

https://dspace.org/

A quantitative study into the perceived differences in expert judgement on factors that influence software development outcomes

Van der Linden, Anthony Cornelis

2018

https://hdl.handle.net/10539/27002

Van der Linden, Anthony Cornelis (2018) A quantitative study into the perceived differences in expert judgement on factors that influence software development outcomes, University of the Witwatersrand, Johannesburg, https://hdl.handle.net/10539/27002

Downloaded from a DSpace repository. https://dspace.org/

A quantitative study into the perceived differences in expert judgement on factors that influence software development outcomes

MSc(Eng) Dissertation

by

Anthony v.d. Linden

429773

University of the Witwatersrand

School of Electrical and Information Engineering

August 2018

Abstract

Why do some software development initiatives fail while others succeed? In most cases the answers to these questions are based on the perspectives of experts rather than measurement and empirical data collection. Are the views of these experts consistent, or do they differ? This is important because many innovations in software development methodologies are based on a response to the perspectives of different groups of experts.

The research presented in this dissertation tests whether different groups of experts had different perspectives on the outcomes of software development projects. The research methodology was guided by a quantitative design using objectivism as an epistemology and positivism as a theoretical framework. It included survey research and to facilitate generalisation of the overall result, collected data from a sample size of 384 participants at a 5% margin of error.

The research found that there is a statistically significant difference between experts in various roles and having different levels of experience. It concludes that expert judgement with respect to the outcomes of software development projects contained several cognitive biases and suggests that experts and organisations alike should consider adopting measurement and empirical data collection techniques to evaluate the value of their current practices before injudiciously adopting new methodologies.

Keywords

Software Crisis, expert judgement, cognitive bias, cognitive theory, methodology evolution, software failure, software success

Declaration by Students

I, Anthony v.d. Linden (Student number: 429773) am a student registered for Master of Science in Engineering (MScEng) in the year 2017. I hereby declare the following:

- I am aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong.
- I confirm that ALL the work submitted for assessment for the above course is my own unaided work except where I have explicitly indicated otherwise.
- I have followed the required conventions in referencing the thoughts and ideas of others.
- I understand that the University of the Witwatersrand may take disciplinary action against me if there is a belief that this is not my own unaided work or that I have failed to acknowledge the source of the ideas or words in my writing.

Signature: _____ Date: _____

Personal acknowledgement

Foremost, I would like to express my sincere gratitude to my supervisor, Professor Barry Dwolatzky, for his continued support not just during this research study but throughout my entire professional career. Your patience, motivation, enthusiasm and wealth of knowledge helped shape many individuals such as myself. Your looming retirement will leave a gap that cannot be filled, but your name will live on not just in research reports such as this one, but through the immense impact, you have had on the industry and the professionals you served. It was an honour to have you as a supervisor and mentor and I would like to wish you well with your future.

Besides my supervisor, I would like to thank Mr John Bielovich for his encouragement, insightfulness, tough questions and motivation. Your guidance and voice of reason were crucial and a driving factor for completing this work.

Lastly, my family and friends. Thank you for your undeniable support and understanding of the many missed engagements and interactions.

Contents

Contents ..... **iv**

Table of Figures ..... **vii**

Readers guide ..... **ix**

1 Introduction **1**

1.1 Background 1

1.2 Problem statement 3
1.2.1 Problem Context 4

1.3 Purpose of the study 6

1.4 Significance of the study 6
1.4.1 Reset focus on formality 7
1.4.2 Lack of engineering maturity 8

1.5 Definition of key terms 10
1.5.1 Influence 10
1.5.2 Internal variability 10
1.5.3 Political impacts 11

1.6 Scope and delimitations of the study 11
1.6.1 Research aspect 11
1.6.2 Research components 12
1.6.3 Methodology 13
1.6.4 Geographic delimitations 13

1.7 Summary 14

2 Literature review **16**

2.1 A maturing software crisis 16
2.1.1 Summary 23

2.2 Software Crisis 26
2.2.1 Then and now 26
2.2.2 Is the software industry in crisis? 29
2.2.3 Summary 31

2.3 Software development outcomes 32
2.3.1 Measurement 32
2.3.2 Success versus failure 35
2.3.3 Failure Categorization 44

2.3.4 Factors for software development failure 46
2.3.5 Summary....57
2.4 The Human factor and cognitive bias....66
2.5 Summary 69

3 Theoretical foundation....71

3.1 Preamble....71
3.2 Research theory 72
3.2.1 Background assumptions: 72
3.2.2 Theoretical premise 74
3.2.3 Variables....79
3.3 Methodology selection & validation 83
3.3.1 Method....84
3.3.2 Methodology....90
3.3.3 Theoretical perspectives 92
3.3.4 Epistemology 94
3.4 Summary 97

4 Research methodology 98

4.1 Research questions 98
4.1.1 Primary question (RQ1)....98
4.1.2 Supporting questions 99
4.2 Research Design 101
4.2.1 Formulate factors that influence software development outcomes 103
4.2.2 Instrumentation....107
4.2.3 Data collection....126
4.2.4 Data analysis....128
4.2.5 Interpret data....135
4.2.6 Inform 136
4.3 Summary 137

5 Data analysis....138

5.1 Data boundary 138
5.2 Measurement guidelines....139
5.2.1 Hypothesis testing....140
5.2.2 Relationship testing 141
5.3 Sample....142
5.3.1 H3 Stratum: Generation....144
5.3.2 H4 Stratum: Tenure 145

5.4 Research findings 146
5.4.1 Introduction 146
5.4.2 Description....147
5.5 Summary 167

6 Discussion....170

6.1 Evolution 170
6.1.1 Software crisis 170
6.1.2 History and the need to succeed 171
6.1.3 Problem significance 172
6.2 Cognitive theory....173
6.2.1 Problem significance 174
6.3 Cognitive bias....175
6.3.1 Problem significance 177
6.4 Chaos contributors....179
6.4.1 Problem significance 181
6.5 Argument....182
6.6 Research findings 183
6.6.1 Experts in different roles 183
6.6.2 Experts in different generations....184
6.6.3 Experts with different tenure 184
6.6.4 Experts in different groups 185
6.7 Research deficiency....186
6.7.1 Social class 186
6.7.2 Data collection focus 187
6.8 Conclusions 192

7 Future work 197

7.1 Replicating research 197
7.2 Research generalisation....200
7.2.1 Generalization....200
7.2.2 Transferability 201

8 Summary 202

9 References 204

Annexure A: Sample size calculation reasoning i

Annexure B: Literature review summary on reasons for software project failure i

Annexure C: Participants information sheet i

Annexure D: Question design i

Annexure E: Questionnaire i

Annexure F: Ethical Considerations i

Annexure G: Ethics Clearance Certificate i

Annexure H: Choosing an appropriate statistical procedure i

Annexure I: Research Data Analysis i

Table of Figures

Figure 1 Reading streams xiv
Figure 2: The maturing software crisis 25
Figure 3: The Standish CHAOS report 2010 41
Figure 4: Theoretical view 74
Figure 5 Quantitative approach 84
Figure 6 Research design decision tree 102
Figure 7 Research process 103
Figure 8 Factor aggregation 106
Figure 9 Question composition 113
Figure 10 General sample makeup 126

Figure 11: Role mean comparison 149
Figure 12: Role P-Value view 151
Figure 13: Role R-Value view 153
Figure 14: Generation mean comparison 155
Figure 15: Generation P-Value view 157
Figure 16: Generation R-Value view 160
Figure 17: Tenure mean comparison 162
Figure 18: Tenure P-Value view 164
Figure 19: Tenure R-Value view 167
Figure 20: Role, Generation & Tenure hypothesis test 168
Figure 21 Variable margin of error (e) versus estimated population (n) ix
Figure 22 Sample Size calculation distribution x
Figure 23: Role and Generation versus Tenure sample distribution v
Figure 24: Role and Tenure versus Generation sample distribution viii
Figure 25: Role mean comparison xiv
Figure 26: Role P-Value view xix
Figure 27: Role Probability test view xxi
Figure 28: Role R-Value view xxiv
Figure 29: Generation mean comparison xxvii
Figure 30: Generation P-Value view xxxi
Figure 31: Generation Probability test view xxxiv
Figure 32: Generation R-Value view xxxviii
Figure 33: Tenure mean comparison xli
Figure 34: Tenure P-Value view xlvi
Figure 35: Tenure Probability test view xlix
Figure 36: Tenure R-Value view lii

Readers guide

Document outline

This dissertation documents the comprehensive journey the researcher undertook whilst investigating and understanding the phenomenon in terms of the effect that expert judgement has on the evolution of software development practice.

Chapter 1 presents an introduction to the study and in addition to the scope, delimitations, significance and problem statement, provides a high-level background view pertaining to the state of software engineering. The chapter proposes the primary objective of the research in terms of investigating whether perceptions of cause and effect in software outcomes are similar between experts in dissimilar roles, generations and tenure.

Chapter 2 provides an overview of the existing literature and discusses several topics, all related to providing insight in support of augmenting the understanding of the research area. The literature review reflects on the history of the software engineering industry with a specific focus on how the industry has evolved over the years. The focus is then adjusted to incorporate the software crisis as a filter and the relevance of such on modern software engineering initiatives. The literature review continues by providing a view of software outcomes and unpacks reasons why some software development projects fail while others succeed. Insight into existing failure classifications and an in-depth explanation of the factors for software development failure are then discussed. Chapter 2 concludes with existing research related to the human factor as a primary contributor towards software development initiatives and describes the importance of cognitive bias as a contributor towards software development failure.

Chapter 3 provides a bridge between the literature review and the research methodology by incorporating a review of existing research practices and how they relate to the research phenomenon. This chapter details the reasoning in terms of a theoretical foundation by illustrating the development of a research theory that underpins the phenomenon and research questions described in chapter 4. Chapter 3 concludes by providing the basis for the methodology selected, by identifying its suitability in terms of a method, methodology, theoretical perspective and epistemology.

Chapter 4 focuses on the research methodology by stating the research question and providing a description of the overall design. The research question is proposed in conjunction with several granular supporting questions. The research design section explains the combined research effort in terms of a bespoke design, based on the learnings of the methodology selection and validation section in chapter 3. This design incorporates a six-step process capable of guiding the research from data collection to completion.

Chapter 5 deals with finding relevant research facts within the data sets collected during the research process. An explanation is provided on the research samples as they relate to the supporting research questions before statistically evaluating the dependence and covariance of each data set. Statistical analysis techniques such as P-Value, T-Test, Student's probability and Pearson Correlation Coefficient are used. This chapter, however, documents the statistical findings as they relate to the research questions at a high level only, and a complete analysis of the data is provided in Annexure I (*Research Data Analysis*).

Chapter 6 documents a discussion by incorporating the information gathered during the literature review and the data analysis phase of the research. The chapter aims at providing insight in terms of the evolution of software engineering, the role of cognitive

theory, and the state of software engineering in conjunction with the statistical relevance of the research finding, to provide a conclusion on the effect of expert judgement on the cause and effect of software development project outcomes.

Chapter 7 is dedicated to research fundamentals such as research replication, generalisation and transferability and chapter 8 provides a high-level summary of the dissertation. Chapter 9 provides a conclusive list of all external research sources and literature used during the course of the study. Annexure's A through I provide the supporting documentation referenced and used to prepare this dissertation.

Reading instructions

This research deals with the effects that expert judgement has on the evolution of software development practice. The dissertation explains that experts from dissimilar backgrounds reason differently and illustrates that the construction of personal perspectives is based on the constructor's interests. A fundamental principle of research is that a research result should be clear and unambiguous and although readers are encouraged to interpret the result based on their own experiences and knowledge, an attempt is made to guide readers through the research journey by providing relevance in terms of the readers own interests. As such, the document is structured in such a way that readers with dissimilar interests can experience the research journey in the manner that focuses on their specific interests.

