for the degree Masters of
Science in the Management of Technology and Innovation**

At

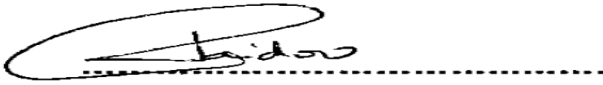
The Da Vinci Institute for Technology Management

Academic supervisor: Attilio Dalvit, MSc

2017

DECLARATION OF AUTHENTICITY

I declare that the research project, the lack of commitment to development timelines; inattention to delivering quality software and avoidance of accountability as predictors of a Dysfunctional software development team, is my own work and that each source of information used has been acknowledged by means of a complete reference. This dissertation has not been submitted before for any other research project, degree or examination at any university.



(Signature of student)

25/08/2017

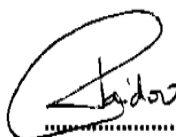
(Date)

Como, Perth, Australia

(City/town of student's residence)

DA VINCI COPYRIGHT INFORMATION

This dissertation may not be published either in part (in scholarly, scientific or technical journals), or as a whole (as a monograph), by the researcher or any other person unless permission has been obtained from The Da Vinci Institute. I agree that I have read and that I understand the copyright notice.



E. Sidov

ACKNOWLEDGEMENTS

Proverbs 16: 3 “commit to the LORD whatever you do, and he will establish your plans”. This accomplishment would not have been possible without God.

I must express my very profound gratitude to my son Dean Naidoo, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing my dissertation. Thank you for being forever interested, encouraging and always keen to know how I was proceeding.

A thank you to Da Vinci Institute for allowing me to complete my studies while based abroad. A very special gratitude goes out to Rinaka from Da Vinci Institute for her unfailing support and prompt assistance.

SUMMARY

In today's world of digital revolution, Information technology has become vital and an integral part of organisations. Organisations are increasingly depending on improving productivity through technology solutions leading to the role of the software development team becoming important. It has become increasingly important to understand which characteristics within a software development team significantly influence performance.

The purpose of this research was to analyse whether the following three characteristics: (1) the lack of commitment to development timelines; (2) the inattention to delivering quality software and (3) the avoidance of accountability, can be considered as predictors of a dysfunctional software development team. This quantitative theory study will be built upon a framework based on the findings process and introduce new literature for the apparent gap created in where no empirical attention or studies have been established in literature that exists in identifying the characteristics of a dysfunctional software development team.

The participants numbered 185, and were a combination of information technology managers, software architects, software developers and software testers, from large, medium and small sized organizations. The research is not limited to a specific geographical zone, and therefore the research findings will be relevant to a global audience.

The research demonstrated that a correlation exists between (1) the lacks of commitment to development timelines; (2) the inattention to delivering quality software, and (3) the avoidance of accountability and this negatively impacted the performance of a software development team. The research corroborated that these three characteristics are predictors of a dysfunctional software development team.

This research makes a contribution to this effort: the advancement of identifying three characteristics that, if present in a software development team, contribute to the team being dysfunctional.

LIST OF ACRONYMS

BPR	Business Process Reengineering
WWW	World Wide Web
NSF	National Science Foundation
DSRS	Division of Science Resource Statistics
QAT	Quality Assurance Team
E2E	End to End
GIF	Goodness of Fit Index
RD	Research Design
ICT	Information Communication Technology
ANOVA	Analysis of Variance
N	Total population
N	Sample size
R	Bias-relative bias
D	Population data
IT	Information Technology

TABLE OF CONTENTS

DECLARATION OF AUTHENTICITY	iv
DA VINCI COPYRIGHT INFORMATION.....	v
ACKNOWLEDGEMENTS.....	vii
SUMMARY	viii
LIST OF ACRONYMS	viii
LIST OF FIGURES	xii
LIST OF TABLES.....	xii
CHAPTER 1: OVERVIEW OF THE RESEARCH	1
1.1. Introduction	1
1.2. Problem statement	3
1.3. Purpose of study	5
1.4. Research objectives	7
1.4.1 <i>Theoretical objectives</i>	7
1.4.2 <i>Empirical objectives</i>	7
1.5. Justification and contribution of research	7
1.6. The scope	8
1.7. Limitations of the research	8
1.8. Ethical consideration.....	9
CHAPTER 2: LITERATURE REVIEW	10
2.1. Theoretical framework	10
2.2. Characteristics of dysfunctional teams.....	11
2.2.1 <i>The lack of commitment to development timelines</i>	11
2.2.2 <i>Inattention to delivering quality software</i>	13
2.2.3 <i>Avoidance of accountability</i>	15
2.3. Contemporary history of software development teams.....	17
2.4. Software development teams in organisations.....	18
2.5. Team development resources	18
2.6. Empirical framework.....	19
CHAPTER 3: METHODOLOGY FRAMEWORK	25
3.1. Introduction	25
3.2. Conceptual model.....	25
3.3. Hypothesis statement.....	26

3.4.	Ontological and epistemological approach	26
3.5.	Research methodology	27
3.6.	Research design	28
3.7.	Quantitative research	29
3.7.1	<i>Sampling design</i>	29
3.7.2	<i>Target population</i>	30
3.7.3	<i>Sample frame</i>	30
3.7.4	<i>Sample size</i>	30
3.7.5	<i>Sample method</i>	31
3.8.	Questionnaire design	31
3.8.1	<i>Measurement instruments of constructs</i>	31
3.8.2	<i>Lack of commitment to delivery timelines questions</i>	32
3.8.3	<i>Inattention to delivering software quality questions</i>	33
3.8.4	<i>Avoidance of accountability questions</i>	33
3.8.5	<i>Additional questions: combining lack of commitment to delivery timelines, inattention to delivering quality software, and inattention of accountability characteristics</i>	34
3.9.	Data collection	34
3.10.	Data analysis approach	34
3.11.	Reliability and validity of measurement scales	35
3.12.	Data analysis procedure	36
3.13.	Descriptive statistics	36
3.14.	Structural equation modelling approach	36
3.14.1	<i>Model fit (goodness of fit)</i>	37
3.14.2	<i>Path modelling</i>	37
3.14.3	<i>Confirmatory factor analysis</i>	37
CHAPTER 4:	DATA ANALYSIS & DISCUSSION OF RESULTS	38
4.1.	Introduction	38
4.2.	Response rate	39
4.3.	Active programs	40
4.4.	Medoid clustering	40
4.5.	The Medoid clustering program of number	40
4.6.	Discriminant analysis	42
4.7.	The design of simulation	44
4.8.	Findings	46
4.8.1	<i>Lack of commitment to delivery timelines</i>	49

4.8.2	<i>Inattention to delivering quality software</i>	50
4.8.3	<i>Lack of accountability</i>	50
CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS		52
5.1.	Conclusions	52
5.2.	Recommendations: Overcoming the three predictors of dysfunctional software development teams	54
5.2.1	<i>Inattention to delivering quality software</i>	54
5.2.2	<i>Lack of commitment to development timelines</i>	57
5.2.3	<i>Avoidance of accountability</i>	59
REFERENCES		62
APPENDICES		72

LIST OF FIGURES

Figure 3-1:	Conceptual model.....	40
Figure 5-1:	Results of Medoid clustering	41
Figure 5-2:	Algorithm construction.....	46
Figure 5-3:	Estimated R-Bias means against sample size	49

LIST OF TABLES

Table 5:1:	Respondent assignment to the clusters in a three cluster model	41
Table 5:2:	Tabular results of initial discriminant analysis	43
Table 5:3:	Item numbers of generated data in simulation	45
Table 5:4:	Analysis of variance results for the R-Bias and R-RMSE in the study.....	48

CHAPTER 1:

Overview of the Research

1.1. Introduction

The purpose of this quantitative theory study is to analyse three characteristics as predictors of a dysfunctional software development team; the lack of commitment to development timelines, the inattention to delivering quality software, and the avoidance of accountability.

The Merriam-Webster dictionary defines dysfunctional as “the condition of having poor and unhealthy behaviours and attitudes within a group of people”. At this point, it is also important to define the meaning of a team. Katzenbach and Smith (2005) define a team as “a set of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable.”

According to Attarzadeh (2011), in software development there exists a relationship between cost, quality and time. The task of conveying a cost competitive, quality software in today’s market is complex. In today’s dynamic world of technology, software development teams are playing a critical role in the development of various industries, fundamentally changing many sectors which have in certain instances led to the restructure of many businesses (Attarzadeh, 2011).

Technology has dramatically changed the lives of individuals and organisations. Bradley describes Information Technology as a broad concept that refers to breakthroughs in science which have allowed for a better or automated solution within organisations (as cited by James, 2017). The Merriam-Webster dictionary defines information technology as “the development, maintenance, and use of computer systems, software, and networks for the processing and distribution of data”. It goes without debate that information technology individuals are becoming a commodity, influencing organisations and providing industries with a competitive edge. Organisations possess a strategic advantage when they are able to leverage the collective talents and contributions of information technology employees (Mason, 2015). One can debate that there is a direct correlation between the extent of

performance of the organisation's information technology teams and the overall performance of the organisation. Effective and profitable teams develop a desire to be responsible and have a strong commitment to a conventional approach; which according to Jon, is how they will work together to realise their goal or to accomplish their purpose (as cited by Katzenbach & Smith, 2005).

Since the need for information technology teams has grown significantly, research and knowledge referring to information technology team performance has not been widely researched. The limitations of organisational resources, training, and consultants, all of which are required to assist organisations with their information technology capability, consequently leads to most organisations reporting that they struggle with maintaining effective software development teams. The spread of Internet technology and the proliferation of portable computing and communication devices have accelerated trends that began in the past two decades and now are viewed as "electronic commerce" or e-commerce.

Organisations now use the World Wide Web (www) and open source applications to communicate with the general public and also use similar, but more secure, intranets and extranets to contact and communicate with employees, suppliers, and distribution partners (National Science Foundation, Division of Science Resources Statistics, 2014).

However, the highly poignant question remains: Is there a correlation between ineffective teams within organisations in general and more technical teams such as software development teams? If the answer is no, are there characteristics unique to software development teams that could be used as predictors to prevent software development teams from becoming ineffective and dysfunctional? Conceivably there are many reasons which lead to software development teams failing to deliver as promised and as obligated by the stakeholders.

According to Jon, a team is a "set of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable" (as cited by Katzenbach & Smith, 2005). The researcher aims to examine how the following three characteristics, specifically: 1) the lack of commitment to development timelines; 2) the inattention to delivering quality software, and 3) the avoidance

of accountability, contribute to the team becoming a dysfunctional software development team and consequently and sadly condemned collectively as poor performers.

The main reason for the investigation is to present research that expounds on the analysis of three specific characteristics that contribute to software development teams becoming dysfunctional. Within the information technology software development environment, and consequently the financial industry, there exists the challenge, at some point in an information technology's leader's career, to deal with dysfunctional teams which debilitate the leader's energy and mental powers, and consequently result in a waste of the organisation's financial resources.

This current challenge is faced by many development software team leaders, and is also prevalent in the researchers' current place of employment. An easier question to answer is: What is the definition of a successful and high performing team? According to Jon, the definition of a successful and highly performing team is a group of highly motivated individuals coming together to bring maximum value for the organisation (as cited by Katzenbach & Smith, 1992).

This research was aimed at demonstrating an understanding of these three characteristics and to show how their existence leads to a dysfunctional software development team, and so contribute to the field of study which will allow others to learn from and extend the researcher's findings. Presently, to the researchers' knowledge, no literature exists on the characteristics of a dysfunctional software development team.

1.2. Problem statement

In today's technology driven and global arena, organisations are faced with significant pressure to perform and maintain the company's strategy as a leading innovator whilst simultaneously keeping the company competitive and profitable. These pressures, mostly financial and strategic, have resulted in software development teams missing committed timelines, and when pressure is at its peak, code is passed on within the software development life cycle which is of poor quality and rife with software development defects. (Bloch, *et al.*, 2012)

Research recognizes that there is an unparalleled adverse impact on organizations when a software development team is dysfunctional. (as cited by Cameron & Green, 2015). Since the need for information technology software development teams has grown significantly, the need for more research and knowledge referring to information technology software development team performance has become imperative.

With the upward trend on the reliance on technology, most organizations are not equipped with, or do not have access to, theories and approaches to identify and eliminate characteristics existing within dysfunctional software development teams, and therefore are missing the benefits of turning the team around to a high performance software development team. A current challenge which adversely affects both leaders and team members within software development teams (Bloch *et al.*, 2012), and is also prevalent in the researchers' previous and current place of employment.

The alarming issue noted by the researcher in the current organisation is the behaviour adopted by development teams; demonstrating a complete lack of accountability, generally justified by poor planning, lack of business requirements clarity, and environments and resource constraints. This trend continues with no signs of abating and potentially exposes the company to reputational risk and financial penalties when the organisation is found to be transgressing regulatory requirements of compliance.

A technology leader at some point in their career will be confronted with software development teams not performing at their optimum level. The behaviour of non-committal to timelines for the delivery of solutions requiring software development, specifically within a project, could potentially adversely impact the organisation's competitive positioning within its respective market. (Busseri & Palmer, 2000)

Another common issue that exists within development teams that the researcher noted in the previous organisation of employment, was that developed software was passed onto the testing team but was riddled with defects, begging the question of how developers are coding and handing over to testing teams if they are not unit testing their code. This type of

behaviour impacted the organisation whereby product launches were delayed so as to provide more time for the quality assurance team to test the end-to-end (E2E) solution and further provide approval as to when the code was ready for production.

This research fundamental goal is provide findings and literature that supports the following characteristics as predictors of a dysfunctional software development team: 1) the lack of commitment to development timelines, 2) the inattention to delivering quality software, and 3) the avoidance of accountability.

1.3. Purpose of study

The goal of this research is to research the influence of 1) the lack of commitment to development timelines, 2) inattention to delivering quality software, and 3) avoidance of accountability as predictors of a dysfunctional software development team.

Teamwork is fundamental to any organisation's ability to achieve its business goals. This research will provide insight into characteristics of a dysfunctional software development team, equipping organisations to set up strategies around resources and training to eliminate these characteristics within a software development team. The rewards of striving to create a functional, cohesive team is one of the few remaining competitive advantages available to any organisation looking for a sustainable yet powerful point of differentiation. Focusing on eradicating these characteristics will empower the team to become a functioning team making higher quality decisions and accomplish more in less time and with less distraction and frustration. (James, 2017).

According to Jeff (as cited by Urban, 2009), there is an unparalleled adverse impact on organisations when a software development team is dysfunctional. Most organisations are not equipped with, or do not have access to, theories and approaches to eliminate characteristics existing within dysfunctional software development teams, and never reap the benefits of turning the team around to a high performance software development team. Jeff states that having a dysfunctional team, at best limits a team's ability to optimally achieve

their goals, and at worst, it cripples a team from accomplishing anything (as cited by Urban, 2009).

Jeff states that software development teams are an important component of an organisation, because organisations claim many of their accolades of existing products in the market, as a result of their software development teams that design and implement the products (as cited by Urban, 2009). Information technology is considered one of the most important enablers of process change (Grover *et al.*, 1998). In one of the first articles about business process re-engineering, Davenport and Short (1990) argue that, together, processes and information technology can be seen as a new industrial engineering that may revolutionise the way in which organisations operate.

Bradley (as cited by James, 2017) states that a noticeable benefit of information technology within an organisation is increased productivity, leading to a lower cost structure. Organisations have acknowledged an increase in revenue, reduction in costs, and a significant improvement in speed, storage of information, the ease of which information can be shared, and a decrease in human error when there is a presence of high performing software development team (James, 2017).

It is becoming increasingly impossible in the current digital era to attain long term business success without leveraging the benefits of information technology. “The path of technological innovation in business means doing something different, smarter or better that will make a positive difference in terms of value, quality or productivity by using emerging or proved technologies of the world” (James, 2017).

Today’s organisations rely on their information technology teams to increase efficiency, productivity and effectiveness, and attain long-term business growth; thus successfully achieving the goals of the business (James, 2017). Research completed over the last two decades has shown that organisations investing in technology have increased their global footprint, market share, and overall market competitiveness, and have decreased the time taken to resolve complex problems and planning business scalability (Butt, 2015).

1.4. Research objectives

In this section the researcher critically examines the theoretical and empirical research objectives of this research.

1.4.1 Theoretical objectives

The following academic goals were developed:

- To review literature on the lack of commitment to development timelines,
- To review literature on inattention to delivering quality software, and
- To review literature on the avoidance of accountability.

1.4.2 Empirical objectives

The following practical objectives were developed:

- To investigate the influence of the lack of commitment to development timelines, and the correlation leading to a dysfunctional team,
- To examine the impact of the inattention to delivering quality software, resulting in a dysfunctional team,
- To explore the effect of avoidance of accountability, leading to a dysfunctional team.

1.5. Justification and contribution of research

To justify this research, it is evident that it has and will make a fundamental contribution to literature, whilst simultaneously pioneering the theory of successful teams within the discipline of information technology.

Teamwork is fundamental to any organisation's ability to achieve its business goals. This study will provide insight into the characteristics of a dysfunctional software development team, thus equipping organisations to set up strategies around resources and training to eliminate these characteristics within a software development team. Creating a functional, cohesive team is one of the few remaining competitive advantages available to any organisation yearning to achieve a long lasting, sustainable differentiation. Focusing on eradicating these characteristics will empower the team to become a functioning team,

making quality decisions, and accomplishing more in less time and with less distraction and frustration.

According to Abid, the role of the software development team has become critically important. In the current and future competitive and lean business environment, the quality of a software development impacts business failure rates (as cited by Butt, 2015).

1.6. The scope

The extent of the research is to bring out the factors that hinder service delivery, to show how accountability is important to software development group, and to explore the way inattention to delivering quality software leads to dysfunctional software development teams. Further, the research answers the question as to how a lack of commitment and trust affects the software development timelines.

The research is scheduled to take at least three months from its onset. The researcher will initiate the role of the literature review which will be in quantitative theory form. The studies are to provide a framework for the findings process and introduce new research for the apparent gap in the literature that exists when identifying the characteristics of a dysfunctional software development team. The quantitative study design is an excellent way of finalising results and proving or disproving a hypothesis.

After analysis was completed using statistical software the resulting answers achieved was discussed and the drawn conclusions published. Conducting numerical experiments will also reduce some of the outside factors because of good handling and design which results in real gain and non-biasedness (Martyn, 2015). The information will be collected and analysed using secondary data in a tabular way.

1.7. Limitations of the research

The research was undertaken with use of secondary data since it would be more expensive and uneconomical to collect primary data from the field. What limited the study further is the fact that the population under study was large, so receiving the stratum and reducing it to the required strata and correlating it with the actual study was tedious.

1.8. Ethical consideration

Due to the human element, measures were placed to ensure that the protection of the respondents' privacy. No fabrication, falsifying or misrepresenting of research data occurred. Also, no member were forced to take part in the study, and no incentives such as money were used to lure participants into taking part. Participants were free to withdraw from the research at any given stage. All the procedures as far as ethical conduct throughout the research were adhered to by the researcher. The researcher kept all collected data as confidential and did not use the data collected for any other purposes other than that applicable to this research.

CHAPTER 2:

Literature Review

2.1. Theoretical framework

The role of the literature review in quantitative theory studies is to provide a framework for the findings process and to introduce new literature for the apparent gap in the literature that exists in identifying the characteristics of a dysfunctional software development team. Quantitative research design is an excellent way of finalising results and proving or disproving a hypothesis.

After analysis with statistical software the resulting answers achieved will be discussed and the conclusions that are drawn published. Conducting numerical experiments will also reduce some of the external factors, if properly designed, and so the results gained can be seen as real and unbiased (Shuttleworth, 2015).

During and after the data analysis, the researcher will endeavour to provide a more comprehensive literature review, providing information and confirming the theory prevalent in the data. Quantitative data analysis is the preferred method. According to The Pell Institute (2015), it is helpful in the data examination; the reason being is that it provides measurable and straightforward outcomes. The data can be scrutinised in a variety of different ways (The Pell Institute, 2015).

The researcher will analyse the discovered data against the literature's theoretical framework. If the data analysis is effective the researcher will be able to disclose three predictors/characteristics of dysfunctional software development teams - the first literature available on this theme which can be expanded on by other researchers. The researcher will continue to refer to the literature to explore the three predictors/characteristics that exist in a dysfunctional software development team for clarity, whilst also reviewing what other researchers have said about software development teams. A quantitative analytical approach provides the report of the summarised results in mathematical terms with a specified degree of confidence (Savitri, 2013).

2.2. Characteristics of dysfunctional teams

2.2.1 The lack of commitment to development timelines

The meaning of "commitment", according to the Cambridge English Dictionary, is a willingness to give your time and energy to something that you believe in, or a promise or firm decision to do something. Inferred in this definition is an interpretation that commitment, being fully engaged, zeal, and going the proverbial extra mile will always come from people voluntarily if the objective or goal is shared with those who want to realise the desired end result. In contrast, those who are not committed will adopt self-preservation tactics such as verbal agreements with the decision but will not support it in their actions or will struggle to provide a timeline for delivery.

Lack of commitment may also mean that when teams engage in productive conflict, and genuinely share opinions and perspectives, they are more willing to buy into decisions that are made and commit to them. In many cases, team members do not need to get their own way to be able to support a decision; rather, they just need to know that their opinions were heard and considered. Great teams make sure that everyone's ideas are heard and discussed, making it easy for people to rally around the decision and ultimately make a better decision. Once everyone is bought into a decision, they will be committed to it.

If by any chance a team has not agreed to a choice, individual members who did not agree with the final decision will be less committed to it. Teams that fail to commit do not have clear goals, directions, and priorities, spend excessive amounts of time analysing situations, and frequently revisit the same conversation. They lack clearly defined goals, direction and priorities, or they may be defined, but they are not apparent because team members have disregarded the agreed upon goals and are working towards their individual goals.

Again, this often occurs because team members are not totally on board with the decisions. Teams find themselves frequently revisiting the same talk over and over, or spend an exorbitant amount of time discussing and analysing. Two of the greatest causes of lack of commitment are the desire for consensus and the need for certainty. Great teams find ways to get buy-in even when a complete agreement is impossible. As the researcher has already

mentioned, great teams understand that reasonable human beings do not always need to get their way; they just want to know they were heard.

Great teams recognise that they cannot delay decisions until they have 100% certainty that it is the right choice or they are without perfect information; rather, they realise that it is better to make decisions boldly and be wrong and then change direction rather than waffle and miss opportunities. Furthermore, if the team members are not committed to the course of action, they are less likely to feel accountable and this is even more risky if they are unwilling to hold their team members responsible.

Software development teams are increasingly failing to commit to timelines, especially when there is an overload in investigation and decision-making is at best slow or where there is a lack of confidence in the project plan timelines with fixed end dates of delivery without consultation regarding complexity, capacity and capability. This pressure inadvertently increases the fear of commitment within the team and consequently provokes the behaviour of self-preservation, and in more extreme situations, team preservation.

By default, teams that are committed will be united towards a specific common goal. This shows that there are clear objectives and roles for the team; hence there should be understanding and full agreement. Furthermore, they can practice their roles and support changes that are essential for the growth of the team. A united team learns from its mistakes of the past and has space to listen to everyone when the need arises and they understand each individual's role. By doing so they can provide assistance to everyone who needs it, discuss the intended goals, and at the end of the discussions there are clear and concrete resolutions (Neilson & Pasternack, 2005).

For a team to move away from being labelled as a dysfunctional software development team, the team members themselves need to hold each other accountable (Grenny, 2011), not be afraid to call others out when they exhibit a lack of engagement, and ensure they get back on the road of commitment. The objectives, goals, team roles, obstacles, risks, and the worst-case scenarios need to be clearly defined (Grenny, 2014). Given the discussions in meeting minutes, decisions that were made and changes in direction must be clearly developed and communicated, thus allowing everyone to stay in the know (Pigeon, 2011).

Software development teams need to stop working in their silo-based mentality and start engaging as a team. Software development team members need to support the team's decisions around any commitments on scope and effort of development required, articulating clearly if they oppose the unfair commitment by team members or unachievable timelines (Carpenter, 2014). Development teams need to focus on delivery as a team effort and not on individual efforts and stop avoiding resolutions and actions in meetings.

The lack of commitment to provide timelines for the delivery of code invokes another elements of behaviour within programmers, that of not being willing to go above and beyond. These individuals do not display an innate need to drive themselves to do more than is anticipated. These individuals will always seek "good enough" rather than "high-quality." Software development teams are accountable to deliver functioning software; if the development team isn't carrying its weight or is making too many mistakes, then the team's performance will be adversely affected.

If individuals in a software development team are not committed, they will not accept being accountable for delivery of software applications to the detriment of others relying on them (Page, 2015). When developers are not committed and refuse to be accountable, they will invariably not care about the quality of their output. Unfortunately the team will only aim for good enough outputs, and mediocrity will triumph, further stunting creativity and leaving no room for innovation (Ditkoff, 2015). In contrast, successful, high performing software development teams deliver vast outputs and acquire fulfilment in not introducing software defects, taking the time out to test their own changes before handover to the testing team, ensuring they own reported defects, and ensuring swift resolutions of the defects. On a global scale, less than one third of members state that they are totally committed to their current employers (Shuman, 2015).

According to Simon, "Working hard for something we don't care about is called stress; working hard for something we love is called passion" (as cited by Sinek, 2015).

2.2.2 Inattention to delivering quality software

A group that focuses on the effects also has the group status at heart. Individuals with team heart will always focus on the team's objectives. However, the people who consider personal

careers or goals tend to be less productive since they concentrate more on individualism than the common purpose of the team. Teams that are not focused on results are often caught up in focusing more on a team and individual status.

When one of the team members of the software development team shows a lack of attention to deliverables or is churning out-of-quality code, it shows a clear lack of respect for the team and sends a message of unprofessionalism. For some members of the team, merely being part of the group is enough to keep them satisfied and make them feel important. For them, concrete results, while nice, are not matter enough to be worth of any extra sacrifice or effort. Another common problem that detracts from a team's focus on results is when people focus on enhancing their positions or careers at the expense of the team, or they are more committed to another team.

Additionally, the software development team that avoids giving any attention to the code tends to cause harm. This is because the failure results in stagnant action points, hence the growth is minimal. When there is no growth, it is rare to exhibit an innovative spirit, thus encouraging a silo way of approaching solutions rather than collectively with the team. In contrast, teams that comprise individuals who focus their attention on the quality of software they produce are willing to make sacrifices for the team (Rosabeth, 2011); they feel the pain when the team fails to meet a goal.

It is quite a mission to eliminate inattention within the team but it requires the software development team members to hold each other accountable for the quality of code they produce, creating a buddy, or peer review, system and call other team members out when they demonstrate a lack of accountability. Teams need to establish the basis where these issues can no longer survive and eliminate the characteristics that facilitate these dysfunctional behaviours. Perfection in the service of the customer is the base of every successful service industry. The directly advantaged service industries and manufacturers among others are those that require reliable services (Valarie, 2014).

Fear means being sceptical about something, especially uncertainty. Teams in fear of conflict tend to hold boring meetings with political discussions and personal attacks. The team that manages to avoid critical related topics leads to the success of the team. More often than

not, the team involves themselves in wasting time with interpersonal risk management. Such a team poses a significant threat to development in any nation.

2.2.3 Avoidance of accountability

When team members are not entirely committed to a course of action they are less likely to be accountable. This is a dangerous situation because they are unwilling to hold their team members accountable. This often ties back to the first two dysfunctions, i.e. the lack of trust and the fear of conflict.

Accountability is achieved when software development team members and management hold each other accountable for their actions and inactions for the agreed output of the software (Carpenter, 2012). Team members who are not entirely committed will opt to avoid the discomfort that accompanies calling out a peer on his or her behaviour. Teams that are not accountable often find that there is much resentment between the members. High performers, or those who are fully committed and responsible, are frequently frustrated by those who have lower standards of performance and allow standards of quality to erode and who are satisfied with being mediocre. More obviously, teams that are not committed and accountable inevitably miss deadlines and key deliverables that quickly devalue the team (Robbins *et al.*, 2000).

Accountability is critical to a team's development, adding the element of trust to each individual that forms part of the team. Teams that avoid accountability will have resentment within the team as one developer's standards and expectations might not be the standard and belief of another developer in the team (Craemer, 2012). Software development teams that are unaccountable teams set very low achievement standards, whether they be for quality of delivery, amount of tasks delivered, or giving due consideration to the innovative aspect of the product delivered. The burden is constantly placed on leadership of reporting back to stakeholders of missed deadlines or commitments and this compels them to take on a dictatorial leadership style (Richardson, 2015).

In contrast, software development teams who are accountable identify and report idle behaviour that goes against the team or the organisational values, standards and

expectations. These software development teams identify potential problems quickly and are prepared with solutions when presenting such problems to management or stakeholders, and they establish a high level of respect for each member in the team. Conquering the avoidance of accountability requires open publication of software development standards, team goals, objectives, plans, metrics, etc. to ensure transparency within the team (Grenny, 2014).

Regular progress reports should be encouraged during outages or defect fixes to ensure that everyone is aware of the situation and that the next steps and impacts are clearly articulated. Perhaps they could, for example, reward the entire team with a bonus amount that is then shared equally amongst individuals in the team, thus almost forcing developers who do not fit in with a high performing development team culture to quickly move out, allowing the remaining team members to get the praise and rewards for the hard work and dedication. Without exception, as the team grows, members are expected to take responsibility for their output (Mackin, 2013).

Accountability must never be used as a device for placing blame or designating scapegoats. Developing accountability does not mean relinquishing accountability on management's part. It must be perceived as a partnership (Mackin, 2007). In the beginning of a group's development, management usually carries the lion's share of the accountability burden, absorbing the brunt of any disappointments. However, as the group matures, members expect to be held more accountable for their own results (Mackin, 2013).

According to the Merriam-Webster dictionary, trust is defined as the “assured reliance on the character, ability, strength, or truth of someone or something”. According to Cynthia, “Successful teamwork is built on a foundation of trust. Each member of the team must establish trust, cultivate trust through his actions and words, and work to maintain it. Each member also needs to be able to trust his team members to make a commitment to the team and its goals” (as cited by Measom, 2015).