The document provides several different reading streams or paths. The first caters for a reader that wishes to experience the entire research journey. This type of reader is not guided by any instruction and will read the document from start to finish. Other types of readers are assumed to have differing interests and each reader would have a specific takeaway. These are described as follows:

	**Academic reader**: A reader interested in the research from an academic perspective. It is assumed that this reader values the design and implementation of the research methodology over the details and as such will spend more time critically reviewing the research methodology and the process around it.
	**Pragmatic reader**: A reader interested in the research from a problem or solution perspective. It is assumed that this reader values the sensibility and practicality of the problem and outcome or solution over the theoretical considerations that have guided the journey from problem to solution.
	**Executive reader**: A reader interested in the research from a high-level perspective. It is assumed that this reader values both the theoretical considerations and practical outcomes but at a non-detail level.

The document is structured in such a way that each type of reader is guided through the text based on their particular interest. In addition, to further simplify a reader's journey, the following reader's note tags can be found throughout the dissertation.

This text provides reader notes on the content it is attached to.			

The contents of the reader's note will assist the reader in determining the level of their personal interest in the text that follows, by evaluating the density of the icons placed on the right, for instance, lighter icons such as  and  suggest that there may be limited interest for academic and pragmatic readers. Equally, darker icons such as  and 

suggest that the text to follow may interest academic and executive readers. In addition, each reader note contains a brief description. This description is intended to provide a high-level note on the text in order to assist the reader in validating the interest suggestion.

Finally, the following graphical representation of each reading stream is provided in order to assist readers with navigating the document.

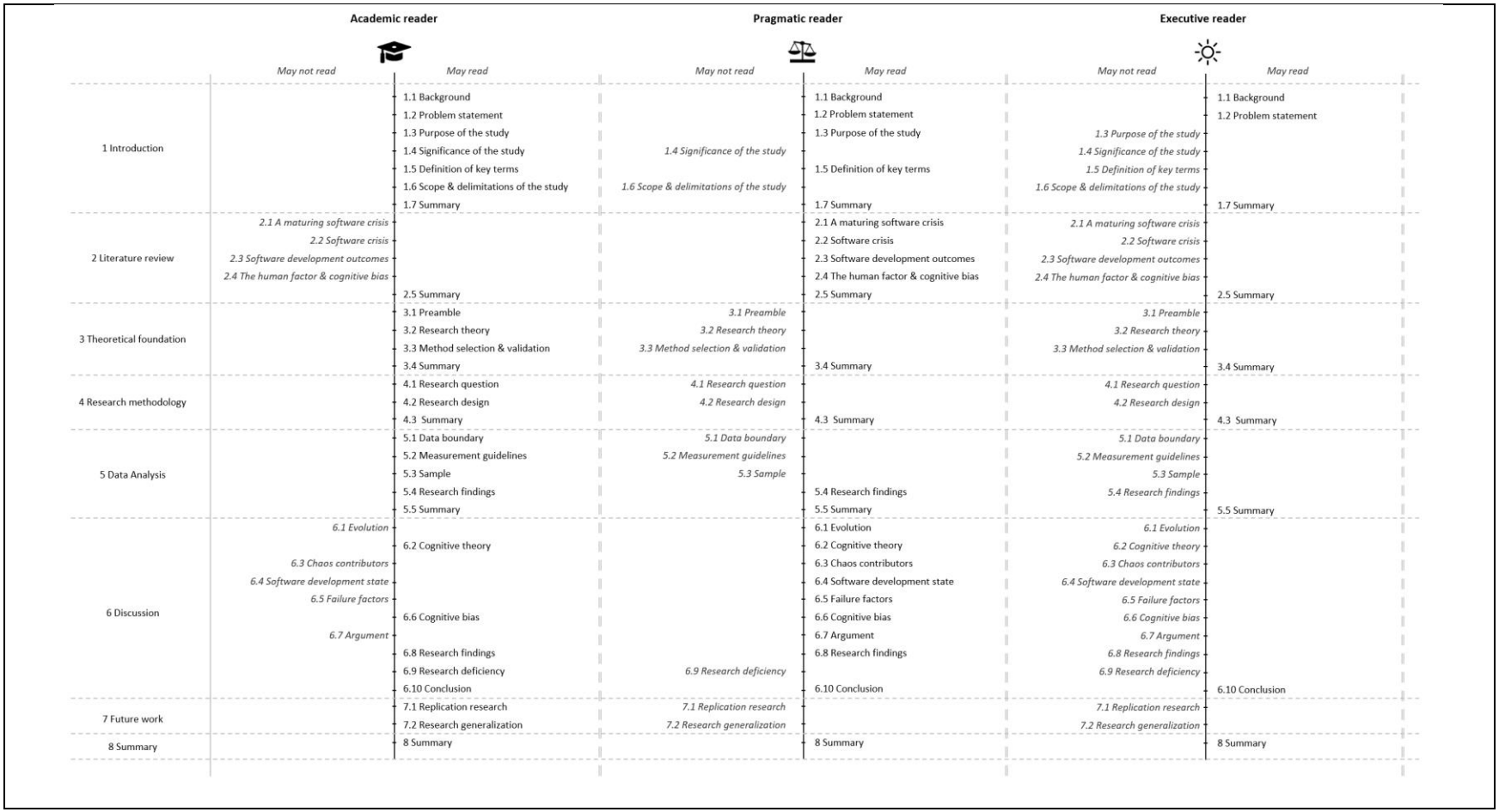


Figure 1 Reading streams

1 Introduction

The author made every attempt to reference existing papers and opinions included in this study. However, in certain parts of the paper, commentary may be found that appears to have been included without citing relevant sources. The author does not claim ownership of any prior work done in those cases but provides a perspective based on nearly two decades of relevant professional software engineering experience. The perspective was formed by a combination of professional experience, academic study and deliberations with well-respected industry experts. Where applicable, these perspectives are marked with a footnote citing "Author's perspective".

This section provides a high-level background to problems in software engineering and is considered relevant to all types of readers





1.1 Background

Unlike other formal engineering disciplines, there are no requirements to licence software engineers as professionals in South Africa. This phenomenon is not constrained to the local software industry but is apparent across the world. Atlee (2002) suggested that the requirement to licence software engineers would "act as a driver to improve our engineering of software and our education of software engineers" (Atlee, 2002). The conflict between the computer science and professional engineering community curbs progress with arguments such as "licensing software engineers would not improve software quality, and would provide false assurances to the public about software quality". Atlee (2002) suggested that "software engineering should be practised by whomever is best, and there is no evidence that engineers are the best" (Ibid.).

It is suggested that the licencing of software engineers would have a significantly positive influence on the South African software industry, as formality and rigour guided by compliance, have the potential to promote quality through predictability and repeatability (Ibid.). However, it appears that the frequency of change in software methods and

practices reduce the ability to settle on a single software standard and as such, limits the formality required in establishing a software engineering controlling body.

A common belief suggests that the turmoil in the arena of software development methodology is rooted in the successes and failures of software development initiatives. It can be argued that the software engineering industry has been inundated with challenges surrounding the development of software, as software methods and practices appeared to be invented and then reinvented to guide successful outcomes. This supports the argument that the evolution of software engineering over many years has led to a plethora of different methods and practices, some spawned by others, but none entirely successful in solving the software crisis. It is suggested that the reasons for this phenomenon are not limited to but include the complexity of understanding factors for determining the success or failure of software projects.

It stands to reason that the software industry has remained in crisis and software development methodologies have been incapable of achieving better results, as its evolution appears to be founded in prejudice. The focus of the research presented in this dissertation was to undertake a study to understand whether the *perceptions of experts* in terms of factors that influence software development outcomes *were suitable drivers* for software practice.

This section articulates the problem the research project aims to investigate and is considered relevant to all types of readers





1.2 Problem statement

It is suggested that software development should be straightforward and predictable.

> *In a perfect world, a software development team would understand the requirements, have the necessary skills to complete its tasks and recognise the pitfalls they could encounter along that way. A team would know how to approach a development project to ensure they meet its objectives on-time, within budget and with the required scope to deem the solution a success. Teams would follow blueprints to guide them through the process and collect and use empirical evidence to forge 'expertism'[1]. Experts are born out of trial and error and drive the evolution of software development practice.*

Unfortunately, it appears that the absence of formal approaches aimed at collecting empirical evidence on the failure or success of software development projects has resulted in recording different expert opinions and perspectives. This supports the idea that as experts innovate in the realm of software practice, methodologies are born underpinning bias, rather than fact.

[1] Refers to the qualities that make up an expert

1.2.1 Problem Context

Creating software is a complex and expensive exercise that requires both engineering and scientific skill as well as domain insight and careful planning to solve its intended problems. Software, especially when crucial to an operation, may hold countless benefits to an organisation. It can be reasoned that software systems have the potential of reducing operational costs or even increase business profitability, but evidence (Guest, 2014) suggests that failed software development project can have severe consequences to the organisation funding it.

Failed software development initiatives are constantly reported in the news, and they impact not just the organisation but have the potential to affect the poorest of the poor. On 8 May 2014, almost a decade after the Integrated Financial Management System (IFMS) project had started, the South African government announced that it was going to abandon the project with an estimated loss of one billion Rand (Guest, 2014). It was reported that the failure of the 2014 IFMS project was due to *subjective criticism* of the intended architecture, strategy, objectives and project deadlines as well as partner selection and management control of the initiative (Ibid.). No official project data was provided and the reasoning supplied by experts was deemed a sufficient explanation to the taxpayers of the country.

It could be argued that software professionals are experts and as such, their interpretations are considered authoritative. It is also suggested that experts are diverse, have different roles, ages and experience levels and come from diverse backgrounds, all contributing towards their perspectives (Thomas and Fernández, 2008). Experts gain their status by learning from failure using empirical evidence. However, it is argued that with a lack of mature measurement frameworks, experts are forced to report reasons for software development failure using matter-of-judgement, rather than matter-of-fact.

Thomas and Fernández (2008) in their search for project success factors wrote that software development success was a challenging and elusive concept and meant different things to different people. They wrote that "*the ascription of success and failure is a social accomplishment dependent on the perspective of the subject*" (Ibid.). People play a significant and essential part in developing software systems, and as such, reasons for the *failure or success* of software development projects *are subjective* (Heeks and Bhatnagar, 1999), *resulting in* an assortment of *different perspectives*, all *based on interest* (Karch, 2011).

The formal gathering of data may reduce the reliance on expert judgment, but the collection of empirical evidence on the failure of software development projects is often considered ineffective, as the quality of data could be questioned (May, 1998). May (1998) suggested that "studies into software project development failures were neglected in an effort to reduce further losses, and in certain cases, where investigations were done, they resulted in hidden or manipulated data in order to protect careers and reputations of the parties involved." (Ibid.). Heeks and Bhatnagar (1999) suggested that the source of failure was rarely found, as failures were swept under the carpet, while successes were shouted from the rooftops (Heeks and Bhatnagar, 1999).

It could be reasoned that experts take responsibility for their successes. Appropriately recognising threats when they occur, could be the difference between the success or failure of the development initiative. However, it is suggested that the appropriateness of expert judgement in recognising threats cannot add value to learning from project outcomes. It is argued that the software crisis can only be solved if expert judgement is used appropriately. As such, understanding that experts in different roles, ages and experience levels have different perspectives, the importance of creating and adopting mature practices for measuring the success and failure of software development initiative

should become a priority for the software industry. It is argued that software methods based on this foundation would have greater success than those founded in opinion.

This section defines the specific purpose of the research study and is considered relevant to all types of readers.			

1.3 Purpose of the study

Based on the problem described above, understanding that software experts reason differently is relevant. As such, the primary objective of the research presented in this dissertation was to undertake a study to understand if the perceptions of factors that influence software development outcomes were similar between experts practising in different roles, generations and levels of experience.

This section specifies why the research is significant. It may be interesting to all readers, but its incorporation is to describe how the research contributes towards the body of knowledge. As such, it is considered more relevant to academic readers.			

1.4 Significance of the study

"Research is a systematic process of collecting, analyzing [*sic*], and interpreting information (data) in order to increase our understanding of a phenomenon about which we are interested or concerned" (Leedy and Ormrod, 2009). Formal research is about knowledge advancement as well as the significant benefits a population may gain out of an increased understanding of its challenges (Ibid.). The phenomenon studied by this research provided relevant insight into the inadequacies of utilising expert judgement to understand software development outcomes. In explaining the phenomenon, the value of fact-based analysis founded in data can be re-established. This increased understanding could offer knowledge advancement to two distinct realms, academia and practice.