Faith is defined as the power of confidence amongst individuals in the team; the people who do not need any peer influence to have a good reputation and to be protective to the group. Building trust allows teammates to get comfortable with being vulnerable with each other;

and sensitive to weaknesses, skill deficiencies, interpersonal shortcomings, mistakes, and requests for help. Trust means respect; not talking about someone when they are not present to defend themselves, and refraining from making jokes at the expense of others.

When participants in a group do not believe in themselves, then it is hard for them to share confidential information openly. Without trust, individual and team weaknesses are exposed in public, and also the group members will not request feedback or any assistance, and cannot take the time to help others in the team. More often, the team without trust quickly jumps to conclusions of negative intentions, not realising that they should recognise and utilise the interpersonal skills and cease wasting time on managing behaviours or holding grudges. Lack of trust leads to incorrect and unhealthy debates, yet solutions could be achieved through good decision making.

2.3. Contemporary history of software development teams

Software has been part of contemporary society for decades. Likewise, so have software development teams. The software development team is the foundation for developing software. Pressman defines a software process as a “framework for the tasks that are required to build high-quality software.” There is a link between quality, cost, and time impacting software development. Software developers are placed under tremendous pressure to deliver within budget and market requirements.

Today’s software processes stress quality over efficiency, which demands that software development teams need to work together to deliver a product of worthiness. Studies concluded by Anthony show that fixing defects after the product has been introduced into the market can cost from 60 to 100 times more than finding and eliminating the defects during the design phase (as cited by Keith, 2002).

IT experts work in an industry where there is a demand for well-designed software development projects, but if executed by a dysfunctional software development team they are prone to malfunction. Due to mounting competition and the present economic environment, businesses are ever more reliant on improving effectiveness and output through technology solutions. The role of the software development team has advanced into

one of noteworthy significance. In a cut-throat business environment, the quality of the software development team does not merely impact project failure rates, but also impacts business failure rates (Matta & Ashkenas, 2013).

2.4. Software development teams in organisations

Due to software development, teams are becoming the unavoidable unit of operation in organisations. The purpose and performance of an organisation is a direct manifestation of the function and performance of its software development teams; i.e. dysfunctional, inefficient teams can produce dysfunctional, inefficient organisations (Guttman, 2008; Tyson & Finn, 2000; Ross & Jones, 2008). Belbin (2000) offers his “Risk v. Complexity” model, signifying how productivity from workers has evolved from the talented individual to a collaborative group, and then to a balanced team, and ultimately, to concurrent balanced teams as the complexity of the task increases. As organisations seek to survive and thrive, it is the use of teams and their competitive advantage which makes or breaks the potential for success after downsizing, reengineering, and restructuring (Farren, 1999).

It is imperative that development teams within an organisation are effective teams and are contributing towards the success of the company, and not risking the organisation’s reputation or credibility. Many organisations need to focus on understanding where they can improve outputs from their software development team as identified by Stevens in 1998 that, “In the highly competitive software industry, improving the software development process can be critical to a company’s success.”

2.5. Team development resources

Teambuilding activities really took off in the 1970s, but managers observed that training was not leading to performance advancement or continued changes in group function (Dyer, 1977). By the early 1990s, research was entrenched in the areas of growing and managing effective teams. Dyer (1977) also formulated a step-by-step process for teams to develop by using a progressive approach, commencing with team members understanding the work of the team and each other’s roles, defining effective work processes and rules of engagement,

developing effectual communication processes and behaviours, and encouraging innovation while managing conflict (as cited by Shonk, 1982).

Additionally, team development principles have been supported in recent research and recommended practices (Castka *et al.*, 2001; Cohen & Bailey, 1997; Farren, 1999; Guttman, 2008; Katzenbach & Smith, 2003; Senge, et al., 1994; Sheard & Kakabadse, 2002). More recent research and organisational resources built on these basic premises and offered tools and techniques for team development and removing team dysfunction (Lencioni, 2002), teambuilding activities (Mackin, 2007), and collective learning (Edmondson *et al.*, 2001.)

Software developers are fundamental to the success of project delivery, as most of today's products are built and designed by the software developers, as stated. "Team members who are not committed to the project or the company are less likely to contribute causing low morale, missed deadlines and lack of trust among members. Tackle this problem in a dysfunctional team by picking members who are already invested in the project and want to make sound contributions" (Lencioni, 2008).

The research will uncover the characteristics of a dysfunctional software development team focusing on the three characteristics: 1) The lack of commitment to development timelines; 2) Inattention of delivering quality software, and 3) avoidance of accountability.

2.6. Empirical framework

O'Donnell (2011) provides a review of the five models that are prevalent in performing teams, beginning with the requirement of building trust, later engaging in constructive conflict, making appropriate commitments, embracing accountability, and scrupulously attending to results and performance. Each part builds on the other, and a deficit in any one of them will wreak havoc in a team.

Patrick (2002) outlines the root causes of politics and dysfunction in teams, and states the causes of dysfunction are both identifiable and curable. Patrick (2002) refers to five distinctive dysfunctions in a team: 1) the fear of being vulnerable with team members prevents the building of trust within the team, 2) the desire to preserve artificial harmony stifles the occurrence of productive ideological conflict, 3) the lack of clarity or buy-in prevents

team members from making decisions they will stick to, 4) the need to avoid interpersonal discomfort prevents team members from holding one another accountable, and 5) the pursuit of individual goals and personal status erodes the focus on collective success (as cited by Lencioni, 2002).

According to Patrick (as cited by Lencioni, 2002), the goal of a dysfunctional team is to lobby knowledge from how united teams behave and create strategies and recommendations to build a stronger team unit. Patrick (2002) defines the characteristics of high performing and cohesive teams as 1) comfortable asking for help, admitting mistakes and limitations, taking risks offering and feedback, 2) tapping into one another's skills and experiences, 3) avoiding wasting time talking about the wrong issues and revisiting the same topics over and over again because of lack of buy-in, 4) making higher quality decisions and accomplishing more in less time with fewer resources, 5) putting critical topics on the table and having lively meetings, 6) aligning the team around common objectives, and 7) retaining star employees. Organisations that employ this structure might have an advantage in achieving highly performing teams (Hamlin, 2008).

Although Patrick (2002) states that the five points, which he deems as relevant to the team, in reality, they only affect some teams. One must consider the interdisciplinary orientation of teams to understand what positive impacts team involvement can have, such as leading to personal recognition, creating attitude and character of each team member, and ultimately the entire team.

Cohesion in a group is necessary primarily because united teams make decisions faster and with more accuracy in achieving the end result. Team members can also rely on the skills and opinions of all members of the team. The cohesive nature of this team culture creates a competitive advantage and reduces time wastage and energy on internal politics and destructive engagements or conflicts. A team is said to be cohesive when it links individuals in that team together and does not get easily confused or side-tracked in achieving the end result (Forsyth, 2010).

In the past, software team members worked together from the same location, having face to face interactions; today a software team member can be located in different parts of the

world and in many instances in different time zones, conducting meetings via telephone or video conferences. Team members can struggle to become high performing teams where the team has never met face to face, communicating in an impersonal way through technology or working abnormal hours due to time zone differences (Asherman & Bing, 2010).

However, due to team members working in different parts of the world, if team members fail to appreciate the importance that culture can have on a team member's behaviour, this could lead to dysfunction in the team. Understanding cultural differences is important because each team member's internal values and attitudes are derived from their own cultural background and this may differ vastly from one country to another (Forsyth, 2010). Hofstede's research shows that it is necessary to understand that communication may differ based on cultural dimensions (as cited by O'Donnell, 2011), particularly in the areas of discussing power distance, individualism, certainty, time orientation and achievement, which will lead to respect amongst leaders and participants, and tolerance for change and feeling safe to share opinions and relative expressions. If the teams' interactions are not cohesive, and where issues of cultural and language differences are not addressed, it is a prominent sign that the team is not likely to meet its goals (on time or on budget) thus affecting the speed at which products get to market (Asherman & Bling, 2010).

The paramount issues of a strong team is based are the strength of cohesion and trust of a team; for example, learning from each other or the ability to work as a team to successfully deliver a task (Forsyth, 2010). Leaders need to build trust by communicating with the team, sharing experiences, giving assurance to their group to feel appreciated and comfortable with each other, and ensuring that the credibility within the team and in each team member is maintained. Some core guidelines that need to be put in place to help the team function well could include a shared understanding that the team will work together, for instance, by adopting Lencioni's model: 1) conflict guidelines, 2) confidentiality, and 3) consensus guidelines (as cited by O'Donnell, 2010).

Additionally, trust within a team can be achieved by requesting team members to complete personality and behavioural preference profiles, assisting individual team members to identify their personal behavioural tendencies which may improve the manner in which team

members communicate with each other, thus building common understanding and developing empathy (Lencioni, 2002). A tool that is widely used, which has indiscriminating acceptance, is the Myers-Briggs Type Indicator. Consider the dual concern, or rather the double conflict, of the model used to resolve conflict by using methods of “going against, yielding, working together and constructive criticism”. There are different levels of processes to be followed which may differ according to the individual concerns as opposed to the opinions of another team’s members (Forsyth, 2010).

Performing teams can work through difficult scenarios posed on the team and make the difficult decisions for the successful delivery of projects. They are able to resolve differences quickly and avoid conflict or disagreement that may cause permanent damage to the team member's relationships. Performing teams understand that conflict cannot just be avoided in the hope it may disappear; however, team members should view conflict as a source of creating team cohesion. The result is minor disagreements over issues which are viewed as important to improve the team’s performance (Forsyth, 2010).

Trust may be achieved through various different activities; for example sharing their personal stories, although the validity of such instruments remains open to question because it is not acknowledged nor is there the proof cited that the team’s confidence or purpose improves as a result of sharing personal information with team members (Hackman, 2006). However, it is paramount for the leader of a team to first create trust among the members to facilitate motivation and cohesion within the team (Lencioni, 2002). Individuals within the team that trust their own capability and that of their teams, develop overall confidence, their focus increases in delivering quality and timely delivery of tasks, and they develop clear objectives hence leading to a highly productive team (Ray & Elder, 2007).

For trust to exist in teams, a safe environment must exist for members to share their opinions, and therefore conflict amongst the team members should not be discouraged, as the encouragement to share opinions may lead to genuine discussions and debates that are crucial to the team working together to reach an agreed end result. However, leaders should control and monitor team members that continuously remain destructive or drive negative behaviour within the team that will eventually lead to the team losing trust in each other

(Lencioni, 2002). Leaders should encourage “good conflict”, which targets real theories or ideas that leads to quality solutions or positive outcomes. Team members will have different viewpoints and difference of opinions may escalate to conflict, but the manner in which conflict is addressed will determine whether it contributes positively or negatively to the team culture. Schellenberger states that the significance of conflict resolution is important, as it raises the individual’s consciousness, increasing the focus on efforts to arrive at a positive solution or agreement (as cited by Hamlin, 2008).

Teams that lack of commitment to team goals and successful delivery, encourage second-guessing among team members. Leaders tend to spend a lot of time within such teams to assist in creating clear outlines for delivery and to some extent taking responsibility for restructuring current action plans and spending time continuously communicating and reporting status. Performing teams take on the role of self-management, outlining their roles and goals to achieve the desired outcome. Additionally, overcoming a lack of commitment by individuals in the team will contribute to respecting deadlines and focusing on the successful delivery of tasks (Lencioni, 2002).

Teamwork is key to the success of team. According to Massey (2010), individuals who work in unity deliver quality and show an increase in the number of tasks completed by the team. The total performance is bigger than the total sum of the individual parts, states Hamlin (2008). Similarly, Forsyth (2010) notes that synergy is the combining of two or more independent systems that yield an effect that is greater than the sum of the individual effect. In groups, if synergy amongst team members exists, the team’s productivity increases. The group can collectively achieve something that could not have been possible if the reliance was on one person to deliver the solution individually. For team members to be responsible for each other’s behaviours and actions, they must have a clear view of what is expected (Lencioni, 2010).

According to the dual concern model of conflict, instead of openly debating, team members who choose to avoid conflict attempt to evade the topic, avoid meetings, or may even choose to leave the team altogether (Forsyth, 2010). When team members choose to either blame other individuals or do not focus on delivering their allocated task, there is a lack of

accountability culture within the team. In such teams, the leader's ability to capitalise on horizontal accountability to enable the members of the group assume high levels of responsibility for goals and performance is minimised (Ray & Elder, 2007). Team members must also hold each other accountable to increase the productivity and quality of work the team delivers. Hackman (2006) points out that avoidance of accountability is driven by the low esteem of the group's way of doing things. Ray and Elder (2007) state that horizontal accountability still stands as the optimal degree of people's communication across the industry. Leaders may utilise horizontal accountability when providing feedback to individuals in the team that are lacking, or not accepting, accountability within the team, emphasising the need for each individual in a team to take responsibility for targets and performance. In doing so, the leader evaluates performance based on feedback from team members on the individual's performance as they are directly impacted by the lack of accountability displayed by any team member. However, the leader should present all feedback in a useful and non-threatening format to the team members, to avoid causing friction or a break in trust levels within the team (Ray & Elder, 2007).

If team members are not held accountable for what they are doing, they will likely attempt to draw the attention to their personal needs and development or behave in a way that is negative to the health of the team. Lack of accountability causes team members to lose morale and the common objective of the team will not be realised (Lencioni, 2002).

Focusing on results is a positive characteristic noticed in performing teams. The outcomes motivate the team to increase effective performance. Luecke (2004:43) comments, "It is nearly impossible to succeed when team members cannot articulate a clear and shared goal." He adds, "People must see their team's goal as being imperative and worthy of purpose" (Luecke, 2004:43). Lencioni (2002) suggests that an efficient manner to ensure team members focus their attention on results is to connect their rewards, such as compensation, to the achievement of the particular outcome. This relates to the social exchange theory, which states that individuals look for relationships that offer them many rewards while taking few costs (Forsyth, 2010).

CHAPTER 3:

Methodology Framework

3.1. Introduction

The methodology used in executing the research was completed in a way that ensured that it was easy to explain. The chapter describes the research design, the target population, sampling design, and the data collection tools and procedures used during the research.

3.2. Conceptual model

Soliciting from the literature review, in extracting the theoretical and empirical literature revealed below, a research model was conceptualised. In the conceptualised research model, the influence of the lack of commitment to development timelines, inattention to delivering quality software, and avoidance of accountability will be predictors of a dysfunctional software development team.

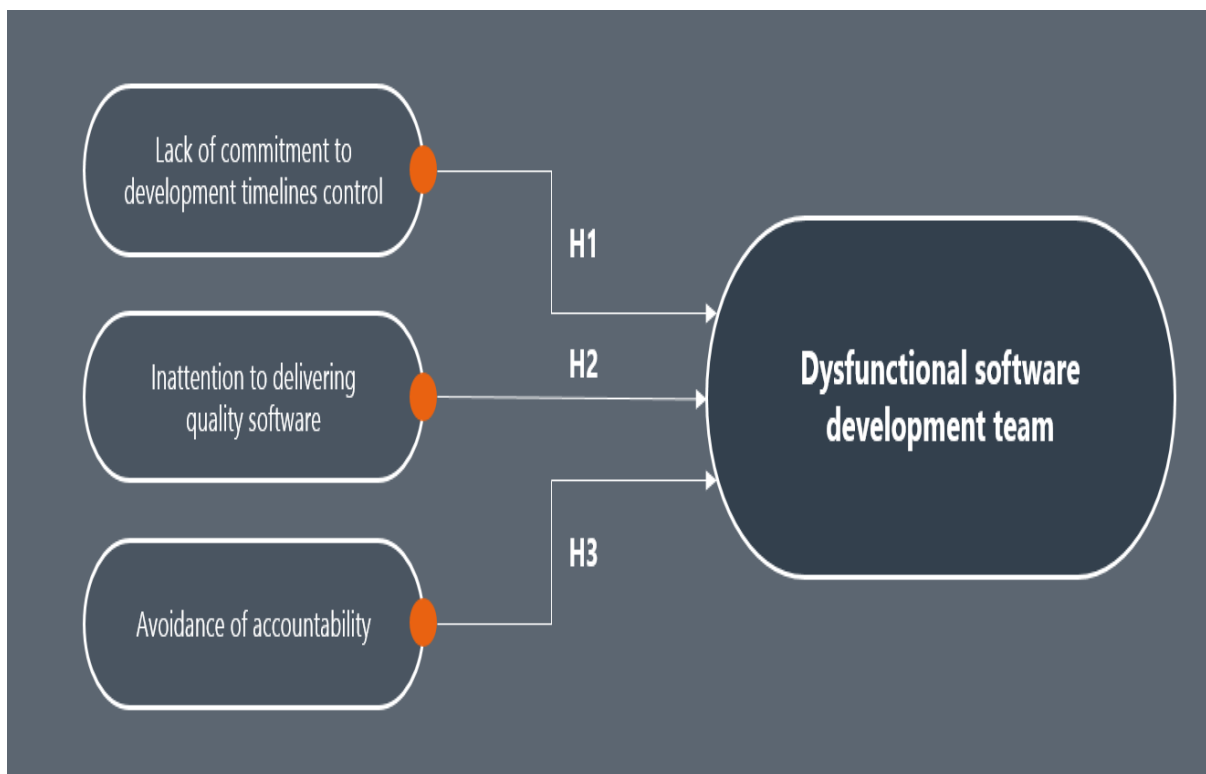


Figure 3-1: Conceptual model

3.3. Hypothesis statement

Based on the above conceptual model the following hypotheses are stated.

Hypothesis 1:

The lack of commitment to development timelines has an influence on dysfunctional software development teams.

Hypothesis 2:

The inattention to delivering quality software has an influence on dysfunctional software development teams.

Hypothesis 3:

The avoidance of accountability has an influence on dysfunctional software development teams.

3.4. Ontological and epistemological approach

This study was set in positivism epistemology. Positivism belongs to epistemology which can be specified as a philosophy of knowing, whereas methodology is an approach to knowing. The reason for choosing to follow the philosophy of positivism, was because factual knowledge was gained by observation and the use of senses, including measurement, leading to trustworthiness of the findings. The researcher was independent of the research, and there were no provisions for human interests within the study. The research findings were observable and quantifiable (Dudovskiy, 2010). Crowther and Lancaster (2008) inform that as a general rule, positivist studies usually adopt a deductive approach, whereas the inductive research approach is usually associated with phenomenology philosophy. The researcher concentrated on facts, as opposed to phenomenology which focuses on the real meaning because it has provision for human interest.

Quantitative research was an appropriate methodology for studying the three characteristics of a dysfunctional software development team and assisted in gaining “novel and fresh slants on things about which quite a bit is already known” (Strauss & Corbin, 1990:19). The research explored the perspectives of a variety of participants.

Quantitative research, or statistical data survey, requires a process involving the collection and analysis of quantifiable data and therefore presented as the best method for the research. The data for this research was quantifiable and could be counted or calculated mathematically using mathematical formulae. The quantifiable data allowed the research the means and methods of statistically calculating the data that was collected. Therefore, quantitative research was the chosen methodology as it provided a way of acquiring information that could be inferred or generalised to a significant or rather large population of people. This concept is referred to as generalisability (McCallister, 2013).

The following section will outline the process taken for participant selection, data collection, analysis, review, and theory development. With regards to data collection and analysis, and initial sampling, the researcher will rely on data provided in the questionnaire (Corbin & Strauss, 2008). The research focused on information technology software developers in selected organisations. All participants received an email with consent forms (following theoretical sampling, which involved the research for relevant data to develop the emerging theory (Charmaz, 2006). The research approach chosen form was theoretical sampling, which was not intended to be random; it was “strategic, specific, and systematic” (Charmaz, 2006: 103).

3.5. Research methodology

Research methodology refers to a system of transparent rules and procedures upon which a study is taken contrarily or against which claims for knowledge are evaluated (Nachmias & Nachmias, 1997). It signifies the techniques used to attain and analyse data to generate new knowledge or insights (Petty *et al.*, 2012). Research methodologies have an influence on the validity and overview of a research, and play a crucial part in knowledge development (Yang *et al.*, 2006). The methodology chosen for this research was Quantitative research.

3.6. Research design

According to Malhotra and Birks (2007), research design refers to a plan for carrying out a research project. It maps out procedures for gathering information that will be used to answer questions and solve problems (Malhotra & Birks, 2007). The research design establishes the foundation for the entire research (Malhotra & Birks, 2007). The design of this study is going to be quantitative in nature. To define quantitative research, Burns and Grove (2005) explain it as a formal, objective, systematic process in which numerical data are used to obtain information about the world.

A survey-based quantitative evaluation was adopted in this research providing the platform to describe certain behaviour/attitudes of software developers, and allowed for the same information to be collected from each and every respondent. A research questionnaire was administered to participants and a four point Likert-scale was used to collect data from the respondents. The use of this design was justified due to its uniqueness of enabling the research to gather original information which was in the first place not available from other sources.

The research method was used to describe variables, to examine relationships among variables, and to identify the cause and the effect or the correlation and interaction between the various variables (Burns & Grove, 2005). The research findings was based on accumulating numerical data to describe the particular behaviour and attitude apparent in a dysfunctional software development team, with specific questions deemed directly suited to being answered. Through the process of analysing and synthesising, processing, the information, data gave credence to testing the conceptual model and hypothesis statements. A systematic organised way of collecting the data was followed during the research, which required careful selection of the units studied and careful measurement of each variable (Howie, 2015).

The research design content took into consideration the following information:

- a) The information essential for this research.

- b) Provided guidance in shaping that the entire design would be exploratory.
- c) A specific order in which measurements were completed.
- d) The questionnaire constructed that was relevant for the research.
- e) The quantitative procedures used in the sampling and the sample size were simple.
- f) The plan developed for the quantitative data analysis was support of the research.

The research was guided by the guidelines of Quantitative research methodology which was necessary in order to recognise the unit of investigation and exploit attuned methods that led to the results. A research design included the following: how data was to be attained, what instruments were used, how the instruments were exploited, and the intended means for analysing the data collected.

3.7. Quantitative research

A quantitative research approach was used for the purpose of this research. Following Polit and Hungler (1995), the approach utilised an organised procedure and technique to gather information under controlled conditions and highlighted objectivity through statistical analysis which proved that the quantitative approach was more appropriate given the nature of the current research.

3.7.1 Sampling design

A sampling design chosen for this research was simple to implement, efficient, and covered various approaches to measure the sample (Grafstrom, 2010). The sample design comprised of the sampling method which included the procedures, methods, and rules for having the elements of the population included in the sample. The **sample design** chosen was simple random sampling, and this was due to the fact that the processes the researcher was testing were assumed to be fundamental and universal in as much as they can be generalized beyond a small sample to a wider population

3.7.2 Target population

The identification of the study population was necessary for the formulation and proving the hypothesis (Klein & Meyskens, 2001). When defining a target population, the research included software developers that apply directly to the research (Kohn, 2008). The populace in this study referred to the entire group of interest that the researcher will investigate. This meant all software development individuals will be considered populace in this research.

3.7.3 Sample frame

The sampling frame in this research referred to the list of all elements in the population from which the sample will be drawn. A sample frame refers to the researched environment and the topics used in a research (Yang *et al.*, 2006). Non-probability sampling was engaged (simple random sampling) for this research, and this was due to the fact that the processes the researcher will be testing are assumed to be fundamental and universal in as much as they can be generalised beyond such a small sample to a wider population.

The sample frame used for the purpose of this research was software developers employed globally (South Africa, Australia, India, Mauritius, UK and Germany). A random sample of 185 developers was used to collect data for this research.

3.7.4 Sample size

Sample size determination was crucial in planning this statistical research, as well as challenging (Length, 2001). The sample size was determined using the Raosoft sample size calculator. Raosoft is statistical software used to calculate sample size and was chosen because it took into the following factors; 1) the margin of error, 2) the confidence level, 3) the population size, and the 4) response distribution. For this research the default settings on Raosoft where a margin of error at 5%, a confidence level at 95%, a population size at 18 500 and a response distribution of 50%. Raosoft then calculated that the sample size needed for the survey would be 185 respondents.

3.7.5 Sample method

The sampling method that was most appropriate for this research was non-probability sampling as it gave every unit within the population an equal chance to be sampled by the researcher (Gaplin, 2011). The chief aim was to ensure that all software developers have an equal opportunity of being included in the sample and attained objectiveness which was why this method was chosen for this research. This method also ensured that the research was unbiased in nature, which was necessary in a scientific research (Bryman, 2012).

3.8. Questionnaire design

3.8.1 Measurement instruments of constructs

The theoretical constructs that are measured are the formative factors for the option of measurement methodology (Fagarasanu & Kumar, 2002). The initial data collection tool used for this research was the research questionnaire. The questionnaire was developed by the researcher. The decision to use a questionnaire to collect data from respondents as it was seen to be the most appropriate tool for collecting the data, whilst still remaining independent of the research. The questionnaire was designed based on the research model.

The researcher's measurement instruments was measured on a 4 point Likert-type scale, anchored by 1= strongly disagree to 4= strongly agree, to express the degree of agreement or lack thereof. Both closed ended and open ended questions were used. Likert scaling was chosen due to its unidimensional scaling method, requiring individuals to construct a decision on their level of agreement, generally on a point scale (i.e. 1) Strongly Agree, 2) Agree, 3) Disagree, and 4) Strongly Disagree) with a statement. The number beside each response became the value for that response and the total score was obtained by adding the values for each response (Trochim, 2006).

The research benefited from the advantage of a Likert-type scale as it was simple to construct; each item was of equal value so that response are scored rather than items, thus it produced a highly reliable scale, was easy to read and complete (LaMarca, 2011).

Questions were designed to collect three different types of information from the target population namely behavioural information, attitudinal information; with classification of each question under the three identified characteristics of a dysfunctional software development team.

Appropriate instructions on completing the questionnaire was attached to the email sent, to facilitate ease of understanding.

As the researcher chose to have a larger set of questions, questions were precise and to the point, with proper numbering, clear font size and was not introduced with a heading of the targeted characteristic researched thus removing any further bias.

Before circulating the questionnaire to its intended respondents, a beta testing was conducted on a small group of developers to ensure that any typographical errors or ambiguous questions were corrected.

3.8.2 Lack of commitment to delivery timelines questions

A 4 point Likert scale was used that assessed a participant's response to how a team member's behaviours affected the team (questions below), for which lower scores (1 - strongly agree, and 4 - strongly disagree.) represented a higher level of perceived characteristic of a dysfunctional team.

- When a team member does not meet development timelines, the team is affected negatively.
- The team member's behaviour is causing resentment within the team by not meeting deadlines.
- The team member's behaviour is causing hostility within the team by not meeting deadlines.
- Other team members have to do more work when a team member does not meet deadlines.

3.8.3 Inattention to delivering software quality questions

A 4 point Likert scale that assessed a participant's response to how a team member's behaviours affected the team (questions below), for which lower scores (1 - strongly agree, and 4 - strongly disagree.) represented a higher level of a perceived characteristic of a dysfunctional team.

- The team is negatively affected when team members are negligent in delivering quality code.
- The team member's behaviour is causing resentment within the team by delivering software with defects that requires another team member to fix.
- The team member's behaviour is causing hostility within the team by delivering software with defects that requires another team member to fix.
- When a team member makes mistakes repeatedly it affects the team negatively.
- Other team members have to do more work when a team member delivers bad quality code.

3.8.4 Avoidance of accountability questions

A 4 point Likert scale that assessed a participant's response to how a team member's behaviours affected the team (questions below), for which lower scores (1 - strongly agree, and 4 - strongly disagree) represented a higher level of a perceived characteristic of a dysfunctional team.

- When a team member does not take accountability of their task the team may be adversely impacted.
- The team member's behaviour is causing resentment within the team by frequently blaming other team members for non-delivery of tasks.
- The team member's behaviour is causing hostility within the team by not completing their development task.
- When a team member does not "carry their load", it leads to bitterness within the team.
- Other team members have to do more work when a team member does not complete their tasks.

3.8.5 Additional questions: combining lack of commitment to delivery timelines, inattention to delivering quality software, and inattention of accountability characteristics

- When team members are resentful there is an increase in conflict.
- When team members are hostile there is a loss of trust within the team.

3.9. Data collection

The research followed the process of gathering and taking measurement of the information on a targeted group of variables in a certain systematic fashion which eventually provided answers to the questions and an evaluation of the outcome of the answers received. To obtain quality scripts, the questionnaires were emailed to software developers. This research adopted the primary method of collecting data, that collected information for the specific purpose of this research and the questions were tailored to elicit the data that would help in this research.

To collect data from participants a comprehensive questionnaire was developed whereby random software developers were selected to answer questions pertaining to the behaviours and attitudes of developers.

3.10. Data analysis approach

The research process followed during data analysis was a statistical process in which raw data was prepared and structured so that valuable information could be extracted from it (Ullah, 2010). Firstly, the collected data was coded in an Excel spread sheet before analysis. To gain comprehension of the attributes of each variable, descriptive statistics analysis was utilised which was depicted by the mean and standard deviation of each factor. The researcher was responsible for analysing the data. Number Cruncher Statistical System (NCSS) was used to analyse the data.

3.11. Reliability and validity of measurement scales

Reliability and validity in this research referred to the logic and accuracy of the data collected (Wilckens, 2010). Attaining reliability for this research meant achieving comparable data, while validity led to ensuring the answer to the questions were valid in logical terms (Wilckens, 2010).

In order to ensure reliability of data, the research employed the following:

- Advanced Statistics package for Number Cruncher.
- Bootstrap.
- Box-Muller transformation of 1998.

As this was a positivist research, the validity in quantitative research was mainly concerned with whether the measurement of the conceptual model fundamentally measured what was it designed to measure (Bryman 2004; Bryman, 2012). The research followed a process by which it could be proven that both the collection technique used and analysis procedures followed were such that coherent results were generated. Reliability was validated by employing the Advanced Statistics package for Number Cruncher to determine the alpha values and the composite reliability values (Bryman, 2012).