In software engineering, technology and practice evolve rapidly, and it is argued that this evolution has been responsible for creating a gap between practitioners and academia (Lee *et al.*, 2002). Lee et al. (2002) suggested that "It has long been recognised that there are significant perception gaps between information systems (IS) academics and IS practitioners with regards to the required IS knowledge/skills to perform their professional jobs successfully" (Ibid.). The value of this research is that it provides academia with an affirmation that formality remains key to instructing students and grounds prospective practitioners in formal data collection.

Additionally, in terms of expert judgement and its role in software evolution, no research could be found that investigated the phenomenon that fuelled software methodology evolution and the fundamental principles that underpinned its motivations. This study provides concrete evidence on the topic and advances the existing body of knowledge by offering a relevant view of why practices evolve and why software development teams should be selective when adopting a software development methodology. It can be argued that trusted and proven methodologies are essential to the success of software development initiatives, and as such, the value to the software industry is the insight that methodologies are created out of necessity, may be based on subjectivity and are not always founded in fact. Using this insight when evaluating software development methodologies for appropriateness of use, would assist development teams in selecting a suitable method aligned to their needs.

1.4.1 Reset focus on formality

The reasons underpinning the success or failure of software development projects are complex. Even though opinions surrounding the licencing of software engineers differ, and the impact of these qualifications on the quality of the software solution is questioned, the purpose of this study was not to contribute towards augmenting the

understanding of this concept. By means of providing a combined perspective of factors that internally affect the performance of a software development team, this research could provide additional insight that could form part of the foundation required in establishing a software engineering controlling body.

This study is significant in that it provides relevant information on how experts reason. Karch (2011) suggested that the grounds for software failure were varied and numerous and included specific factors such as process, rather than the technical challenges a software team experienced (Karch, 2011). Winston Churchill quoted that "The price of greatness is responsibility" and it could be argued that software development teams do not plan to fail but do so due to factors within and outside the realm of their control. It is believed that with an understanding of how expert reasoning blurs the reality of software failure, the focus on formality would be re-established. It is suggested that formal governance processes combined with a high maturity in acceptance and compliance with such governance processes would provide a robust framework for executing a software development project in a predictable high-quality manner.

1.4.2 Lack of engineering maturity

The South African IT industry has historically not been governed by any laws that require a software engineer to be registered as a professional engineer. The underlying reasons are numerous and include the inability to establish a governing body as there is no generally accepted body of knowledge or code of ethical conduct on which to base such a body (Dwolatzky, 2010). Dwolatzky (2010) suggested that "We need, like other professions, to define Bodies of Specialised Knowledge for ICT-related jobs. We need to have a Code of Ethical Conduct, enforced by Government legislation to protect the public from non-professional behaviour. Professionals should be registered by a statutory body and made accountable for the work that they do." (Ibid.). Atlee (2002) suggested that the

"Licensing of software engineers will become serious only when the public demands it - possibly after the catestrophic [sic] failure of some software system" (Atlee, 2002).

A wealth of differing standards are found that could be considered "the" body of knowledge on which to base a software engineering governing body, such as the Software Engineering Body of Knowledge (SWEBOK). Unfortunately, it would appear that no standard is capable of providing advice to software development teams on how to deal with influences that affect the delivery of software development projects. In addition, no standard can offer mechanisms for measuring the impact influences have on the success or failure of a software development initiative, and none define a commonly accepted way of working that software development teams could adopt, that purely act as a guiding principle to software engineering.

It is argued that maturity in software engineering can only be reached through predictability, and the only road to predictability is via a standard approach. This research provides a foundation for classifying factors that negatively influence software delivery on which future research in measuring the impact of these factors can be based.

This section demystifies concepts and terms that are introduced as part of the research study. Readers that are focused on the content will see these terms often, so providing an upfront meaning is important. As such, this section is relevant to academic and pragmatic readers.





1.5 Definition of key terms

1.5.1 Influence

The American Heritage Dictionary of the English Language defines "influence" as "a power affecting a person, thing, or course of events, especially one that operates without any direct or apparent effort" (Houghton Mifflin Company, 2000). Fred Reichheld (2006) suggested that influences were largely associated with advocacy, but could often be negative in nature (Reichheld, 2006). For this research, an "influence" was defined as:

> *a promoting or detracting power that affects the behaviour of a person, process or practice, and the impact it has on the outcomes of a software development initiative.*

"Influences" can exist in one of two categories, "internal variability" and "political impacts".

1.5.2 Internal variability

"Internal variability" is described as the factors that affect a software development initiative, are internal to the software project and include the individuals, technologies and techniques present in a software development team. For this research, "internal variability" was defined as:

> *an "influence" internal to a software development team, with the potential of being controlled by the software development team.*

1.5.3 Political impacts

"Political impacts" are described as the factors that affected a software development initiative and are considered external and overarching to the software development initiative. For this research, "political impacts" were defined as:

> *an "influence" external to a software development team, with no potential of being controlled by the software development team.*

This section specifies the research boundary and documents what cannot be achieved by the research. Its incorporation is to highlight gaps that may create future research interests. As such, it is considered more relevant to academic readers.			

1.6 Scope and delimitations of the study

The phenomenon documented in this dissertation provides a rich foundation to base any research on. However, no single investment in research could have the capacity to cover the full scope of the phenomenon and its associated problems. As such, the study was restricted in several ways to reduce the research scope, while promoting a focus on the importance of collecting relevant information. These included the following:

1.6.1 Research aspect

While failure is considered a general problem in software engineering, the effects of failure in bespoke software development initiatives provide a more specific vantage point on which to conduct research. The information collected for this study may not be relevant to non-bespoke software projects and as such excluded the development of commercial off the shelf products. In addition, embedded software development initiatives may be considered part of the bespoke software group but were specifically excluded due to their close association with hardware and its limitations regarding the

roles required to create software in this space, such as business analysis. As such, these realms were not considered critical to this study.

1.6.2 Research components

The social construction of technology (SCOT) theory suggested that all parties involved should participate in evaluating technology outcomes and specifically included the constructors and consumers of the technology under development (Bijker, Hughes and Pinch, 1987). Although this approach could be assumed an appropriate theory to base this research on, accessing the consumers or users of bespoke software development projects was not possible. In addition, the inclusion of this dynamic had the potential of increasing the problem scope exponentially, as software user perspectives would have introduced a different language into the mix, requiring translation between technical and non-technical interpretations. As such, the research relied exclusively on contributors within a software development team, such as developers, business analysts, architects and project managers. These roles were believed to be responsible for demystifying the factors that influenced the outcomes of bespoke software development initiatives and were considered primary drivers of software practice evolution.

When dependent variables were considered, an appropriate level of granularity was key. Moe (2017) noted that fine-grained granularity could have the potential of bloating the research scope beyond value (Moe, 2017). The undertone of this research was founded in influences of bespoke software development initiatives but defining a concrete list of all factors may have proven impossible. As such, the study aimed at collecting “influences” from an existing body of knowledge by reviewing current literature in the field. Existing factor categorisations were investigated and used in the formation of the research theory to reduce the complexity in dealing with the data at a granular level. Thus, the instrumentation was based on “influences” at a categorisation level, rather than the details

used to create them. This approach provided a less-granular basis to increase the understanding of the phenomenon.

1.6.3 Methodology

Methodology was viewed as a significant part of the research process, and its design was deemed paramount when information gathering was considered (Moe, 2017). Moe (2017) suggested that methodologies were followed to solve common problems, but each method was intended to interrogate the data space differently. As such, a pragmatic approach was favoured that included; descriptive research approaches to understanding existing literature, particularly when "influences" were considered; correlational approaches to collect data from the population and; causal-comparative approaches to increase the understanding of the data collected. Thus, each approach incorporated into the design of the research method, limited the research scope to promote focus in collecting relevant data from its intended space.

1.6.4 Geographic delimitations

The creation of the internet introduced an extensive source of knowledge to the world. Those fortunate enough to tap into this source experienced an unprecedented augmentation to their bodies of knowledge. This knowledge base provides the ability to compare personal experiences across a greater population and is often used in research to generalise opinions. Regarding available literature, the study interrogated existing knowledge bases to increase the foundational understanding of the phenomenon. However, the data collection was bounded to the South African software engineering population in order to maintain an appropriate research scope.

This section concludes the introduction by restating some of the important aspects highlighted in the text, thus suggesting that it is appropriate for all reader types.





1.7 Summary

It could be reasoned that software development should be straightforward and predictable. However, evidence such as the software crisis and the perpetual evolution of software practice, suggests the contrary. Software development approaches and practices have evolved over the years, and it could be argued that each addition intended solving problems introduced by its antecedents. Popular belief suggests that methods evolve to lessen the impact that the software crisis has on the industry (Baggen *et al.*, 2011; Ensmenger, 2017), but opposing arguments reject the notion of a persisting software crisis altogether (Haigh, 2010; Fitzgerald, 2012).

This research is founded on the principle that the software engineering industry has remained in crisis and the evolution of software method has had limited impact on solving this crisis, as methods may be founded in perception rather than fact. It suggests that software experts fuel innovation in software practice and provide relevance to understanding "influences" on software outcomes. However, experts have different beliefs and backgrounds, resulting in various perspectives. It could be argued that a statutory body, guided by standards and a Code of Ethical Conduct, may introduce the necessary engineering maturity required to construct software in a predictable manner, but, where access to trusted information is not available, experts have no choice but to rely on subjective reasoning that introduces bias into their perspectives.

The main purpose of this research was to undertake a study to understand whether perceptions of factors that influence software development outcomes were similar between experts practising in different roles, ages and experience levels. As no current

research is available that highlights the existence of bias in software development methodology evolution, advancement in the existing body of knowledge can be secured.

2 Literature review

This section provides a description of the history of software engineering with a specific focus on how software development methodologies evolved. All reader types may find interests in reading this section and as part of the literature review aligns to appropriate research practices. It is however believed that pragmatic readers may find the section more valuable.			

2.1 A maturing software crisis

T.S. Eliot wrote (1921), “the historical sense involves a perception, not only of the pastness [*sic*] of the past, but of its presence” (Eliot, 1921). Our future is defined by our current actions and learnings from the past.

During the short, but interesting history of software, many authors have written about process and practice cause and effect in software engineering. Barry Boehm (2006) described the history of software engineering in terms of Hegel’s dialectic of increased human understanding. Boehm (2006) suggested that a thesis exists in the form of why things happen the way they do. Subsequently, an antithesis is then established as a better way of doing things due to failures of the thesis. Finally, a hybrid of the thesis and antithesis covering the best of both, but avoiding their defects is arrived at in the form of a synthesis, primarily due to the antithesis rejecting too much of the thesis (Ibid.). In his article, Boehm (2006) illustrated Moore’s law of computer element performance doubling every 2 years, as well as iterative causation in each antithesis during the last sixty years.

In the 1950’s, the practice of software development was analogous to hardware engineering, and strict engineering principles and processes, based on the waterfall model were followed to bring software to market (Ensmenger, 2010). During these years, software was constructed by engineers and mathematicians. Minor slippages concerning

time and budget were experienced, but the result was of high-quality (Boehm, 2006). Herb Benington (1983) reflected on this period and his involvement in the building of a large-scale software solution for SAGE, and stated that "it is easy for me to single out the one factor that led to our relative success: we were all engineers and had been trained to organise our efforts along engineering lines." (Benington, 1983)

Towards the end of the 1950's the industry realised that computer programming was different from other engineering activities and that programmers could not prosper under a rigid climate of hardware engineering management (Boehm, 2006). Software was considerably easier to modify than hardware, and the industry moved towards the "code and fix" approach.

Software became an independent product, and this paradigm shift resulted in software development becoming more people-intensive as the demand for engineers and mathematicians interested in programming outstripped its supply (Ensmenger, 2010). The industry began training and employing non-engineers to fill the gap to overcome this shortage, as non-engineers were more comfortable with the "code and fix" approach (Ibid.). This disruptive innovation in software process had a perpetual causation on software development which spiralled down into the software crisis of 1968 (Mcilroy, 1969).