External validity therefore is chiefly concerned with the generalisability of specific results of a specific research beyond the sampled research context. Therefore, the findings of this study can be replicated to any study that is similar in nature to this research, anywhere in the world. Most importantly, the research was mainly concerned with satisfying whether the analysis and its conclusions were sound, especially when there was a causal relationship between two or more of the variables being discussed (Bryman, 2012).

Following Bryman (2012), the research addressed the reliability I checking the consistency of the measure: checking the stability by evaluating the measure was stable over time, checking if the scales or indices were consistent, checking if there was any subjectivity or lack of consistency present.

3.12. Data analysis procedure

Before the data collected from this research could be analysed, certain checks for the legitimacy of the data had to be conducted and the necessary consideration were given so that no incorrect data were entered on to the Excel spread sheet and if so entered, it was removed. Following Trochim (2000), immediately after receiving the collected data, the research concluded screening for accuracy. The screening process helped to identify any errors that could have been present in the research sample. Guided by Trochim (2000) the following questions were asked in order to successfully check for discrepancies and inconsistencies:

- Are the responses written clearly?
- Did the respondent answer all important questions?
- Did the respondent complete the questionnaire?
- Does the questionnaire contain all the relevant contextual information such as data, time, place and the researcher's details?

3.13. Descriptive statistics

In this research, descriptive statistics involved simple summaries about the sample and the dimensions of the data collected. The descriptive statistics was in of tables that showed the basic data of the main components of the research including statistical procedures that the research adopted to describe the population being studied. The data was collected from a sample population, and the results helped organise and describe the data.

3.14. Structural equation modelling approach

Structural Equation Modelling (SEM) was used as the statistical technique for performing regression analysis, factor analysis and simultaneous equation modelling (Wothke, 2010) of the collected data. This research mainly required the use of Structural Equation Modelling to conduct Confirmatory Factor Analysis and Path Modelling (Hypothesis Testing) for the purpose of the quantitative research approach taken.

3.14.1 Model fit (goodness of fit)

Bootstrap was used as a goodness of fit test (Galpin, 2011). The Bootstrap measure was to completed to evaluate the overall model fit and assessed the magnitude of discrepancy between the sample and fitted covariance matrices. A good model fit would provide an The result at a 0.05 threshold, showed a good model fit (Barrett, 2007).

3.14.2 Path modelling

This research adopted path modelling to describe the relationships between measured variables and theoretical constructs (Roche *et al.*, 2011:1480) and tested the structural paths of the conceptualised research model. Once the model fit was assessed using Bootstrap, the research proceeded to perform Path Modelling using Standard Equation Modelling using the Partial Least Squares Package.

3.14.3 Confirmatory factor analysis

Confirmatory factor analysis is an analytical tool that allowed for the research to explore the stated hypotheses as to what constructs the test in question was measuring and provided an empirical basis for the scientific interpretation (Burton *et al.*, 2003). This process involved the separation of a large number of variables into a smaller number of factors within which all variables were related to each other, to investigate the underlying variance structure of the set of correlation coefficients. A confirmatory factor analysis was performed to obtain the standard regression weights. Model fit indicators such as Bootstrap, Goodness of Fit Index (GFI), Box-Muller transformation, advanced statistics package for Number Cruncher and Structural equation modelling using partial least squares were used to assess the model fit.

CHAPTER 4:

Data Analysis & Discussion of Results

4.1. Introduction

In this chapter the results of the data analysis will be presented. Data analysis is a statistical process in which raw data is prepared and structured so that valuable information can be extracted from it (Ullah, 2010). Firstly, the collected data will be coded in an Excel spreadsheet before analysis. To gain comprehension of the attributes of each variable, descriptive statistics analysis will be utilised which will be shown by the mean and standard deviation of each factor. The researcher will be responsible for analysing the data. The data were collected and then processed in response to the problems posed in Chapter 1 of this dissertation. The data used are a sample of the whole population.

Two fundamental goals drove the collection of the data and the subsequent data analysis. Those goals were to develop a base of knowledge about the software development team specialist, team organiser, construction asset as perceived and utilised relative to other organisation organisers, and to determine if lack of commitment to development timelines, inattention to delivering quality software, and avoidance of accountability are predictors of a dysfunctional software development team.

The following objectives were accomplished:

According to the research and survey carried out, it was discovered that lack of commitment to development guidelines significantly affects the function of a team. Secondly, inattention to delivering quality software and avoidance of accountability correlate well in leading to the dysfunction of a team. The findings presented in this chapter demonstrate the potential for margin theory and practice.

4.2. Response rate

One hundred and eighty five (185) surveys were initially sent to software development teams who were identified within various organisations with a strong information technology presence. The software package directory was sent out but five of the sent samples were not functional and did not offer viable output that could help significantly in preparation.

Therefore, 180 surveys were considered to be legitimate for this research. One hundred and seventy five (175) useable surveys were returned. Ten additional surveys were returned that were not considered useable. The unusable surveys were either blank with a note attached which explained why the respondents was not be able to complete the survey, or were only partially completed surveys with major portions of the survey left blank, and in one case the respondent created and revised categories such that the data could not be entered without serious interpretation and alteration. With 175 returned and useable surveys out of 185, the response rate was 94.59%.

Of the 175 surveys, 62 were sent to organisations with modernised technology and high sales output to identify whether the team there lacks commitment to meeting development timelines. They were to observe and fill in the questionnaires as per the observation, and also they could ask questions on parts of the questionnaire that were not clear to them.

The sample picked was contacted based on random selection. The next 90 individuals were listed and the research questionnaire was emailed, with clear responses from 87 individuals who had full knowledge of the effects of lack of commitment to development timelines, among other factors. Therefore the response rate was 96.67%. Twenty one questionnaires were emailed to the faculties, also randomly selected from the population of the total industrial population where an internal information technology team exists; these individuals were believed to be knowledgeable on technology and could easily understand the dysfunctions of a software development team in an organisation.

4.3. Active programs

Eighty seven per cent (87%) of the total responses indicate that all programs were active and significant or valid for the study. These surveys were considered valid because the information acquired was sufficient for the researcher to proceed with the study.

4.4. Medoid clustering

Following the data entry, the first step in this analysis involved medoid clustering of the data. In the questionnaire, the survey instrument used to capture the respondents feedback, they were asked to evaluate how lack of commitment to development timelines, the influence of the inattention to delivering quality software, and the influence of avoidance accountability lead to the dysfunction of a team. The respondents were asked to rate each of these statements using a Likert scale, where 1 equalled strongly agree, 2 equalled agree, 3 equalled disagree, and 4 equalled strongly disagree.

4.5. The Medoid clustering program of number

Cruncher was used to process this information for the purpose of sorting the respondents into clusters according to their Section 2 responses. To determine how many clusters would produce the best fit for the given data, the default setting of two to five clusters was accepted. The program calculated an objective function value for each cluster combination. The objective function values were: 4126.212 for a two cluster model, 2595.681 for a three cluster model, 1875.044 for a four cluster model, and 1441.752 for a five cluster model.

The figure below is a graph of those values. With the aid of the graph one can see that the objective function value decreases sharply from two to three clusters and then the decrease become less pronounced. Regarding the selection of the number of clusters, Hintze (1992) states, "You want to choose the number after which the objective function seems to quit decreasing at a rapid rate". Therefore, the three cluster model was selected for further analysis. Hintze (1992) suggests discriminant analysis as a means for analysing "the appropriateness of the cluster configuration"

Results of Medoid Clustering

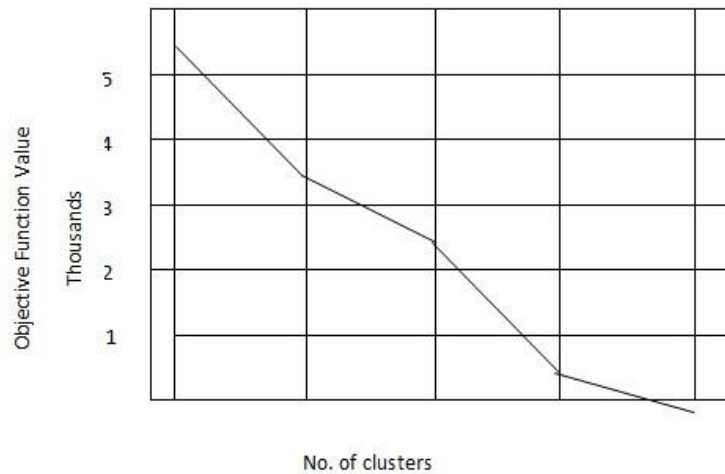


Figure 4-1: Results of Medoid clustering

For appropriate and accurate evaluation of the clusters, the respondent information provided in the Cluster Report was added to the data set. The table below lists the row number of each respondent according to the cluster assignment.

In order to evaluate the appropriateness of the clusters, the respondent information provided in the Cluster Report was added to the data set. The table below lists the row number of each respondent according to the cluster assignment.

Table 4:1: Respondent assignment to the clusters in a three cluster model

Cluster 1	5, 9, 11, 15, 16, 17, 19, 20, 24, 26, 28, 32, 33, 34, 40, 42, 59, 60, 63, 72, 73, 76, 84, 86, 90, 92, 93, 97, 98, 100, 101, 107, 109, 115, 119, 125, 129, 131, 134, 136, 139, 145, 147, 151, 153, 155, 159, 164, 168, 170, 174, 160
Cluster 2	3, 4, 8, 10, 13, 14, 23, 27, 29, 35, 38, 45, 46, 47, 48, 49, 50, 52, 54, 55, 62, 65, 67, 68, 69, 74, 77, 80, 83, 87, 89, 91, 96, 99, 102, 104, 105, 106, 108, 110, 111, 112, 117,120, 122, 124, 126, 130, 132, 135, 137, 142, 144, 146, 149, 154, 156, 158, 163, 166, 169, 172, 175
Cluster 3	1, 2, 6, 7, 12, 18, 21, 22, 25, 30, 31, 36, 37, 39, 41, 43, 44, 51, 53, 56, 57, 58, 61, 64, 66, 70, 71, 75, 78, 79, 81, 82, 85, 88, 94, 95, 103, 113, 114,116,118,121,123,127,128,133,138,140,141,143,148,150,152,157,161,162,165 ,167,171,173

Based on the information displayed above, a dummy variable column was added to the original data set to check whether the clusters were functional and confirm whether the introduction to the dataset clustering with highlight a potential error. Members of cluster 1 were assigned the number 1, the members in group 2 were assigned the number 2, and the members in group 3 were given the number 3.

4.6. Discriminant analysis

Following the assignment of group related dummy variables, the data were evaluated using the discriminant analysis program in the Advanced Statistics package for Number Cruncher. The discriminant analysis program gives a feature function, called the misclassified row report. The row report allows for reassignment of the respondent to the group having the potential for reducing the hidden error. Given each of the re-classified clusters, another analysis was performed. This analysis was automated to instruct and command the program to search the variables with the best fit set.

Further, only variables selected met or exceeded alpha 0.05 probability level for an F-test. In the table below, the necessary information obtained from the original calculation is shown.

Table 4:2: Tabular results of initial discriminant analysis

Tabular Results of Initial Discriminant Analysis - the clusters were Assigned									
According to Medoid Clustering Results					Variable influence report				
if removed					If alone				
variable	lambda	F-value	F-Prob	lambda	F-value	F-Prob	R ² -X's	mean	
A	0.087009	7.76	0.0007	0.74799	18.7	0	0.17008	1.526316	
B	0.90208	5.64	0.0047	0.80877	13.12	0	0.15332	2.052632	
E	0.87305	7.56	0.0009	0.72028	21.55	0	0.24008	2.149123	
G	0.90621	5.38	0.006	0.78847	14.89	0	0.31244	1.736842	
I	0.80811	12.35	0	0.6959	24.25	0	0.22924	2.377193	
J	0.88028	7.07	0.0013	0.68778	25.19	0	0.288	1.850877	
K	0.86411	8.18	0.0005	0.66215	28.32	0	0.29212	2.640351	
L	0.84009	9.9	0.0001	0.74898	18.6	0	0.25043	1.307018	
Group means									
	1	2	3						
A	2.060606	1.380952	1.230769						
B	2.575758	1.595238	2.102564	Overall					
E	2.69697	1.547619	2.333333	1.526316					
G	2.272727	1.666667	1.358974	2.052632					
I	2.636364	1.761905	2.820513	2.149123					
J	2.484848	1.690476	1.487179	1.736842					
K	2.454545	2.095238	3.384615	2.377193					
L	1.69697	1.166667	1.128205	1.850877					
				2.640351					
				1.307018					
Count	33	42	39	114					

F-value is used to test whether this function and those below it are significant. Included in the table above are two different reports that were generated from the data using the analysis program. From the top, there is the Variable Influence Report, and just below it follows the Group Means report. Mainly these reports identify those variables that are unbiased between the layers or clusters. Furthermore, they give a mean value for each variable as it relates to the group. Additionally, the overall percentage of variance in the strata is indicated, and details explained by the cluster combination (Langan-Fox *et al.*, 2007).

4.7. The design of simulation

For a sample size study, the Bootstrap calculates an estimate to be used to compute the alpha hat ($\hat{\alpha}$) rather than the asymptotic estimates (Yuan *et al.*, 2003). The Bootstrap uses the raw data, but on the other hand, the asymptotic method uses the correlation or covariance matrix that does not include any information concerning the sample size. Therefore, the most appropriate method used on Likert-type data is the Bootstrap, and it was generated using the Box-Muller transformation of 1998 in the simulation phase of this research.

In the first stage, D=10,000 population data were generated, each of which included, N= 5,000 observations and different values of randomly generated and determined variables. The coefficients alpha (α) and lambda (λ_1) are obtained from PCA based on the calculation from each of the population data set, 100 samples were drawn by simple random sampling with the replacement for each n=30, n= 100 and n=500. For each sample data set, the Bootstrap estimator of the estimator alpha (α) and lambda (λ_1) were computed. During the final stage of the simulation process, the relative bias (R-Bias) and relative root mean square error written as (R-RMSE) were computed for every one hundred (100) samples (replication) in each sample size level. The relative bias and the relative root mean square error were estimated for each estimator.

$$\text{R-Bias } (\hat{\alpha}_{pi}) = \frac{1}{M} \left[\sum_{j=1}^m \frac{(\hat{\alpha}_{pij} - \alpha_p)}{\alpha_p} \right]$$

$$\text{R-RMSE } (\hat{\alpha}_{pi}) = \sqrt{1/M \left[\sum_{j=1}^m ((\hat{\alpha}_{pij} - \alpha_p))^2 \right] / \alpha_p^2}$$

Where ($\hat{\alpha}_{pij}$) is an estimator of the α_p for the jth replication and ith sample size (i=30, 100,300,500) and in the pth population data set where (p=1, 2,..... 10,000), and M is the number of replications. In this case, when the values of relative bias were less than 0.01 and the values of relative root mean square error were less than 0.05, the estimator for the parameter was deemed acceptable.

Following this observation, the performance of $\hat{\alpha}$ for each sample size (n=30,100,300,500) may be observed. Additionally, another issue that is observed is the item numbers regarding the population data set. Even though the numbers are randomly determined in the simulation,

the lower and the upper limits for each item were determined in advance. Accordingly, they are partially controlled as $5 \leq k \leq 20$. The table below represents the frequency of item numbers in the $D = 10000$ data set generated.

Table 4:3: Item numbers of generated data in simulation

K	frequency	k	frequency	k	frequency	k	frequency
5	621	9	637	13	627	17	594
6	623	10	661	14	634	18	652
7	627	11	614	15	610	19	621
8	635	12	620	16	630	20	594

The simulations in the study were performed using the SIMREL software. A modified form of the SIMREL software application was developed for the simulation phase of the case study. The algorithm of this application was constructed chronologically as per the representation below.

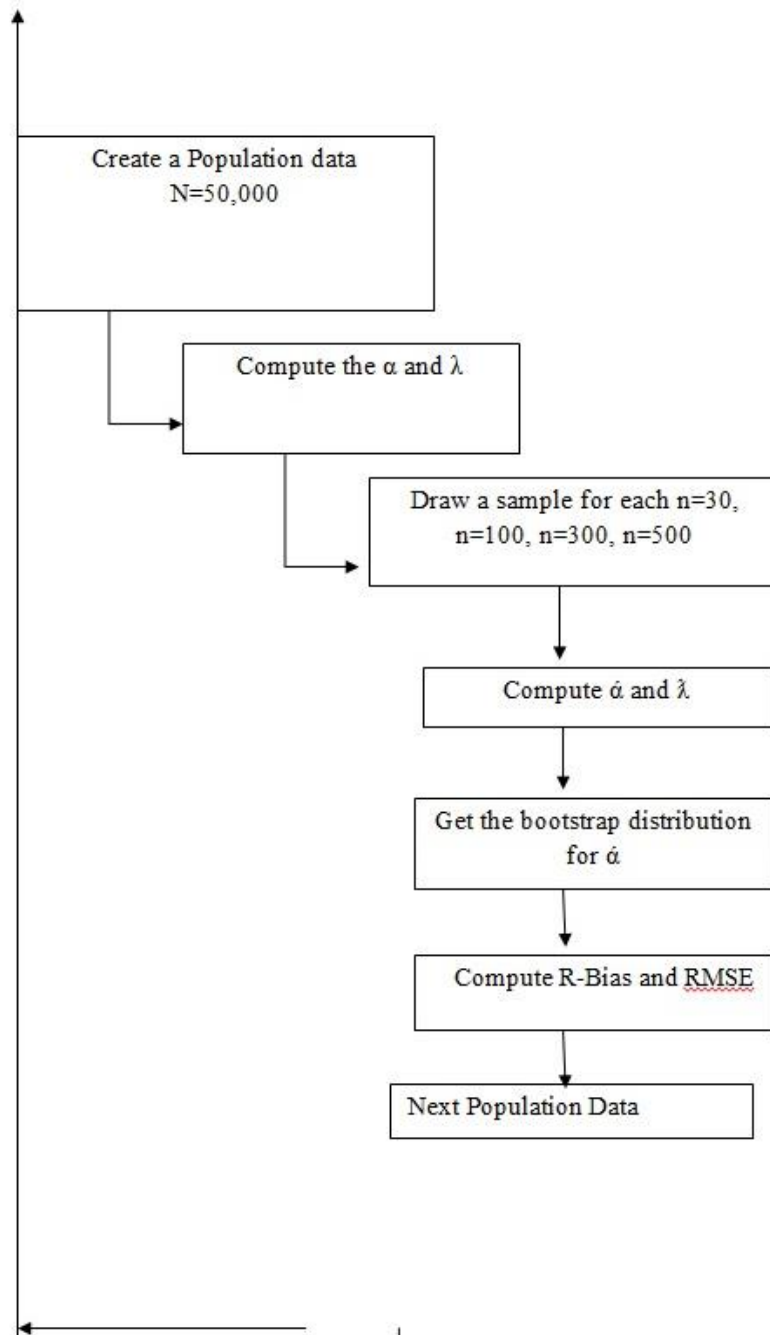


Figure 4-2: Algorithm construction

4.8. Findings

The results of the simulation were examined in two different ways. For the first approach, the lambda hat ($\hat{\lambda}$) was considered a continuous variable $1 \leq \hat{\lambda}_1 < \infty$. Thus the behaviour of

the R-RMSE and the R-Bias values of coefficient alpha estimates were observed each λ_1 value is different a sample sizes.

In the second approach, the λ_1 values were categorised as discrete variables in order to investigate the effects of both the sample size and the λ_1 on the performance of the estimator.

In the figure above, the R-Bias estimator increases with a decrease in sample size (n) and in the greatest eigenvalue. The exactness or accuracy of the coefficient (α) advanced or rather improved with larger sample sizes and higher values of the λ_1 . According to the values of the R-Bias and R-RMSE, when n=30, the magnitude of λ_1 had a critical role due to the robust estimator of the coefficient alpha. Otherwise, the suggested sample size for coefficient alpha is 300 in psychometric literature; it is seen in the research that when the sample size is three hundred (n=300), then the alpha (coefficient estimator) is biased for the small values of λ_1 . However, when n=500, whatever the value of λ_1 , the estimated coefficient alpha is more precise.

In the second approach, the λ_1 values were arbitrary categorised as small when, $\lambda < 3$, moderate when $3 \leq \lambda < 6$, and large when $\lambda \geq 6$. The R-Bias and the R-RMSE of the coefficient alpha estimates were treated separately as dependent variables in the design of ANOVA. In each analysis, the independent variables were sample size, levels of λ_1 , and number of items. The table below is a presentation of ANOVA results for R-Bias and R-RMSE, including the effect size estimates (ω) values.

Table 4:4: Analysis of variance results for the R-Bias and R-RMSE in the study

Analysis of variance results for the R-Bias and R-RMSE in the study									
Source	Bias as dependent variables					RMSE as dependent variables			
	df	SS	F	P	ω	SS	F	P	ω
<u>n</u>	3	3.45	2197.65	0.000	0.142	0.243	3042.95	0.000	0.186
<u>k</u>	15	0.04	5.55	0.000	0.002	0.002	4.84	0.000	0.002
<u>l</u>	2	2.19	2093.86	0.000	0.095	0.179	3359.96	0.000	0.114
<u>nxk</u>	45	0.08	3.32	0.000	0.004	0.004	3.15	0.000	0.004
<u>nxl</u>	6	1.69	539.45	0.000	0.075	0.112	697.68	0.000	0.095
<u>kxl</u>	27	0.31	22.21	0.000	0.015	0.028	39.10	0.000	0.026
<u>nxkxl</u>	81	0.42	9.81	0.000	0.020	0.030	14.07	0.000	0.028
-	-----	-----	-----	-----	-----	-----	-----	-----	-----

SS: Sum of squares; P: probability; ω : effect size estimations, n: Sample size; k: Numbers of item; l: Level of first eigenvalue

SS: Sum of squares; P: probability; ω : effect size estimations, n: Sample size; k: Numbers of item; l: Level of first eigenvalue

Considering the table above, all the main effects of interactions were significant. The largest effect size was the sample size ($\omega_n = 0.142$ for R-Bias and $\omega_n = 0.186$ for R-RMSE), followed by the level of eigenvalue ($\omega_l = 0.095$ for R-Bias and $\omega_l = 0.144$ for R-RMSE). As for the other interactions, the observation was that the sample size x level of the first eigenvalue interaction ($\omega_{nxl} = 0.075$ for R-Bias and $\omega_{nxl} = 0.095$ for R-RMSE). The remaining other effect sizes for the interactions were negligible but statistically significant. Instead of the post hoc analysis for interactive effects the graphical representation of the interaction between sample size and the level of the first eigenvalue for R-Bias is given as per the graph below:

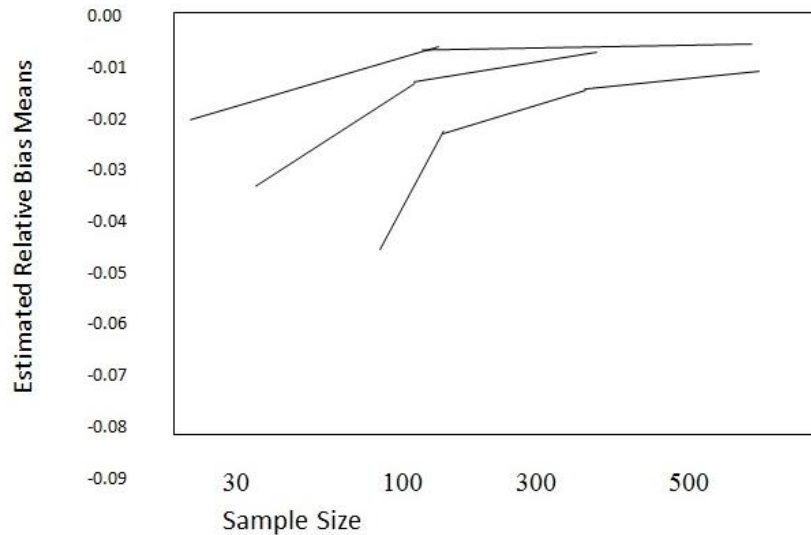


Figure 4-3: Estimated R-Bias means against sample size

The figure above shows the performance of sample coefficient alpha in levels of the first eigenvalue (the first curved line). Referring to the simulation outcome of the research when $\lambda_1 \geq 6$ (Level 3) the estimates of coefficient alpha are said to be unbiased for each sample size. However, the estimates of coefficient alpha are only unbiased for data sets where the eigenvalue is $3 \leq \lambda_1 < 100$. On the other hand, also, the estimates of coefficient alpha are unbiased if the size of the sample is greater than 300 when $\lambda_1 < 3$ (Level 1).

When the sample size is 500 or higher, the performance of the coefficient alpha estimator behaves independently from the biggest eigenvalue of the sample data set. When the sample size is smaller than 500, the execution of the alpha as a sample coefficient also depends on the first eigenvalue of the data set 4.

4.8.1 Lack of commitment to delivery timelines

Study participants reported that the team experiences resentment and hostility as a result of a team member not delivering to timelines. Analysis of the data revealed that 31% (54) strongly agreed with the statement that not meeting deadlines contributed to resentment within the team, whilst 69% (121) agreed, and 53% (93) strongly agreed that not meeting deadlines led to hostility within the team with 38% (67) stating they agreed with the statement.

In response to the statement, “Other team members have to do more work, when a team member does not meet deadlines”, 21% (31) strongly agreed, and 67% (118) agreed. The overall response on the statement, “When a team member does not meet development timelines, the team is affected negatively”, was an overwhelming 100% strongly agreeing.

4.8.2 Inattention to delivering quality software

Alarming, participants reported 100% strongly agreeing with the statement, “When a team member makes mistakes repeatedly, it affects the team negatively”. Participants were noticeably aware that delivering software with defects requiring another team member to fix, led to hostility, where 98% (171) agreed and 2% (4) strongly agreed. A high number of participants also observed resentment present in the team when mistakes were repeated, with 10% (18) strongly agreeing and 90% (157) agreeing. Some 74% (130) of study sample strongly agreed, and 26% (45) agreed with the statement, “The team is negatively affected when team members are negligent in delivering quality code”. Also, the study sample revealed that 95% (166) and 2% (4) strongly agreed or agreed with the statement that other team members have to do more work when a team member delivers bad quality code.

4.8.3 Lack of accountability

Analyses of the baseline quantitative data revealed that all of the study participants strongly agreed that team members who do not complete their task create hostility within the team. Some 91% (160) of the study sample strongly agreed that when a team member does not take accountability of their task, the team may be adversely impacted; 9% (15) agreeing. Most importantly, 97% (169) of study sample reported strongly agreeing that when a team member frequently blames other team members for non-delivery of tasks, it created a culture of resentment within the team. Lastly, 87% (152) of the study sample agreed that other team members have to do more work when a team member does not complete their tasks.

Lastly, according to the study sample responses, the presence of hostility and resentment within the team as a result of a team member’s lack of commitment to delivery timelines, inattention to delivering quality software, or inattention to accountability has compounded conflict and mistrust within the team. Some 97% (170) strongly agreed that when team

members are hostile there is a loss of trust within the team, and 59% (104) agreed that they noticed an increase of conflict within the team when there is a culture of resentment present.

CHAPTER 5:

Conclusions and Recommendations

5.1. Conclusions

The research was valid and it shows that the dysfunctionality of a team is directly proportional to lack of commitment to meeting the timelines, inattention to the delivery of quality software, and the avoidance of accountability.

In literature regarding reliability, the necessary minimum sample size for coefficient alpha is commonly suggested as 200, 300, or 500. According to this research, the least sample size for approximating factor alpha is directly dependent on the first level of the first or greatest eigenvalue obtained from the PCA. If the starting eigenvalue in the data sample is larger than six, then the coefficient sample alpha even though it is $n=30$, is an exceptionally robust estimator of the population factor alpha. If the starting eigenvalue in the data sample is between three and six, then the required least sample ($n=100$) will be enough of a coefficient sample for the unbiased estimator of coefficient alpha.

Considering the sample of 185 000, the study is relevant for the sample of $n=185$ in this case. However, there are limitations of the survey; one of the limitations is that the simulation is based on average data. According to Yuan (2002), the coefficient alpha estimate's asymptotic distribution is obtained on consideration of simple sampling distribution. The process is valid within a large class of abnormal distributions.

Several observations were derived from the literature review, which are further supported by the baseline findings of this research regarding the characteristics of a dysfunctional development teams.

First, the study data clearly documented that the lack of commitment to development timelines is a predictor of a dysfunctional software development team. For example, 100% of the participants strongly agreed that there is a negative impact on the team when a team member does not meet development timelines; with 31% agreeing and 53% strongly agreeing

that this type of behaviour introduces hostility and resentment within the team. Similar to these findings, 67% agreed that other team members have to carry more workload when a team member does not meet deadlines.

Second, the findings from this baseline study also revealed that inattention to delivering quality software is a predictor of a dysfunctional software development team. For example, 98% of study participants were in agreement that when a team member is not “carrying their load” there is evidence of bitterness within the software development team. Also, 100% of the study participants strongly agreed that by not completing their development tasks, a team member introduces hostility within the software development team. These findings strongly suggest that software development teams are significantly impacted by the inability of a developer within the team to deliver quality software and this supports the need to incorporate materials skills training and increase the knowledge of the software developer as essential components in the development of performing software development teams, especially where there is evidence of hostility and resentment within the team.

Third, the findings from this baseline study further show that avoidance of accountability is a predictor of a dysfunctional software development team. For example, 100% of study participants reported that when a team member does not complete their tasks, hostility enters into the team. Further, a majority (97%) of study participants felt that when a developer frequently blames other team members for non-delivery of tasks, a culture of resentment is evident within the team, and 91% participants agreed that when a team member does not take accountability of their task, the team may be adversely impacted. Such findings have significant implications for the team, as team dynamics will be weakened, leading to a less cohesive team unit.

These findings also lend credence to the researcher’s model, and are further supported by prior findings from other researchers on the five attributes of a dysfunctional team.

The most important contribution of this study is the fact that three different sets of predictors emerged as characteristics of a dysfunctional software development team. The similarity in the predictors revealed that when these three characteristics are present in a software

development team, the team will not be able to perform at acceptable levels. Specifically, there will be the underlying presence of resentment, conflict, breakdown in trust, and pressure on other team members. For the purpose of this research, the three predictors of a dysfunctional software development team have been identified and future focus should be on creating measures to effectively address these three characteristics.