The synthesis of the formal 1950's paradigm and crafted 1960's paradigm gave birth to the Waterfall Method in the early 1970's (Boehm, 2006). This methodology brought formality in terms of requirements engineering and design (before code) to the unstructured "code and fix" approach and additionally brought software-oriented techniques to the more formal hardware engineering process (Ibid.). The waterfall process quickly became the new thesis but was document-intensive, slow-paced and expensive to use (Ibid.). The exponential cost of change was considered the major

contributor to software costs. Boehm (1972), in his article, "Software and its impact", noted that in certain cases, software costs were around three to seven times that of hardware costs (Boehm, 1972).

The 1980's saw the emergence of approaches aimed at achieving an improvement in productivity while reducing the excessive cost of software development. These included high-level programming languages, powerful workstations and ground-breaking advances in software reuse (Boehm, 2006). The use of visual programming and object orientation opened the door for domain architecture.

In an attempt to highlight the complexities in software engineering, Fred Brooks (1986) in his paper called "No Silver Bullet" argued that user requirements (essential complexities) cannot be optimized to improve time-to-market, whereas problems injected as part of the engineering effort (accidental complexity) can be minimized (and in certain occasions removed) by automating repetitive tasks (Brooks, 1986). Brooks' (1986) suggestions of (a) Rapid Prototyping as part of a planned iteration in establishing software requirements, and (b) growing software organically, adding more and more functions to systems as they are run, set the stage for the second antithesis in software engineering by giving birth to evolutionary development methods (Boehm, 2006). During this period the United States Department of Defence sponsored the Software Engineering Institute (SEI) at Carnegie Mellon. The institute was tasked with establishing a process capable of increasing reliability in the software engineering industry (Berander *et al.*, 2005). The federally-funded research and development group developed and published the software capability maturity model (SW-CMM) in 1984, as a process for assessing an organisation's software process maturity (Boehm, 2006). Boehm (2006) recalled that "based extensively on IBM's highly disciplined software practices and Deming-Juran-

Crosby quality practices and maturity levels, the resulting SW-CMM provided a highly effective framework for both capability assessment and improvement" (Ibid.).

With the birth of the World Wide Web in the early 1990's, a strong emphasis on time-to-market emerged. The software industry moved away from the conventional sequential waterfall processes and focused more on new concurrent engineering techniques such as Barry Boehm's spiral model and Tom Gilb's Evolutionary Life Cycle (EVO) (May and Zimmer, 1996; Boehm, 2006). Boehm (2006) noted that "in the late 1980's Hewlett Packard found that several of its market sectors had product lifetimes of about 2.75 years, while its waterfall process was taking 4 years for software development." (Boehm, 2006). May and Zimmer (1996) of Hewlett-Packard noted that the evolutionary method benefited both internal operations and marketing as it reduced the risks of a software project (May and Zimmer, 1996). The authors wrote that "from a business perspective, the biggest benefit of EVO is a significant reduction in risk for software projects" (Ibid.). This period was also the era of the "software hacker", whose contributions towards the disruption of process in the 1960's grew into what was considered a major contribution to parallel engineering (Boehm, 2006). The open-source movement, institutionalised by the establishment of the Free Software Foundation and the General Public License (GNU) gave birth to open-source software such as Linux, Apache, Python, Perl and Mozilla (Ibid.). By 1994, research in the field of software process maturity resulted in a paper published on the personal software process (PSP) by Watts Humphrey. Humphrey (1996) stated that "early work on the PSP indicates that a structured, disciplined, and measured personal software process can provide the guidance and feedback needed to help engineers improve their personal performance." (Humphrey, 1996). The author suggested that software engineers were more aware of their work and process measures provided rapid and unambiguous feedback that reinforced the use of sound engineering principles. (Ibid.)

The effect of Brooks' work in 1986 resulted in an extensive list of Agile methods used towards the end of the 1990's and early 2000's. These included methods such as Stapleton's Dynamic System Development Method (DSDM), Cockburn's Crystal Methods (Crystal), Highsmith's Adaptive Software Development (ASD), Beck's eXtreme Programming (XP) and Scrum by Beedle, Devos, Sharon, Schwaber and Sutherland.

These Agile methods were all based on the notion of incremental development and delivery but provided different processes to achieve it. As investors nursed the scars inflicted by the dot-com crash of 2000, a synthesis was arrived at in 2001 with the launch of the "Agile Manifesto" that reflected the philosophies and core principles of all these methods (Fowler and Highsmith, 2001). The Agile Manifesto proclaimed the importance of people rather than process, was geared towards rapid software delivery and was described as "a set of principles encapsulating the ideas underlying Agile methods of software development" (Sommerville, 2011). Late entrants to the Agile family were Palmer and Felsing's Feature Driven Development (FDD), Poppendiek and Poppendiek's Lean Development and Ambler's AgileUP (AUP) in 2002. This era saw the realisation of Software-Intensive Systems (SIS) described by Boehm (2006) and others. "Concurrently between now and into the 2010's, the ability of organizations and their products, systems, and services to compete, adapt, and survive will depend increasingly on software and on the ability to integrate related software-intensive systems into systems of systems (SOS)" (Boehm, 2006; Lapham and Woody, 2006). Wirsing reported on the results of the strategic research workshop on "Engineering Software-Intensive Systems" held in Edinburgh on May 23 and 24, 2004 and mentioned that "the actual practice shows that the techniques for engineering software-intensive systems suffer from many severe deficiencies in quality and methodological shortcomings" (Wirsing, 2004). In this report, the author suggested that pragmatic methods did not scale up to complex Software-

Intensive Systems and that formal approaches were not well integrated into these practices. The early 2000's set the stage for more than a decade of research and development on software process maturity by the SEI. The SW-CMM became the capability maturity model integration (CMMi), and Watts Humphrey (1999) released his technical manual on the team software process (TSP), based on the work he started in 1996 (Humphrey, 1999).

Humphrey (1999) stated that the TSP "guides engineering teams in developing software-intensive products. Early experience with the TSP shows that its use improves the quality and productivity of engineering teams while helping them to more precisely meet cost and schedule commitments" (Humphrey, 1999). The author further stated that the "Team Software Process (TSP) development follows the quality strategy that was originated by W. Edwards Deming and J.M. Juran. This strategy was extended to the software process by Michael Fagan in 1976. It was further extended with the introduction of the Capability Maturity Model (CMM) in 1987 and the Personal Software Process (PSP) in 1995." (Ibid.)

By 2010, software was well-established as a commodity, and the "browser wars" (Peter, 2004) of the late 1990's provided the fuel needed to secure global connectivity by using the internet. Global collaboration made it possible to execute software engineering efforts around the clock, enabling rapid development but not without a cost (Boehm, 2006). Boehm (2006) noted that "again there are significant challenges in management visibility and control, communication semantics, and building shared values and trust" (Ibid.). Concepts such as Data-Intensive systems (Big Data) and distributed computing became popular towards the end of the 2000's. Jim Tully (2008), vice president and analyst at Gartner, stated that "organizations [*sic*] are switching from company-owned hardware and software assets to per-use service-based models. This will impact the industry in

various ways" (Gartner, 2008). The birth of Cloud Computing in the form of Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) disrupted the way of working in the software industry. Several authors argued that this phenomenon was what Agile required, as cloud computing could accelerate Agile delivery, enable business agility and reduce IT costs, to name a few (Green, 2009; Shriver, 2012).

In 2012 Forrester reported that "Agile thought leaders have not provided much clear or detailed guidance about scaling Agile practices to the enterprise level, where many governance issues are ready to pounce." (Visitacion, Grant and Brown, 2012). Agile's inability to scale became more evident in Scott Ambler's work on the Agile Scaling Methods (ASM) and Disciplined Agile Delivery (DAD) (Ambler, 2010), which was defined by Ambler (2013) as "a governed, hybrid approach that provides a solid foundation from which to scale agile solution delivery within enterprise-class organizations." (Ambler, 2013). Another attempt at solving the Agile scaling issue was Dean Leffingwell's Scaled Agile Framework (SAFe). Scott Ambler (2010) wrote that "it is fairly straightforward to apply agile on a handful of projects, but it can be very difficult to evolve your organizational [*sic*] culture and structure to fully adopt the agile way of working" (Ambler, 2010). Ambler (2010) further stated that SAFe was intended to facilitate real adoption and altering of agile strategies to meet the unique challenges faced by software delivery teams (Ibid.).

In the midst of this chaos, Bertrand Meyer, Richard Soley and Ivar Jacobson (2012) founded the Software Engineering Method And Theory (SEMAT) with a goal of fundamentally changing the way people work in software engineering (Jacobson *et al.*, 2012). Their call for action suggested that software engineering was hindered by a vast number of immature practices, methods and method variants (Ibid.). These practices were

becoming "fads" and lacked sound and widely accepted theory as well as credible experimental evaluations and validations (Ibid.). Jacobson et al. (2013) in a paper entitled, "Agile and SEMAT – Perfect Partners" stated that "both initiatives promote non-prescriptive value-based philosophies that encourage software development teams to select and use whatever practices best fit their context and, more importantly, continuously inspect, adapt, and improve their ways of working" (Jacobson, Spence and Ng, 2013).

The work of SEMAT produced a framework called the "Essence Kernel" designed to describe software development practices (Ibid.). Once they were outlined in the SEMAT notation, methods could be combined, and by leveraging commonalities, methods could start leveraging off one another (Ibid.). Overlapping practices could be reduced to form an overarching composition framework. The Essence Kernel was designed to form a layer lower than methods and practises (Ibid.).

2.1.1 Summary

In his paper "A view of 20th and 21st Century Software Engineering", Barry Boehm (2006) provided an interpretation of Hegel's dialectic of increased human understanding in relation to software development methodologies, in which Boehm (2006) suggested that an initial methodology was developed to control a way of working (thesis). Boehm (2006) noted that a similar but somewhat different methodology (antithesis) was then drawn up based on the failures of the thesis in order to improve its way of working. A third method or synthesis was then arrived at, as a hybrid of the first and second, that encapsulated the best of both approaches, but avoided their pitfalls (Boehm, 2006). The "maturing software crisis" explained above, incorporates Boehm's (2006) perspective and illustrates the evolution of software methods and practice over the last seven decades.

When considering the graphical representation of the text above (Figure 2 -*The maturing software crisis*), it can be argued that each paradigm shift moved the primary focus of software practice between two distinct opposites, idealistic approaches and pragmatic approaches. It is argued that each opposing side differs in terms of its level of formality. Pragmatic approaches such as formal engineering, Waterfall and the Team Software Process thrive on conventionalism, whereas idealistic approaches such as code and fix, Scrum and Agile values the individual over formal practice. It can also be noted that each paradigm shift coincides with specific industry drivers, such as engineering in the early 1950's, software craftsmanship in the early 1960's and formality in the early 1970's to name a few. When combined, the human aspect of software practice evolution is highlighted. It is suggested that software development experts are a primary driver for software practice evolution. As such, their perspectives on factors that influence software development outcomes are relevant when practice evolution is considered.

In terms of this research and the problem described above, it is suggested that due to the nature of the industry, a lack of a statutory body responsible for guiding "Specialised Knowledge for ICT-related jobs" (Dwolatzky, 2010) and the continuous evolution of software methods and practice, constructing software in a repeatable and predictable manner will remain a challenge.

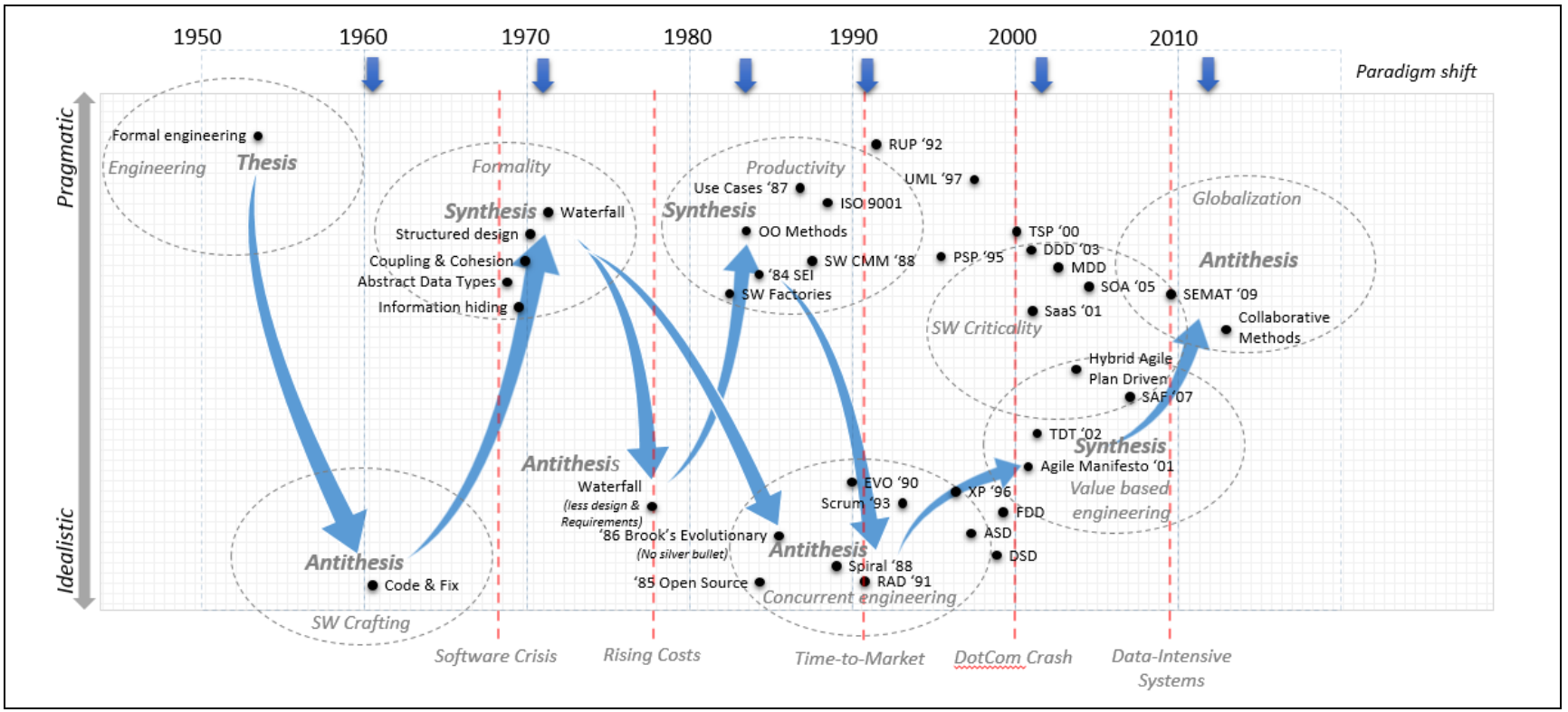


Figure 2: The maturing software crisis[2]

[2] Author's own extended representation of the maturing software crisis, based on "A view of the 20th and 21st Century Software Engineering" (Boehm, 2006)

This section provides a description of software crisis as it relates to the present and the past and attempts to answer the question of if the software industry is still in crisis. This section may prove to be interesting to all reader types but the description may add the most value to pragmatic readers.





2.2 Software Crisis

2.2.1 Then and now

The phrase software crisis has been used by the software industry for many years. This phrase is readily used as a common reference to the state of the software engineering industry. In general, the phrase was used to describe the quality of software produced by the industry. Tedre (2015) noted that the software crisis was fuelled by "overbudget [*sic*], poor-quality, unmanageable, and overdue software projects" (Tedre, 2015). Ensmenger (2010), however, suggested that there were a "broad variety of descriptions of the software crisis, and not all of them agree on many aspects of the 'crisis'" (Ensmenger, 2010). Even though its current interpretation paints a picture of doom and gloom, the term was originally used to describe a crisis of a different nature in software engineering.

Prior to the mid to late 1960's, software was constructed by engineers and mathematicians. Even though strict engineering principles and processes were followed, minor slippages regarding time and budget were experienced, but the result was considered high-quality (Boehm, 2006). Towards the late 1950's software construction moved away from the hard engineering practices and adopted a "code and fix" approach, as software was considerably easier to modify than hardware (Ensmenger, 2010). With an ability to 'productionise'[3] software faster, demand increased. The focus on "code and fix" resulted in a sudden shortage of skilled programmers to meet the new demand for computer software. Ensmenger (2010) presented what was considered the first published

[3] The ability to release software to a production environment

use of the phrase "software crisis" that appeared in an article from Business Week (November 5, 1966) highlighting the skills shortage in the industry (Ensmenger, 2017).

Even though it could be argued that similar skills shortages are experienced in the software industry today, no evidence could be found where this phenomenon was referred to as a software crisis. Various authors have suggested that the crisis experienced in early 1960 never dwindled and was ever-present. In 2007, Douglas Crockford (2007), father of JSON and modern JavaScript stated that "we are now in year forty of the software crisis. It was never solved. We never found a solution to the software crisis, we just stopped recording it, just because it was not newsworthy anymore" (Crockford, 2007).

Three years later, in an article entitled, "Crisis: what Crisis?", presented at the "2nd Inventing Europe/Tensions of Europe Conference" held in Sofia, Thomas Haigh (2010) wrote that "historians have noted with some frequency that basic debates over its identity were never really resolved and that the rhetoric of a crisis in software development has likewise endured for many decades" (Haigh, 2010). Haigh (2010) gave an elaborate explanation for the existence and origin of the software crisis of 1969.

Baggen at al. (2011) suggested that the ongoing software crisis was closely related to software quality and this could be addressed not only from a product perspective but also by improving the development process (Baggen *et al.*, 2011). In their paper, the authors referred to software process improvement practices, such as the Capability Maturity Model Integration (CMMi) and the Software Process Improvement and Capability Determination (SPICE) models and proposed that these reference process models could be viewed as collections of best practices aimed at delivering excellent software.

As a "disclaimer", Baggen at al. (2011) also stated that the above processes could in no way guarantee a certain level of technical quality as the impact of the course on the programming work was only indirect (Ibid.).

In contrast to many, Fitzgerald (2012) referred to the concerns in software engineering as the "Software Crisis 2.0" (Fitzgerald, 2012). "Early efforts in computer-aided software engineering (CASE) sought to automate software development, but they haven't solved the problem. Similarly, initiatives in software architectures, patterns, reuse, and software product lines sought to provide improvements by building on existing foundations" (Ibid.). Fitzgerald's (2012) controversial argument suggested that innovation in software engineering was far behind that of hardware and data, but made clear that progress, regarding incremental advances in software development, had improved lives.

Rai, Madan and Anand (2014) suggested that the software crisis began four decades ago and continues today. However, the authors noted that "today, the situation is quite different. We have a Science of Programming. We know a great deal about how to design and document software, but the 'Software Crisis' continues unabated!" (Rai, Madan and Anand, 2014).

Research has led to numerous conflicting positions around the existence (Buettner *et al.*, 2006) or non-existence (Colburn *et al.*, 2008) of the software crisis of 1969. Colburn et al. (2008) suggested that "any apparent crisis in software engineering today is due merely to the volatile nature of the modern software industry, but this is more an indicator of the health of the industry than of any problem" (Colburn *et al.*, 2008).

2.2.2 Is the software industry in crisis?

Crockford (2007) suggested that the fundamental problem with software was that computer science had not taught the industry how to manage software products. This situation had been exacerbated by a lack of metrics. He mentions that even though some metrics such as lines of code were available, they were not sufficient for measuring software quality or its state of completeness (Crockford, 2007).

Jacobson, Ng, et al. (2012) wrote that developing good software was complex (Jacobson *et al.*, 2012). The authors highlighted three main characteristics of software development. The first was that challenges were multidimensional, and dimensions were not independent of one another. Secondly, writing good software required capable individuals and the authors suggested that it was not just about individuals, but also the team requiring competencies covering all dimensions of the software endeavour. "The creation of good software is a collective intellectual achievement, and to create nontrivial [*sic*] software good collaboration is essential" (Ibid.).

Reto Knobel (2009), Head of Digital at RetoRek provided evidence of thirteen catastrophes caused by software errors between the years of 1962 and 2004 (Knobel, 2009). These included the

- destruction of the Mariner 1 rocket in 1962,
- the self-destruction of the French weather satellite in 1972,
- the inability of the Nimbus 7 weather satellite to detect the hole in the ozone over Antarctica between 1978 to 1984,
- the sinking of the HSM Sheffield in 1982,
- the nuclear war scare of 1983,
- loss of life due to radiation exposure between the years of 1985 to 1987,
- the stock market crash of 1987 (Black Monday),

- the Iranian passenger airliner shot down in 1988,
- the death of 28 U.S. soldiers in Iraq in 1999,
- the destruction of the Mars Climate Orbiter in 1999,
- the Mars Polar Lander crash of 1999,
- the five-day power failure in the United States of 2003, and
- the unemployment benefit loss in Germany of 2004.

An example of large-scale software development failure, relevant to the context of South Africa, is the recent abandonment of the South African Integrated Financial Management Systems (IFMS). The local ICT community welcomed the multibillion-rand government project with open arms due to the enormous opportunities associated with a large-scale project such as the IMFS development (Guest, 2014). On 8 May 2014, almost a decade after the project had started, the South African government announced that it was going to abandon the project with an estimated loss of one billion Rand (Ibid.). Kimberly Guest (2014) stated that "the IFMS initiative turned out to be a constant source of controversy, attracting criticism on the system architecture; partner and/or solution selections; top-level project strategy and management; operational planning and oversight; and the feasibility of the stated objectives and deadlines" (Ibid.).

Boehm (2006) highlighted the importance of dependability of software as critical to software consumers and wrote that "this situation will likely continue until a major software-induced systems catastrophe similar in impact on world consciousness to the 9/11 World Trade Center [*sic*] catastrophe stimulates action toward establishing accountability for software dependability" (Boehm, 2006). Boehm (2006) further predicted that the increasing number of software vulnerabilities in all industry sectors would likely cause a software-induced catastrophe before the year 2025.

2.2.3 Summary

What are the reasons the software crisis remained a topical over the last six decades? It can be argued that software crisis is associated with the failure of software development initiatives over the years. The text above highlights several instances of well-known large-scale software development failures. South Africa's contribution to this list includes the recent failure of the South African Integrated Financial Management System (IFMS) initiative. It appears that the underlying meaning of software crisis may have morphed over the years but when considering recent literature, the phrase is still widely used to describe the state of the industry.

Haigh (2010) explained that in principle, a crisis is created to serve the interest of a particular group and in the case of the software crisis, the implied scope increased while its entrenched position gradually led to the distortion of its actual nature and context as time went by (Haigh, 2010). When considering Haigh's (2010) view, the perspectives of experts (*as a particular group*) on factors that influence software development outcomes are questioned.

This section deals with describing what measurement means to software outcomes. Specific focus is given to describing the differences between software failure and success. This section incorporates existing failure categorisation frameworks, and related factors for software failure to the research categories of "political impacts" and "internal variability". This section may prove to be interesting to all reader types but the description may add the most value to pragmatic readers.			

2.3 Software development outcomes

2.3.1 Measurement

It could be argued that quality is at the heart of software development and over the years, the software industry has incorporated various practices aimed at measuring software success. None have been considered as successful as what is referred to as the "Iron Triangle", "Golden Triangle" or "Triple Constraints" of cost, scope and time as originally described by Dr Martin Barnes in 1969 (Weaver, 2007; Dalcher, 2014). Weaver (2007) wrote that Barnes (1996) "first described the 'iron triangle' of time, cost and output (the correct scope at the correct quality) in a course he developed in 1969 called 'Time and Money in Contract Control'" (Weaver, 2007). Dalcher (2014) noted that what is currently referred to as scope or performance, was initially called quality, as performance is said to reflect the final product aims. The author suggested that "performance means satisfactory function of the product, which has to be fully defined" (Dalcher, 2014).

Ebbesen and Hope (2013) wrote that "it has become the de-facto method to define and measure project success, with the general perception amongst project managers that a successful project is based on these three criteria alone" (Ebbesen and Hope, 2013). Summers (2015) reiterated this but added that "quality may mean meeting technical specifications; however, quality is a subjective concept and will depend upon the perspective of the judge" (Summers, 2015). The author further suggested that incorporating time, quality and cost was effective, but it was flawed in that cost and time became targets instead of constraints, ultimately reducing the focus on the project's purpose (Ibid.). Dalcher (2014) noted that the idea of the triple constraint "presupposes

high estimation accuracy with regard to the initial formulation of the variables of the triple constraint when the degree of uncertainty is at its greatest" (Dalcher, 2014).