The researcher has clearly demonstrated that the baseline findings from this research may have a significant impact on a better understanding of the characteristics of dysfunctional software development teams. Future research can now be focused on further developing and potentially supporting the researcher's findings.

The researcher has also found that when these characteristics are present within a software development team, a culture of resentment and hostility exists, which further leads to an increase in mistrust and conflict within the team. Some 59% strongly agreed that there is an increase of conflict within the team when the team has a culture of resentfulness, and 97% strongly agreed that when team members are hostile, there is a loss of trust within the team. Nevertheless, there is a need for additional research to further explore the characteristics to assist software development teams in negating these characteristics so that they may reach their full potential as a team.

5.2. Recommendations: Overcoming the three predictors of dysfunctional software development teams

5.2.1 Inattention to delivering quality software

The software development team should **spend time defining what the team understands as "quality"**. To hold individuals accountable for delivering quality software, there has to be a shift within the team from different meanings of "quality" to a solid definition of quality that all developers within the team understand and agree to, therefore creating buy-in from every team member. By defining quality, each team member will understand the need for attention

to value-added features, detail, and quality across the software development lifecycle. Many different criteria can be set by the team to define quality, which extend further than producing defect free code, for example 1) the solution must meet stakeholder requirements, 2) the solution should be user friendly and usable, 3) the solution should not introduce security issues, 4) the solution should be scalable, 5) the code should be reliable and easy to maintain, and 6) the solution should be easy to monitor or extend (Bessin, 2004).

Every developer within the team **should strive against losing time reworking, re-factoring, and rewriting software, focusing on achieving the best quality software** during development, integration, and testing the software development processes. If a developer is developing with the aim to “cut” quality in order to improve development speed, reduce costs, or add functionality, the team should call this out immediately, as this misconception is far from the reality of achieving quality code. Meskimen states, "There's never time to do it right, but always time to do it over." In reality, however, it is the delivery of quality software that enables software teams to deliver more tasks or projects timely, at a reduced cost, and with additional features. If each software developer focus on quality, all defects will be identified during the development process, thereby eliminating the time and effort required in involving other team members to find and fix defects in production once code has been deployed (Bessin, 2004).

To ensure quality delivery from individuals within a software development team, **technology leaders managing software development teams should have a quality improvement mindset**. This should include inculcating a culture of quality throughout the team, reducing the strain on management to make difficult decisions to accept defects deployed into production, setting demanding processes (team responsibility matrices, objective metrics etc.) to enforce predictable quality, and re-scoping requirements as software developers will then focus their attention on producing accurate scope, estimations and schedules, and will deliver according to stakeholder specifications.

Software developers need to **adopt the most appropriate processes and tools** to produce according to the requirements set out by stakeholders. When software team members notice the need for a team member to attain further knowledge via training, they should be referred

to management to ensure that the appropriate actions are taken to assist the team member to meet the requirements of delivering quality software. Where the team member themselves identifies gaps in their knowledge resulting in the delivery of poor quality code, they should seek the appropriate guidance to close the identified gaps.

The Meta Group reports that up to 80% of the issues leading to poor quality code which do not meet customer requirements can be traced to a poor understanding of requirements. It is imperative during the analysis phase of the software development cycle that developers **raise the issue if the requirements provided are not clear or are incomplete**. This will avoid time and cost wastage in rewriting code where requirements were assumed, but which the stakeholder rejects as the solution does not meet their perception of the end product (Bessin, 2004).

During the design phase of the software development cycle, the developers' **focus should be on the quality of the architecture**. If the architecture is not designed well, it may lead to the software being fragile when future modifications are considered and it will lack scalability. Software developers should **spend time detecting and resolving any architectural deficiencies, and protecting the architecture's integrity and flexibility**, so as to not involve more team members during the later stage of the cycle or after the solution is deployed to assist with debugging, thus adding a burden on fellow colleagues' workload.

During the design phase of the software development cycle, developers **focus on developer-led testing and analysis**, meaning building tests prior to building code to reduce risk and increase the predictability of delivering quality software. The software team should agree on the "basic" guidelines on the definition of code passing tests. This type of testing done by developers, prior to code moving to the testing team for testing, will assist in identifying and fixing defects that would ordinarily be identified by testers thus wasting time during the test cycle awaiting fixes from the developers.

Delivering **quality** software should be viewed as not only being **the responsibility** of each software developer but **the entire development team**. Software development teams must focus on establishing traceability and integrating workflows, whilst simplifying communication within the team.

5.2.2 Lack of commitment to development timelines

Software development teams should **agree on the method where estimations and timelines** will be conducted to establish a unified way of providing estimates and timelines; assist stakeholders to set realistic target goals, deadlines, and associated progress tracking; and hold each other accountable to meeting the timelines according to the committed estimations. Software developers should guard against providing underestimation as a result of not having a thorough knowledge of the technical complexities of the application or having no idea about the technical challenges involved in completing the task which eventually leads to the task not being delivered according to the timelines committed to (Soni & Acharya, 2017).

To avoid the scenario of underestimation, the software development team should **identify the software developer that has ample knowledge** to complete the task and is familiar with the application. Where a software developer has no knowledge of the impacts of a requirement, the software developer should not provide estimates until he or she has spent time with a more experienced developer to understand how a particular module or feature would be impacted. All software developers, irrespective of experience or knowledge, should conduct a system level design to understand the interfaces among other modules to gain as much insight into the work involved, prior to providing an estimation of timelines for delivery of code and to avoid deviating from delivering as per requirements.

While software developers are working on providing estimation and timelines, they must be **mindful to accommodate not only unit testing but system level or integration testing** that will be done on their module or modules. A further consideration for software developers is to allocate time for beta releases if the stakeholder requests for the product to be validated by users or customers during development as part of the process of presenting a workable release to users or customers, and incorporating the suggestions or feedback into the final software version does demand a significant amount of time from the software developer (Soni & Acharya, 2017).

The software development management team must **take responsibility to upskill a new software developer** that joins the team, and ensure that the individual is not attending

estimation meetings by themselves, and recognise that he or she may require time to understand the scope of the application, as well as the requirements to keep pace with other team members. A technology leader must take into account that a learning curve is needed for new team members joining the team who might not have all the specific technical competencies required from the onset. Therefore attempting to place the new comer into a project or task that requires immediate timelines and estimations may result in the provision of incorrect estimation.

In an age of globalisation and subcontracting, it has become **the new norm for a development team to be distributed over diverse geographical regions**. The software developer needs to take into account different time zones and working cultures, especially where tasks are separated between software development team members, to avoid providing estimations that may be completely incorrect if no consideration has been given to the work culture of the region where the team member resides. It is important for teams working in different time zones, or working from different geographical locations, to agree on the definition of “urgent”, “high priority task” changes with tolerance to culture and region of each team member (Soni & Acharya, 2017).

The software development team also needs to **spend time understanding the manner in which the team members communicate**, and consider differences and avoid team members being embarrassed to report issues faced during execution that may delay the initial estimations provided, and may lead in additional workload for team members. The software developer must bring important issues to the notice of the team members and ensure timely escalation if the agreed timeline will not be met to provide ample time for a plan B.

Software developers should only **commit to providing timelines and estimations once clear requirements and specifications have been received**. Firstly, if the software developer chooses to provide estimations on requirements that are unclear at the time of estimation, it may result in a delay in the completion of the task. Secondly, if the software developer estimates a requirement which is not yet matured or still evolving, frequent changes in the specifications will change the requirement for development and the developer will continue to work on an incomplete requirement, resulting in rework that will render the initial

timelines incorrect. Thirdly, if a software developer receives new requirements which are added as the task evolves towards completion, there should be a call made by the software developer to revisit the initial timelines provided and ensure work is not continued to incorporate the new requirements without amending the initial estimations to avoid delay in delivering the completed solution (Soni & Acharya, 2017).

Software teams should **ensure that estimations include the effort required to improve system performance, memory issues, user interface quality and usability** to avoid addressing these types of technology changes towards the end of the task that may require design changes and additional effort which mean not delivering the task within the time estimated.

Software development **teams should hold each other accountable** and not accept poor programming skills, inability to adapt to modern programming practices and having ad hoc development processes which lead to higher number of defects, eventually requiring the assistance of other team members to fix resulting in slippage in timelines committed to. The team should also hold the software developer responsible for the completion of tasks which were not completed from previous milestones due to inefficiency, to avoid increasing the load on the team and thus not meeting the deadline (Soni & Acharya, 2017).

Globally, software development teams are faced with the challenge of on-time delivery. To have complete control over the estimations provided, it is imperative for the software developer **to identify the elements in the development cycle that may be impacted**, and to have **open and honest communication** within the team prior to providing estimations, thereby making the best decisions to avoid missing development timelines.

5.2.3 Avoidance of accountability

Initially, an open session, where all software development team members are mandated to attend, should be set up to **clarify what the team is required to achieve, who is responsible for delivery of what tasks and actions, and formulate team behaviours** to achieve success to avoid the team not having a clear set of expectations that they can hold one another accountable for. As a leader of the software development team, it is equally important to define what the **expectation from management** is for the team, and this play an instrumental

role in creating a culture of accountability by encouraging team members to regularly communicate about how team members are doing against the agreed objectives and standards set by the team.

The technology manager should **clearly articulate the duties, roles and responsibilities of those within the software development team, clearly documenting these required skills/roles, what the expectations and dependencies are, and any key performance indicators that management wishes to achieve.**

The manager must be willing to **set and lead by example**, confront difficult issues, and demand personal accountability from software development team members, thus embedding the concept of accountability into the team culture. By doing so, the team is given more autonomy and the leader has more time to focus on other demanding issues (Upguard, 2017).

All software development team members should be **involved in the planning and tracking of projects and tasks**. On a daily basis, every team member's performance should be known to the entire team, not to put negative pressure on individuals, but a poor performer would be under massive peer pressure to review their own performance. This also leads to other team members being given an opportunity to offer their assistance and further ensures that the team is working on its commitments, creating the right focus, reviewing if everything is being done as required, and all avenues are explored to meet the deliverables. Role models (key performers) exist within the team to show other team members a living, breathing example of what each individual should aspire to be (Varma, 2014).

Software development teams should **foster a team culture** where developers can share, explain, and justify the actions taken or not taken, the decisions made or not made, and all associated considerations that have led to the actual result, thereby creating a transparent, focused, "doing the best you can", responsible team in a sharing environment; one where team members can call upon the team to help if they are faced with difficult tasks, are creatively challenged, or experience unpredictable work without the fear of alienation or burn-out (Verheyen, 2014).

Creating a software development **team incentive program** will self-motivate developers to reach their highest potential and accountability. Software development teams are compensated by reaching or exceeding the measurable, predetermined performance-based incentive objectives. Financial incentives could include employee share options, merit pay, profit sharing, commissions, and bonuses. Non-financial incentives could include praise for good work, further career opportunities and recognition. Measurable performance-based incentive plans will give developers control, holds them accountable for performance, and provides them with control over their resources to complete the goal and create an innovative drive within the team (Crampton, 2011).

From an Intellectual Merit perspective, the characteristics of a dysfunctional software development team need to be understood in order to explain why the software development team is not performing at its peak or is impeded in becoming a top performing team. This research makes a contribution to this effort: the advancement of identifying three characteristics that, if present in a software development team, contribute to the team being dysfunctional. This complementary contribution may ultimately lead to a more nuanced understanding of why software development teams are dysfunctional.

From a broader Impacts perspective, the research may lead to new training and mentoring opportunities specifically designed to address the three identified characteristics. The findings and knowledge generated as part of this research have the potential to transform an information technology software development leader's conceptualization and measurement of software development teams.

These research baseline findings can support future research into conflict and loss of trust emanating as a result of the three characteristics of a dysfunctional software development team

REFERENCES

- Adams, C.M. & Forsyth, P.B. (2010). The nature and function of trust in schools. *Journal of School Leadership*, vol. 19, pp. 126-152.
- Armor, D.J. (1974). *Theta reliability and factor scaling*. In Costner, H.L. (Ed.). *Sociological Methodology*. San Francisco: Jossey-Bass.
- Attarzadeh, I. (2011). Software Development Cost and Time Forecasting Using a High Performance Artificial Neural Network Model. *CCIS*, vol. 134, pp. 17-28.
- Barnard, R. (2011). Post-Apartheid Modernism and Consumer Culture. *Modernist Cultures*, 6(2), pp. 215-244.
- Barnes, D.R. (1992). *From communication to curriculum*. Portsmouth, NH: Boynton/Cook Publishers.
- Belbin, R.M. (2010). *Management Teams: Why they succeed or Fail*. 1st edition. UK: Butterworth-Heinemann.
- Blau, G. & Andersson, L. (2005). Testing a measure of instigated workplace incivility. *Journal of Occupational and Organizational Psychology*, 78(4), pp. 595-614.
- Bloch, M., Bumberg, S. & Laartz, J. (2012). *Delivering large-scale IT projects on time, on budget, and on value*. Available from: <http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value>. (Accessed on:12/07/2015)
- Boehm, B., Horowitz, E., Madachy, R. & Abts, C. (2000). Future Trends, Implications in Cost Estimation Models. *Crosstalk*, pp. 3-7.
- Boehm, B.W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), pp. 61-72.
- Box, G.E.P. & Muller, M.E. (1958). A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29: 610–611.

Branner, J. (2005). *Mixed Methods Research: A Discussion Paper*. Economic and Social Research Council National Centre for Research Methods Review Papers. Available from: <http://www.ncrm.ac.uk>.

Browne, J.A., Covington, B.G. & Davila, Y. (2004). Using information technology to assist in redesign of a fall prevention program. *Journal of Nursing Care Quality*, 19(3), pp. 218-225.

Busseri, M.A. & Palmer, J.M. (2000). *Improving teamwork: the effect of self-assessment on construction design teams*. 1st edition. Thousand Oaks, CA: Sage.

Cameron, E. & Green, M. (2015). *Making sense of change management: a complete guide to the models, tools and techniques of organizational change*. USA: Kogan Page Publishers.

Carpenter, M., Bauer, T. & Erdogan, B. (2012). *Principles of Management and Organizational Behaviour*. 1st edition. USA: Flat World Knowledge.

Chin-Min, H., Shi-Jer, L., Chi-Chang, L. & Pei-Ling, W. (2014). Identification of dysfunctional cooperative learning teams and troubled individuals. *British Journal of Educational Technology*, 45(1), pp. 112-140.

Collins, H. (2011). *Creative Research: The Theory and Practice of Research for the Creative Industries*. USA: AVA Publications.

Conrow, E.H., Hantos, P., Holt, G., Evans, M., Segura, C., Doherty, F. & Glazewski, S.R. (2005). *CrossTalk: The Journal of Defense Software Engineering*, vol. 18, no. 2.

Cortina, J.M. (1993). What is coefficient alpha? An examination of theory and applications. *Journal of Applied Psychology*, 78 (1), 98-104.

Craemer, M. (2012). *Group Accountability for Effective Teamwork*. Available from: <http://blog.seattlepi.com/workplacewrangler/2012/11/19/group-accountability-for-effective-teamwork/>. (Accessed 19/01/2015).