Additional criticisms included that of Baratta (2006), who suggested that the "current Triple Constraint is a tool which leads to poor decision making and which ignores significant opportunity costs" (Baratta, 2006). The author stated that no true relationship was presented by the triangle as time was a dimension of cost and as such a relative metric (Ibid.). Serrador and Pinto (2015) suggested that "researchers were increasingly measuring success by the impacts on the organization [*sic*] rather than success at only meeting the triple constraint" (Serrador and Pinto, 2015). The authors found that scope contributed the most in determining software development success and suggested that scope was more than an aspect of efficiency, but that it had the ability to directly impact customer satisfaction when the perspective of quality was considered (Ibid.).

Whilst considering quality in terms of the initial Barnes (1996) description, it could be argued that customer satisfaction is a form of success. Quality is defined by the Oxford Advanced Learner's Dictionary as the standard of something when it is compared to other things (Oxford University Press, 2017a). The term "quality" could be approached from *several different perspectives, all dependent on interest*. Garvin (1984) suggested that each quality view could be categorised as a transcendental view, user view, manufacturing view, product view, or value-based view (Garvin, 1984). Malan and Bredemeyer (2001) explained quality with a specific focus on solution requirements and suggested that quality could be defined as nothing more than the characteristics of the software system. The authors further stated that a stakeholder's degree of satisfaction was directly affected by these properties. Malan and Bredemeyer (2001) defined quality as "the degree of match between the product requirements (stated or otherwise) and the actual product. It is defined from the user's perception, expectation and goals or needs"

(Malan and Bredemeyer, 2001). The Malan & Bredemeyer (2001) paper further proposed that a distinction be made between an Executing System or Run-Time Qualities[4] and Work Products or Development-Time Qualities[5], where quality is driven by user goals. It could be argued that quality was motivated by the development organization's goals, as trade-offs may be required between Development-Time and Run-Time Qualities to resolve conflicting attributes, such as performance and maintainability. The authors suggested that Development-Time traits provided business value, needed for the long-term competitiveness of the business and Run-Time Qualities provided value to the user and influenced the short-term competitive differentiation of the business (Ibid.).

The focus on the customer and their perspective on quality were found to be the primary measure of the success of a software product. Agile defined four values and twelve principles, all aimed at building better software. It is suggested that Agile as a mindset was designed to promote communication and as such, manage the expectations of all involved, specifically the user. The four values guide this by suggesting that processes and tools were considered significant, but not as important as the associated individual and the interactions one could have with them. Equally, working software was valued over documentation, collaboration with a customer over negotiations of contracts and finally, the ability to respond to change, over following a defined plan (Fowler and Highsmith, 2001). Agile frameworks suggested that success could only be measured by demonstrating the capabilities of the working product, and the relevance of the triple constraints were limited in terms of what was required, how much time was available and what the cost of development was, as each of these were largely variable and could be controlled in short cycles (Ibid.). The *principles of Agile were founded in failure*, and

[4] Refers to non-functional requirements, i.e. usability, configurability, supportability, correctness, reliability, availability, performance, fault tolerance and operational scalability.
[5] Refers to the properties of the artefacts of the development process, i.e. localizability, modifiability, extensibility, evolvability, composability and reusability.

signatories of the Agile Manifesto such as Chet Hendrickson (2011) suggested that “if a project is going to fail, I'd rather know that after one month than after 15” (Fowler and Highsmith, 2001). By focusing on development in short iterations, failure could be transformed into learning, allowing a software development team to succeed in the long run.

2.3.2 Success versus failure

The most elusive answer in the short but exciting history of software engineering remains why some software development projects fail while others succeed. Numerous attempts have been made over the last fifty years to demystify failure factors in software development projects, but the collection of empirical data has proven to be the biggest challenge as found by Lorin May (1998). The author stated that “having wasted so much on a fruitless venture, few organizations [*sic*] will invest more time or money to collect and analyze [*sic*] additional data, whereas any data that had been collected may be massaged or hidden to protect careers or reputations” (May, 1998). In a study done by Al-Ahmad et al. (2009) on the causes of software development failures, the authors wrote that “*success and failures are complex concepts, and their perceptions are complicated, unstructured, and not readily quantifiable*” (Al-Ahmad *et al.*, 2009).

Summers (2015) found that failure was still thriving and suggested that in terms of software failure the reasons for failure were ignored and in situations where retrospectives were performed on failed projects, the outcomes masked the true grounds of failure. Also, the author noted that *underperformance of software development initiatives continued as there was no real understanding of what failure was and due to a lack of understanding*, responses to failed projects only dealt with the symptoms and not the causes (Summers, 2015).

Boehm (2006) suggested that the software crisis, as defined in 1968, remained through the ages and has never been solved, regardless of maturing processes and practises

(Boehm, 2006). Boehm (2006) further suggested that "this situation will likely continue until a major software-induced systems catastrophe similar in impact on world consciousness to the 9/11 World Trade Center [*sic*] catastrophe stimulates action toward establishing accountability for software dependability" (Ibid).

Working definition

Heeks and Bhatnagar (1999) proposed the "Conception-Reality Gap" as the largest contributor towards software development success or failure. The authors noted that the assumptions made by system stakeholders significantly contributed towards success and failure and "more specifically, we can say that the conceptual models held by key stakeholders or implicit within reengineered [*sic*] information systems are important. They are important because the gap that exists between these conceptions and public sector realities will determine success or failure" (Heeks and Bhatnagar, 1999).

It is evident that in terms of software development, no generally accepted definition of success and failure exists as both are considered quasi-conceptual and are comprehended differently (Thomas and Fernández, 2008; Al-Ahmad *et al.*, 2009). For this study, success was defined as a

> *software development project where a solution was delivered on time, within budget and was used by its intended audience with the required functionality to deliver business value.*

Equally, failure was defined as software development projects that fell outside the bounds of success or violated any part of the definition of success.

What is defined as success in software engineering?

In his research, Boghossian (2002) defined fifteen factors essential for successful software product delivery. These included a defined product development lifecycle , executive management support throughout the development lifecycle, user involvement at the early stages of development, strong project management, small and miniature project milestones, clear statement of requirements, realistic expectations on the product and development schedule, proper resource and strategic level planning, competent, trained and focused workforce, ownership at all levels, clear vision and objectives, software development and engineering practices, well-defined processes, software estimation and finally independent verification and validation (Boghossian, 2002). In 2014, Dalcher proposed that successes in software development projects needed to be reconsidered and focus had to be shifted beyond failure factors. The author suggested that development teams recognised that user involvement, satisfaction and buy-in were most crucial to development success. Dalcher (2014) further stated that informal approaches increased the likelihood of user acceptance as these were developed to maximise the potential for stakeholder satisfaction (Dalcher, 2014). Thomas and Fernández (2008), however, suggested that "satisfying the customer, or stakeholders, may not constitute overall success if company goals have not been met. Thus it may be possible to satisfy customers or users but not produce benefits for the company" (Thomas and Fernández, 2008).

It could be argued that a successful software project is one that is used by its intended audience. Over the years, similar arguments were made for and against measuring successful projects based on the Iron Triangle. The Standish Group (1994) used this measure as the primary measure of software development success (The Standish Group, 1994). Dalcher (2014), however, suggested that "project success is a rather nebulous concept and the focus on the triple constraint can be too limiting" (Dalcher, 2014). Thomas and Fernández (2008) in their search for project success factors wrote that

software development success was a challenging and elusive concept and meant different things to different people. They wrote that "*the ascription of success and failure is a social accomplishment dependent on the perspective of the subject*" (Thomas and Fernández, 2008).

Why do software development projects succeed?

Thomas and Fernández (2008) suggested grouping success conditions into three sets namely, project management success, business success and technical success, with each set containing several factors unique to that set (Thomas and Fernández, 2008). The authors consolidated budget and time constraints together with the satisfaction of stakeholders, customers, users, steering group and sponsors within the project management set and business continuity, objectives and the delivery of benefit within the business success set. System implementation, requirements met, system quality and system use were confined to the technical success set that included both customer or user satisfaction and stakeholder satisfaction from the project management set (Idib.). The authors, however, concluded that "our study found that the very act of defining and measuring IT project success contributed to success itself" (Ibid.).

In contradiction, Greenwood et al. (2010) suggested that upfront risk mitigation was key to project success and stated that "IT failures diagnosed as errors at the technical or project management levels are often mistakenly pointing to symptoms of failure rather than a project's underlying Socio-complexity which is regularly the actual source of failure" (Greenwood, Khajeh-Hosseini and Sommerville, 2010). Taherdoost and Keshavarzsales (2015), in a paper entitled "How to Lead to Sustainable and Successful IT Project Management", extended this explanation and proposed the "5P" concept as a complementary framework for sustainable, successful development in volatile projects. The authors suggested that successes in software development can be accredited to a balance between efficient management (presiding), planning (process), development

maturity (pragmatic), measurement (performance) and people. They further suggested that success was not guaranteed by facilitating these areas, but rather managing or mitigating the risks associated with them by stating that "as success/failure factors are tied with project success/failure, therefore, the importance of risk management toward success of the projects is undeniable" (Taherdoost and Keshavarzsaleh, 2015b).

What is defined as failure in software engineering?

In their paper entitled "How to Lead to Sustainable and Successful IT Project Management?" Heeks and Bhatnagar (1999) uncovered several project management factors that cause software development failure. These included an inadequate project basis, the wrong person as project manager, inadequately defined interactions, lack of technical project management, lack of commitment to the project as well as the management model utilised (Heeks and Bhatnagar, 1999). Taherdoost and Keshavarzsaleh (2015) extended this view by suggesting that "there is a wide range of literatures [*sic*] on information technology project failures comprising both theory and case study which are not unrelated to factors or variables that underpin successful project management and failure avoidance, embracing technical, managerial, planning, resourcing, and environmental factors (Taherdoost and Keshavarzsaleh, 2015a).

In understanding failure, Heeks and Bhatnagar (1999) proposed that software development failures could be categorized as either total or partial failures. The authors described a complete failure as the conclusion of a software development project where no workable solution was produced and a partial failure as a conclusion of a software development project where only fragments of a viable solution were delivered but were resisted by its intended users. In addition to this categorisation, Heeks and Bhatnagar (1999) suggested two distinct aspects of failure namely, goals unattained and undesired outcomes, where the former refers to user expectation not being met and the latter unexpected system behaviour. (Heeks and Bhatnagar, 1999)

With their "CHAOS" publications, the Standish Group (1994) provided insight into the state of the software engineering industry over several years (The Standish Group, 1994). The group defined failure as software systems that were cancelled before delivery or were delivered but never used. In addition, the Standish Group (1994) also provided a different aspect of failure, aptly named "Challenged", where software systems were completed and delivered, but were over-time, over budget or had fewer features than what was required. Taherdoost and Keshavarzsaleh (2015) supported this approach by suggesting that "in spite of the problem with providing commonly accepted definition of project failures, there is little doubt that failed projects are considered as they do not deliver anything, deliver a product later than expected, or deliver a product that is not useful" (Taherdoost and Keshavarzsaleh, 2015b).

Why do software development projects fail?

Urban myths in software engineering suggest that some projects are doomed before they start. Influential 20th century American thinker and poet, George Santayana (2011) wrote that "those who cannot remember the past are condemned to repeat it" (Santayana, 2011). Lorin May (1998) suggested that the collection of empirical evidence was hampered by a lack of investigations into failed software development projects, and when done, the quality of the data could be considered questionable (May, 1998). May (1998) found that studies into software project development failures were neglected to reduce further losses, and in certain cases, where investigations were done, they resulted in hidden or manipulated data to protect careers and reputations of the parties involved (Ibid.).

Al-Ahmad et al. (2009) however, stated that "success and failures are complex concepts, and their perceptions are complicated, unstructured, and not readily quantifiable" (Al-Ahmad *et al.*, 2009).

In the CHAOS report of 2010, the Standish Group reported an increase of 13.5% in the number of successful projects and revealed that 37% of the sample projects were

delivered on time, on budget and with the required features and functions (The Standish Group, 2012). A total of 42% of the sample projects were challenged, as they were late, over budget, or had less required features than what was defined. The report further stated that 21% of the sample projects were considered failures, as they were cancelled before delivery, or were delivered but never used (*Figure 3 The Standish Chaos report 2010*). Two years later, in an attempt to explain the increase in project success, the Standish Group (2012) reported that "looking at the research differences between successful projects and unsuccessful projects we observed that the successful projects either had a skilled executive sponsor or did not need a strong executive sponsor since they were more mechanical projects" (The Standish Group, 2012).

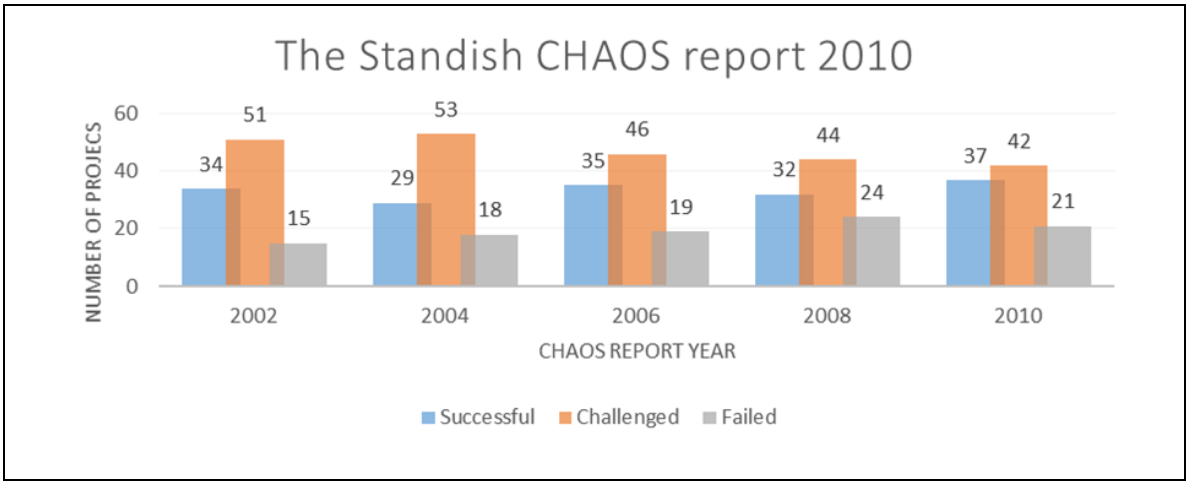


Figure 3: The Standish CHAOS report 2010

Al-Ahmad et al. (2009) proposed that failure factor commonalities existed across domains and suggested six generic types of factors for software development project failures. The authors stated that "it is clear that most of the symptoms of IT project failure belong to the project management root cause" (Al-Ahmad *et al.*, 2009). The generic types defined by Al-Ahmad et al. (2009) included 1. *Project management factors*, where user involvement was not secured, or the scope and objectives were not clear, 2. *Top management factors*, highlighting the lack of top management commitment and no-one to

champion the project, 3. *Technology factors*, focusing on technical experience and team commitment, 4. *Organizational factors*, suggesting that the culture and structure of the team were not appropriate as conflicting interests existed, 5. *Complexity or size factors*, where the project was large and multifaceted, and finally 6. *Process factors*, highlighting the lack of suitability regarding processes and practices.

McKinsey and Company (2012) presented four groups in which reasons for software development project failures were categorized. The formulation of these groups was based on the findings of research conducted by McKinsey and the BT Centre for Major Programme Management at the University of Oxford, entitled "McKinsey - Oxford study on reference-class forecasting for IT projects". The paper suggested that blurred and vague objectives, as well as a lack of business focus with regards to minimal to no stakeholder alignment, were the major causes for *inadequate focus*. *Problems with system content* were due to the technical complexity of the implementation, combined with changing user requirements. Team misalignment and a lack of skills were contributors to *skills problems* and *problems in execution* were due to unrealistic schedules and responsive project planning (Bloch, Blumberg and Laartz, 2012). In their study, the authors found that a sample of 5400 software projects, a total of 45% of large projects had an overrun in budgets that equated to a cost overrun of $66 billion, 7% had time overruns, and 56% has less functionality delivered than what was promised. It was also noted that cost overruns increased by 15% year-on-year for multi-year projects and 17% of overruns on projects were so large that they threatened corporations with bankruptcy (Ibid.). The authors noted that failure regarding these numbers was largely attributed to:

- A lack of alignment between client needs and corporate strategy
- A lack of alignment between business strategy and requirements or sizing of requirements
- Inaccurate cost-benefit analysis
- Changes in requirements
- Inaccurate time allocation and sizing of scope - Proxy Sprint

- No minimum viable product (MVP)
- Business value did not focus on requirements and delivery
- Focus on technology over project goals
- Lack of focus on quality and short delivery cycles

Both categorisation models presented by the Al-Ahmad et al. (2009) and McKinsey – Oxford (2012) studies provided insight into problem classifications, and an alignment between these models are shown in Table 1(*Category alignment between Al-Ahmad et al. and the McKinsey – Oxford study*) below.

The Al-Ahmad et al. (2009) and McKinsey – Oxford (2012) studies revealed that each category had an impact on the next, and the following suggested the interconnectedness of these categories.

- An inadequate focus is a cause of problems with system content.
- An inadequate focus is a cause for skills problems.
- A problem with system content is the cause of problems in execution.
- A skills problem is a cause of problems in execution.

Table 1 Category alignment between Al-Ahmad et al. and the McKinsey - Oxford study

		McKinsey - Oxford study			
Al-Ahmad et al.		Inadequate focus	Problems with system content	Skills problems	Problems in execution
	Top management factors	x			
	Organizational factors	x			
	Project management factors		x		
	Complexity or size factors		x		
	Technology factors			x	
	Process factors				x

It is suggested that software development projects are affected by “political impacts” that fall outside the control of the development team, and development efforts require inputs to deliver their required outputs. Also, the variable action of turning inputs into outputs are affected by factors the development team can control and are thus considered internal.

One may thus deduce that inadequate focus equated to political influences, where problems with system content, skills problems and problem in execution, were "internal variability" factors. In terms of the research, this rationale guided the formation of the research categorisation in terms of "internal variability" and "political impacts" in terms of the categories specified by the Al-Ahmad et al. (2009) and McKinsey – Oxford (2012) contributions. A consolidated view of failure factors for each of these categories is provided in Annexure B (*Literature review summary on reasons for software project failure*) and are discussed in the following section.

2.3.3 Failure Categorization

Political impacts

"Political impacts" describe "influences", overarching to software development projects and as such are considered external to the project. In terms of the Al-Ahmad et al. (2009) and McKinsey – Oxford (2012) research, the stakeholder area, with a specific focus on alignment, commitment, focus and commercial pressure are major contributors to political failure factors. Henderson (2006) stated that "there is too much optimism as to the potential benefits of IT and as to the cost of delivering those benefits" and that these factors were some of the largest contributors towards IT project failure (Henderson, 2006). Krigsman (2012) stated that "too many executives expect technology magically to solve business problems, an almost delusional misconception that leads to unhealthy risk" (Krigsman, 2012a). The most reported political factor for software failure was that objectives and goals were not articulated, vague or blurred and in many cases, very fluid.

In terms of this study, other "political impacts" include failure to manage the user's expectation, failure to gain user commitment, inadequate user involvement, project sponsors lack of experience and poor user input.

Internal variability

"Internal variability" describes "influences" on software development projects commonly regarded as internal to the project and include the team, technology and techniques used in the development initiative.

May (1998) proposed that software development projects failed due to ineffective planning, based on poor cost and schedule estimations, combined with skills that did not match the job. The combination of the former resulted in a general breakdown in communication and ultimately caused a belated awareness of warning signals (May, 1998).

Craig Buckler (2010) suggested that project failures were related to inadequately dissected and poorly scoped projects that tended to change direction, where the estimation was often provided by non-developers or overly optimistic developers. Often, these enthusiastic estimations were taken literally and became the de facto execution time (Buckler, 2010). The author also reported that failures of software development projects could be contributed to what Frederick Brooks (2002) described in his book entitled "The Mythical Man-Month", where additional developers were added to software development projects to speed up the execution (Brooks, 2002). Buckler (2010) concluded that testing time was sacrificed in an attempt to mitigate the risk of schedule slippage (Buckler, 2010).

Kaur and Sengupta (2011) suggested that "the reason of failure can be project team, suppliers, customers and other stakeholders, but the most common reasons for project failure are rooted in the project management process itself and the aligning of IT with organizational [*sic*] cultures" (Kaur and Sengupta, 2011).

2.3.4 Factors for software development failure

The literature review incorporated a study of both formal and informal sources dedicated to demystifying factors responsible for failure in software development projects. This review revealed more than three hundred explanations touching most areas of a software development endeavour. It was however noted that the list of failure factors was not exhaustive, and it could be reasoned that a research study dedicated to consolidating all factors for software development failure might never be able to conclude with a consolidated list.

An exercise in reducing duplication of what was found revealed a list of 209 explanations that were rated and categorised into the classifications provided by Al-Ahmad et al. (2009), McKinsey – Oxford (2012) and the research categories of "internal variability" and "political impacts" (*Annexure B: Literature review summary on reasons for software project failure*). In addition, additional aggregation was performed on the reduced list, and a final consolidated and categorised list of 55 factor-sets were uncovered (*Table 2 Failure reason roll up*). This section provides an overview of each factor set in terms of a refined failure explanation, a consolidated problem area, a description of what was impacted, the categorisation group assigned to the issue and finally, the reason for the categorisation.

Political impacts

Inadequate focus – Top management factors

- Commercials: The project failed due to an imbalance between the costs involved in developing the product versus the value it would bring to the organisation in a financially constrained environment. Kappelman, McKeeman and Zhang (2006) found that even though an estimated costing was provided, the budget approved for the project was less (Kappelman, McKeeman and Zhang, 2006).

- Commitment: Management's involvement is fundamental to the success of a software development initiative. Al-Ahmad et al. (2009) suggested that a lack of executive support jeopardised project success (Al-Ahmad *et al.*, 2009). Schmidt et al. (2001) wrote that management support extended to their commitment and suggested that it included "oversight by executives and visibility of their commitment, committing required resources, changing policies as needed" (Schmidt *et al.*, 2001).
- Communication, control and maturity: Grey (2011) in describing Thejendra (2014) noted that "business decisions made by senior management are commands to be executed, which results in problems and issues being addressed by team members because there is no open communication channel to senior management" (Grey, 2011; Thejendra, 2014). Kappelman, McKeeman and Zhang (2006) noted that in terms of control and maturity, the quality, budget, schedule and scope was not controlled by the development team (Kappelman, McKeeman and Zhang, 2006). In addition, an upfront negotiation and mismanagement of client expectation resulted in failure where delivery dates negotiated up front had a direct impact on the development process (Verner, Sampson and Cerpa, 2008).
- External influences: Grey (2011) through Thejendra (2014) found that in certain cases, failure was attributed to external factors, such as vendor and regulatory requirements and influences (Grey, 2011; Thejendra, 2014).
- Ownership, planning, value and impact: Research suggested that weak business practices such as due diligence, return on investment and impact analysis resulted in a weak demand for the delivered solution. Holgeit (2013) in referencing Kappelman et al. (2006) noted that a lack of a business case for the project was one of the "most influential process-related risks" of a development initiative (Kappelman, McKeeman and Zhang, 2006; Holgeid, 2013). In addition, delivery decisions were made without considering the requirements, and in certain cases, no business ownership was taken of the development project.