Crano, W.D. & Brewer, M.B. (1973). *Principles of research in social psychology*. New York: McGraw-Hill.

- Creswell, J.W. (2003). *Research Design: Qualitative and Quantitative and Mixed Method Approaches*. Thousand Oaks, CA: Sage Publications.
- Cronbach, L.J. (2004). My current thoughts on coefficient alpha and successor procedures. *Educational and Psychological Measurement*, 64, 391-418.
- Davenport, T.H. (1998). Putting the enterprise into the enterprise system. *Harvard Business Review*, 76(4).
- Davis, A.M. (1995). *201 principles of software development*. USA: McGraw-Hill, Inc.
- Dean, B.V. (1963). *Operations research in research and development*. New York: Wiley.
- DeMarco, T. (1995). *Why Does Software Cost So Much?* New York: Dorset House Publishing.
- Ditkoff, M. (2015). *The power of pause*. Available from:
http://www.ideachampions.com/weblogs/archives/2015/08/the_power_of_pa.shtml
(accessed 9/11/2015).
- Dumaine, B. (1994). The trouble with teams. *Fortune*, vol. 130:5. pp. 86 (5).
- Evans, M.W., Abela, A.M. & Beltz, T. (2002). Seven characteristics of dysfunctional software projects. *Crosstalk*, 15(4), pp. 16-20.
- Forsyth, D.R. (2010). *Group Dynamics*. (5th Ed.). Belmont, CA: Wadsworth Cengage Learning.
- Galorath, D.D. & Evans, M.W. (2006). *Software sizing, estimation, and risk management: when performance is measured performance improves*. USA: CRC Press.
- Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J. & Hornik, K. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5(10), p. 1.
- Grenny, J. (2014). *The Best Teams Hold Themselves Accountable*. USA: Berrett-Koehler Publishers.
- Hackman, J.R. (2006). The Five Dysfunctions of a Team: A Leadership Fable. *The Academy of Management Perspectives*, 20(1), pp. 122-125.

Hall, E.M. (1998). *Managing risk: Methods for software systems development*. USA: Pearson Education.

Hamlin, J.L. (2008). *Team effectiveness: A validation study of Lencioni's five functions of a team*. Order No. 3301963, University of La Verne. Available from: <http://0-search.proquest.com.woodhous.aquinas.edu/docview/304401931?accountid=8340>. (304401931). (Accessed 20/09/2016).

Hanlan, M. (2004). *High Performance Teams: How to Make Them Work*. USA: Praeger.

Harris, J.M., Butcher, A.M., Morton, D.A., Satterfield, J.C. & Denney, J. (2012). *Command user interface for displaying selectable software functionality controls*. U.S. Patent 8,255,828.

Hayes, P. (2011). *Leading and coaching teams to success: the secret life of teams*. UK: McGraw-Hill Education.

Hoevemeyer, V. (1993). *How effective is your team?* 4th edition. USA: Newtown Square.

Hsiung, C.M. Luo, L.F. & Chung, H.C. (2014). *Early identification of ineffective cooperative learning teams*. Available from: <https://www.deepdyve.com/lp/wiley/early-identification-of-ineffective-cooperative-learning-teams-boERlznRe9> (Accessed 13/02/2016).

Humphrey, W.S. (1998). Three dimensions of process improvement. *The Journal of Defense Software Engineering*, 14, pp. 39-72.

Humphrey, W.S., Webb, D.R., Seshagiri, G., Starrett, E., Lipke, W., Putnam, L.H. & Wegner, K.R. (2000). *CrossTalk. The Journal of Defense Software Engineering*, vol. 13, Number 6.

Johnson, R.B. & Christen, L. (2008). *Educational Research Quantitative, Qualitative and Mixed Method Approaches*. 3rd Edition. Thousand Oaks, CA: Sage.

Johnson, S.D., Suriya, C., Yoon, S.W., Berrett, J.V. & La Fleur, J. (2002). Team development and group processes of virtual learning teams. *Computers & Education*, 39(4), pp. 379-393.

Jones, C. (1994). *Assessment and control of software risks*. USA: Yourdon Press.

Katzenbach, J.R. & Smith, D.K. (1992). *Wisdom of Teams*. 3rd Edition. Mass: Harvard Business School Press.

Katzenbach, J.R. & Smith, D.K., 2005. The discipline of teams. *Harvard Business Review*, 83(7), p. 162.

Keshari, K.R., Tsachres, H., Iman, R., Delos Santos, L., Tabatabai, Z.L., Shinohara, K., Vigneron, D.B. & Kurhanewicz, J. (2011). Correlation of phospholipid metabolites with prostate cancer pathologic grade, proliferative status and surgical stage—impact of tissue environment. *NMR in Biomedicine*, 24(6), pp. 691-699.

Kets de Vries, M.F.R. (2011). *Build and Manage High Performance Teams: The How to Guide*. USA: Control Addison-Wesley Publishing Co.

Kline, P. (1986). *A handbook of test construction: Introduction to psychometric design*. New York: Methune & Company.

Kline T.J.B. (1999). The team player inventory: Reliability and validity of a measure of predisposition towards organizational team-working environments. *Journal for Specialists in Group Work*, 24, pp 90-120.

Richardson, T. (2015). *The responsible leader: Developing a Culture of Responsibility in an Uncertain World*. USA: Kogan Page.

Kohn, S.E. (2008). *6 Habits of Highly effective Teams*. USA: ReadHowYouWant Publishers.

LaMarca, N. (2011). *The Likert Scale: Advantages and Disadvantages*. Available from: <https://psyc450.wordpress.com/2011/12/05/the-likert-scale-advantages-and-disadvantages/> (Accessed 13/06/2015).

Langan-Fox, J., Cooper, C.L. & Klimoski, R.J. (Eds.). (2007). *Research companion to the dysfunctional workplace: Management challenges and symptoms*. USA: Edward Elgar Publishing.

Latham, G.P. & Pinder, C.C. (2005). Work motivation theory and research at the dawn of the twenty-first century. *Annu. Rev. Psychol.*, 56, pp. 485-516.

Lencioni, P. (2002). *The five dysfunctions of a team: A leadership fable*. San Francisco: Jossey Bass.

Lencioni, P. (2005). *Overcoming the five dysfunctions of a team: A field guide for leaders, managers, and facilitators*. (Vol. 16). USA: John Wiley & Sons.

Lencioni, P.M. (2007). *The five dysfunctions of a team: Participant Workbook*. (Vol. 8). USA: John Wiley & Sons.

Levi, D. (2015). *Group dynamics for teams*. USA: Sage Publications.

Liu, Y. & Zumbo, B.D. (2007). The Impact of Outliers on Cronbach's Coefficient Alpha Estimate of Reliability: Visual Analogue Scales. *Educational and Psychological Measurement*, 67, 620-634.

MacCallum, R.C., Widaman, K.F., Zhang, S. & Hong, S. (1999). Sample size in factor analysis. *Psychological Methods*, 4, 84-99.

Mackin, D. (2007). *Team Building Tool Kit: Tips and Tactics for Effective Workplace Teams*. 2nd edition. USA: AMACOM.

Mao, X. & Yuan, C. (2006). *Stochastic differential equations with Markovian switching*. London: Imperial College Press.

Mark, W., Tyler, S., McGuire, J. & Schlossberg, J. (1992). Commitment-based software development. *IEEE Transactions on Software Engineering*, 18(10), pp. 870-885.

Massey, T. (2010). *The Ten Commandments for Building High Performance Teams*. USA: Robert Reed Publishers.

Matta, N.F. & Ashkenas, R.N. (2003). *Why Good Projects Fail Anyway*. Mass: Harvard Business Publishing.

Mendenhall, J.A., Hearn, D.R. & Lencioni, D.E. (2002). *Comparison of the EO-1 Advanced Land Imager Performance with the Landsat Data Continuity Mission Specification (No. PR-EO-1-8)*. Mass: Massachusetts Inst. of Tech Lexington Lincoln Lab.

- Mosakowski, E. (1997). Strategy making under causal ambiguity: Conceptual issues and empirical evidence. *Organization Science*, 8(4), pp. 414-442.
- Moss Kanter, R. (2011). *How Great Companies Think Differently*. Available from: <https://hbr.org/2011/11/how-great-companies-think-differently>. (Accessed 8/09/2016).
- Mason, M.K. (2015). *Knowledge Management: The Essence of the Competitive Edge*. Available from: <http://www.moyak.com/papers/knowledge-management.html>. (Accessed 6/01/2017).
- Naghavi, M., Libby, P., Falk, E., Casscells, S.W., Litovsky, S., Rumberger, J., Badimon, J.J., Stefanadis, C., Moreno, P., Pasterkamp, G. & Fayad, Z. (2003). From vulnerable plaque to vulnerable patient: A call for new definitions and risk assessment strategies: part I. *Circulation*, 108(14), pp. 1664-1672.
- Noor, I., Joyner, T. & Martin Jr. R.J. (2002). *Challenges of implementing risk management processes*. AACE International Transactions, pp. RI41.
- Nunnally, J.C. & Bernstein, I.H. (1994). *Psychometric theory*. (3rd Edition). New York: McGraw-Hill.
- O'Donnell, K. (2011). On leadership and teamwork: a narrative and a model. *Journal of Psychology and Theology*, 39(1), 73+.
- Ottenssooser, A.B. (1999). *Determination of software functionality*. Available from <https://www.google.tl/patents/US5905856>. (Accessed 14/09/2015).
- Pajerek, L. (1999). *The Age of Process: Applying a Historical Perspective to the Present Day*. In INCOSE International Symposium, vol. 9, no. 1, pp. 910-915.
- Parker, G.M. (1990). *Team Players and Teamwork: The New Competitive Business Strategy*. 2nd Edition. San Francisco: Jossey-Bass Inc.
- Patrick, L. (2002). *The Five Dysfunctions of a Team: A Leadership Fable*. San Francisco: Jossey-Bass.

Peterson, R.A. (1994). A meta-analysis of Cronbach's coefficient alpha. *Journal of Consumer Research*, 21, 381-391.

Poppendieck, M. & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit*. USA: Addison-Wesley.

Poulin, L.A., Reifer, D., Craver, J., Ellis, M., Strickland, D. & Kimmerly, P.J., Software Technology. *Crosstalk*, 801, pp. 586-0095.

Putnam, L.H. (1997). *Industrial Strength Software: Effective Management Using Measurement*. California: IEEE Computer Society Press, pp. 100-150.

Ray, D., & Elder, D. (2007). Managing Horizontal Accountability. *Journal for Quality & Participation*, 30(4), 24-28.

Raykov, T. (1997). Scale reliability, Cronbach's coefficient alpha, and violations of essential tau-equivalence with fixed congeneric components. *Multivariate Behavioral Research*, 32, 329-353.

Reed, M.F., Bartholomay, R.C. & Hughes, S.S. (1997). Geochemistry and stratigraphic correlation of basalt lavas beneath the Idaho Chemical Processing Plant, Idaho National Engineering Laboratory. *Environmental Geology*, 30(1-2), pp. 108-118.

Richter, H., Abowd, G.D., Geyer, W., Fuchs, L., Daijavad, S. & Poltrock, S. (2001). *Integrating meeting capture within a collaborative team environment*. In proceedings of the International Conference on Ubiquitous Computing, pp. 123-138. Heidelberg, Berlin: Springer.

Robbins, H. & Finley, M. (2000). *The new why teams don't work: What goes wrong and how to make it right*. USA: Berrett-Koehler Publishers.

Sale, J.E.M. (2002). Revisiting the Quantitative-Qualitative Debate: Implications for Mixed-Methods Research. *Quality & Quantity*, 36:43-53.

- Salegna, G.J. & Goodwin, S.A. (2005). Consumer loyalty to service providers: an integrated conceptual model. *Journal of Consumer Satisfaction, Dissatisfaction and Complaining Behavior*, 18, p. 51.
- Savery, J.R. (2015). *Overview of problem-based learning: Definitions and distinctions*. Essential readings in problem-based learning: Exploring and extending the legacy of Howard S. Barrows. West Lafayette: Purdue University Press, 12, pp. 5-15.
- Scott, J.E. & Kaindl, L. (2000). Enhancing functionality in an enterprise software package. *Information & Management*, 37(3), pp. 111-122.
- Segall, D.O. (1994). The reliability of linearly equated tests. *Psychometrika*, 59, 361-375.
- Stahl, G. (2006). *Group Cognition: Computer Support for Building Collaborative Knowledge (Acting with Technology)*. Cambridge: MIT Press, pp. 12-20.
- Teddlie, C. & Tashakkori, A. (2009). *Foundations of Mixed Methods Research: Integrating Quantitative and Qualitative Approaches in the Social and Behavioural Sciences*. Los Angeles: Sage.
- Tjosvold, D. (1986). *Working Together to Get Things Done*. 5th Edition. USA: Lexington Books.
- Trimble, S. & Rottier, J. (1998). *Assessing team performance*. Available from: <http://files.eric.ed.gov/fulltext/ED422305.pdf>. (Accessed 10/09/2016).
- Tyler, W.M.S. & Schlossberg, J.M.J. (2002). *Commitment-Based Software Development*. California: IEEE Computer Society Press.
- Valkenburg, R. & Dorst, K. (1998). The reflective practice of design teams. *Design Studies*, 19: 249–271.
- Van Zyl, J.M., Neudecker, H. & Nel, D.G. (2000). On the distribution of the maximum likelihood estimator of Cronbach's alpha. *Psychometrika*, 65, 271-280.
- Voo, K. (2011). *Build and Manage High Performance Teams: The How-to Guide*. Kindle Edition.

Trochim, W.K. (2006). *Likert Scaling*. Available from: <https://www.socialresearchmethods.net/kb/scallik.php>. (Accessed 22/08/15).

Yuan, K., & Bentler, P.M. (2002). On robustness of the normal-theory based asymptotic distributions of three reliability coefficient estimates. *Psychometrika*, 67(2), 251-259.

Yuan, K., Guarnaccia, C.A. & Hayslip, B. (2003). A study of the distribution of the sample coefficient alpha with the Hopkins Symptom Checklist: Bootstrap versus asymptotics. *Educational and Psychological Measurement*, 63, 5- 23.

APPENDICES

Appendix A: RESEARCH QUESTIONNAIRE

1. Team is negatively affected when team members are negligent in delivering quality code?
2. Other team members have to do more work, when team member delivers bad quality code
3. When a team member does not take accountability of their task, team may be adversely impacted
4. The team member's behaviour is causing resentment within the team
 - Not meeting deadlines
 - Delivering software with defects that requires another team member to fix
 - Frequently blaming other team members for non-delivery of tasks
5. The team member's behaviour is causing hostility within the team
 - Not meeting deadlines
 - Delivering software with defects that requires another team member to fix
 - Not completing their development task
6. Other team members have to do more work, when team member does not meet deadlines
7. When a team member does not "carry their load" , leads to bitterness within the team
8. When a team member makes mistakes repeatedly , affects the team negatively
9. Other team members have to do more work, when team member does not complete their task
10. When a team member does not meet development timelines, team is affected negatively
11. When team members are resentful , there is an increase of conflict
12. When team members are hostile, there is a loss of trust within the team