Inadequate focus – Organizational factors

- Commitment: Commitment, in terms of execution and process was the largest contributor towards the failure of software development initiatives. This dimension included a lack of buy-in from project stakeholders, no written management commitment, conflicting interests or cultures, a total lack of executive support, sponsor commitment not lasting the duration of the project and finally poor to no reviews of the outcome, or where stakeholders were engaged, slippages were experienced due to delays in securing a decision. In terms of team formation, commitment was lacking as project resources failed to commit to the project due to multiple project assignments. In addition, commitment extended to users or customer involvement. In certain cases, user involvement was not secured for the duration of the project, was not involved in schedule estimates, was not willing to cooperate or did not make enough time for requirements gathering.
- Conflict and commitment: Commitment extended to communication, expectation management as well as the team. The literature revealed that major contributors to software development failure included conflict between user departments, stakeholder conflict regarding a lack of alignment with project stakeholders due to unrealistic or unmanaged stakeholder expectations. It was also found that conflict existed between the organisations involved in the software projects due to little support from management. Also, project expectations were not managed as no project communication plans were created and no resources were devoted to managing communications between parties.
- Environmental & organisational maturity: Kappelman et al. (2006) found that project team productivity was severely hampered due to IT operation maturity as infrastructure and networks were not available or adequate to support the development (Kappelman, McKeeman and Zhang, 2006). In terms of execution, a

poor organisational maturity resulted in project managers not being able to control the project as they were not given full authority to manage the project.

- Focus: It was found that a lack of business focus resulted in a mismatch between the cost of the solution and the value the solution was meant to provide, ultimately leading to a solution that was not needed any longer. In addition, a lack of focus in terms of the team management led to project team members being overscheduled as a unified vision and objectives of the solution was not clear.
- Maturity: In terms of communication, control and organisational environment, several factors were found as primary contributors to software development failure. These included a total breakdown in communication between project stakeholders due to different and unrealistic expectations, senior management impacting the development initiative due to an insufficient organisational structure or an unstable organisational environment due to restructuring or changes in senior management.

  It was found that in terms of a cost/value proposition a low organisational maturity contributed towards failure as projects were approved based on "intuitive, personal or political reasons, instead of conducting the necessary assessments" (Lock, 2007; Grey, 2011). Kappelman et al. (2006) suggested that in certain cases failure was attributed to a lack of contingency budget required to manage known risks and the rate of change (Kappelman, McKeeman and Zhang, 2006).

  It was noted that in some instances, morale was severely affected due to low organisational maturity in terms of organisational politics and culture clashes. Grey (2011) in referencing Lientz and Rea (2003) noted team morale was reduced as the personal interests of team members were not taken into consideration (Lientz and Rea, 2003; Grey, 2011). Staff was not appreciated for the efforts they were putting into completing the project, even though the schedule affected their personal lives. Additionally, a certain amount of abuse was experienced by subject matter experts

where they were required to retain their prior duties, whilst providing substantial participation in the software development initiative.

In addition to a general conflict of interests in the development team, team members did not have defined roles and responsibilities and finally, the total lack of organisational maturity in terms of change management, placed a strain on the users and technical support team who felt threatened by a project aimed at replacing their legacy system.

In terms of team and execution, the literature revealed that teams experienced a lack of resources (time, money and expertise), had inappropriate structures and poorly defined roles, responsibilities and authority. Verner et al. (2008) noted that project execution was hampered where more developers were added to the project team in an attempt to increase implementation and to meet aggressive project schedules (Verner, Sampson and Cerpa, 2008).

Finally, when project control and implementation was considered, it was found that poor skills and skill shortages resulted in weak managers with poor and inadequate leadership capabilities. Project sponsors did not have the appropriate skills and team managers and team members alike were not provided with adequate training to hone their abilities. Grey (2011) in referencing Charvat (2003) noted that the former resulted in mistakes that had previously led to failure being repeated (Charvat, 2003; Grey, 2011).

Internal variability

Problem with system content – Project management factors

- Communication: In addition to several authors, Ahmadzai (2016) noted that poor communication was a critical factor responsible for software development failure (Ahmadzai, 2016). Team misalignment due to improper communication resulted in the inability to act as a team, and El Emam and Koru (2008) suggested that inappropriate management skills were responsible for a misalignment between IT and business (El Emam and Koru, 2008). Also, the literature suggested that insufficient critical project communication ultimately failed to manage the end-user's expectations (Al-Ahmad *et al.*, 2009).
- Control: In terms of execution, quality and scope, it was found that software development initiatives experienced significant goal, scope and schedule changes, immediately after the project was started. It was suggested that poor control and a total lack of managing changing user requirements and scope during execution as well as changing objectives and goals that were often unarticulated, vague or blurred had major impacts on the quality of the delivered solution or even the failure to provide the solution at all.
- Estimation: From a budget perspective, Dalal and Chhillar (2012) suggested "institutional or opinionated pressures that encourage unrealistically low bids, unrealistically low budget requests, and underestimates of time requirements" as a common cause of software development failure (Dalal and Chhillar, 2012). In addition to several authors, Verner et al. (2008) noted that customer and users did not have realistic expectations (Verner, Sampson and Cerpa, 2008) when estimations were considered.
- Focus and maturity: Lock (2007) found that software development projects failed or neglected to deliver all the required functionality as project managers gave

insufficient attention to the management of cash flow and fund provisioning (Lock, 2007). Also, project managers were found to abandon the development initiative under pressure resulting in stakeholder concerns and interests being neglected.

- Planning: In terms of budget, execution, morale and quality, it was found that enthusiastic or optimistic estimations were taken literally and became the projects' execution time. As such the project schedule became too optimistic and, in some cases, estimations were not revisited as new information became available resulting in an aggressive project schedule affecting team morale. In addition, it was found that when major risks were identified after the project had started, testing time was sacrificed to avoid late delivery, no robust project delivery plans were created, schedule deadlines were not reconciled to the project schedule and finally limited to no planning and estimation documentation was created.

  In terms of risk, it was found that risks were not identified at the start of the project and project risk assessments were not conducted. Where it was done, risks were not incorporated into the project plan and were not controlled. Holgeit (2013) noted that insufficient risk management was in the top ten contributors of the most influential process-related failure factors of software development initiatives (Holgeid, 2013).

  In considering project scope and in addition to ineffective planning, task assignment and team direction, Summers (2015) suggested that insufficient sizing in terms of poor estimation and scheduling was one of the top five contributors towards software development failure (Summers, 2015).
- Process: In terms of control and project execution it was found that an earned value system was not in place to monitor execution, no project status progress processes were in place, and no change control processes were created. In addition, no project charter document was developed during the early phases of the project. When considering communication and transparency, it was found that no performance and

reliability requirement metrics were available, nor were milestones and delivery due dates documented. Finally, project success criteria were not defined and documented.

- Quality: When considering the quality of scope, Grey (2011) in referencing Melton and Loch (2007) noted that the project scope was not robust and understandable (Lock, 2007; Melton, 2007; Grey, 2011). The author further suggested that poor, unclear and misunderstood scope and objectives were a general reason for the failure of software development projects.
- Skill: The literature revealed that estimation techniques were inadequate, and estimations were either done by non-development staff, or by overly optimistic developers. Also, it was found that status reporting was either incorrect or optimistic and lacked success and failure criteria. Project planes were found to be ineffective, and projects were not re-planned when requirements changed. Kappelman et al. (2006) and Schmidt et al. (2001) noted that project managers were poorly trained and in certain cases did not have the experience that is gained from managing a large-scale project (Schmidt *et al.*, 2001; Kappelman, McKeeman and Zhang, 2006). The authors further noted that deliverable due dates were missed as early as in the first ten percent of the project schedule. Summers (2015) noted as part of his top five reasons for software development failure that quality assurance was "short changed" as the testing phase was often cut short to accommodate time pressures (Summers, 2015).

  When considering expectation management, it was found that customer and user expectations were not managed and stakeholders were not correctly identified in terms of their influences and interests in the project.

  It was also found that project managers lacked general management skills. Project problems were identified too late; early project delays were ignored, and project plans were not revised. Also, project managers were unable to lead the development team efficiently and communicate with clients.

When planning and process were considered, it was found that project scheduling was unrealistic and inaccurate as task time was used as schedule time. In some cases, planning was severely neglected. It was also found that planning tools were inappropriate or insufficiently utilised, key stakeholders were not included in planning and reviewing the solution, and finally, planning was insufficient as the project manager jumped straight into delivering the solution.

Finally, when considering quality, it was found that there was a failure to align with the constituents and stakeholders and a late realisation of warning signals with a refusal to recognise or admit that the project was in trouble, suggesting responsive project planning.

- User involvement: When considering requirements, it was found that the requirements highlighted poor user involvement, resulting in questionable requirements (Gulla, 2011). Al-Ahmad et al. (2009) concluded that there was a failure to gain user commitment as well as a lack of user involvement (Al-Ahmad *et al.*, 2009).

Problem with system content – Complexity or size factors

- Complexity and size: System complexity was found to be a primary driver for software development project failure. Kappelman et al. (2006) noted the system complexity in terms of difficulty in determining the inputs and outputs of the system (Kappelman, McKeeman and Zhang, 2006). Additional complexity factors included scores of interfaces to other systems and the project involved implementing a custom or beta version of the system. From a size perspective, it was found that complexity increased as the solution was multifaceted and included multiple integration points. This phenomenon resulted in an inability to handle the project's technical complexity.

- User involvement: From a system quality perspective, it was found that reasons for failure included situations where users could not be involved in determining the quality attributes of the system due to a lack of understanding of the capabilities of the new system.

Skills – Technology factors

- Business Analysis skills: Understanding and translating business requirements is believed to be a critical success factor in software. As such, failure to correctly define a solution in technical terms is a primary driver for software failure. It was found that project stakeholders were not interviewed to get a better understanding of the requirements. As such, no clear and robust business cases and project definitions were created. In addition, Grey (2011) in referencing Hedeman and Heemst (2010) noted that failure in certain cases was attributed to the production of poorly produced deliverables (Frederiksz, Hedeman and Heemst, 2010; Grey, 2011). It was also found that functional, performance, and reliability requirements, as well as scope, were not documented. Finally, business strategy alignment was not documented.
- Development skills: It was found that from a technical perspective, developers lacked the technical experience to fulfil the requirements appropriately. Also, it was discovered that no team members had experience with the chosen technology and the team had poor technical designs and practices. In terms of quality, it was concluded that the development team had too high a reliance on the technology, was prone to introducing new and immature technologies, had skills that did not match the job and finally lack design and technical experience.
- Team skills: From an execution and quality perspective, it was found that the tooling used in the unit was not appropriate for the development and in certain cases, teams lacked the experience to use the selected tools and techniques successfully. The Standish Group (1995) reported that technology illiteracy was one of the top ten

factors for software development impairment (The Standish Group, 1995; Kessler, 2001; Smith, 2002). From a testing perspective, it was found that incorrect steps to reproduce a problem, as well as improper fault assignment, were responsible for a generally inferior quality of testing. When architecture was considered, it was found that technology evaluations were neglected or were insufficiently done.

Problem in execution – Process factors

- Commitment, control and maturity: Verner et al. (2008) found that projects were underestimated as the developers of the team were not involved in providing requirement estimations (Verner, Sampson and Cerpa, 2008). It was also found that quality control was lacking and change control was not monitored effectively. In terms of process maturity, the execution was riddled with poor and sloppy development practices.
- Method: Methodology and the incorporation of frameworks and practices were found to be lacking. Verner et al. (2008) noted that development methodologies were inappropriate for use (Verner, Sampson and Cerpa, 2008) and in certain cases, no clear methodology was followed. It was also found that no project management methodology was used, and project managers were not involved in sizing and estimation. In addition, it was concluded that there was a lack of change, risk, financial and performance management and measurement frameworks. Additional approaches such as a method for requirements gathering, review gates at the end of each phase and a limited approach to quality and acceptance criteria resulted in a lack of delivering benefit and value to the organisation.
- Skill: When considering execution in terms of process it was found that there was a misalignment to the approaches and practices within the organisation. Grey (2011) in referencing Charvat (2003) suggested that “in practice project management theory is not executed correctly” and the “intended project management strategy is inappropriate” (Charvat, 2003; Grey, 2011).

2.3.5 Summary

How do software development teams know when they have succeeded? When considering both pragmatic and idealistic approaches to software development, it appears that there is a divide when success and failure are considered. The Waterfall method (pragmatic) suggests that success is measured through a balance and compliance to the triple constraint of time, scope and budget. However, Scrum, an Agile approach (idealistic) to managing software development initiatives, focus on the concept of working software and suggests that that the iron triangle may be inconsequential as time and scope can be managed by short iterations.

It is suggested that neither Waterfall nor Scrum eliminates differences in expert perspective when factors that influence software development outcomes are considered. The literature revealed 209 distinct factors believed to have fuelled software failure. It was noted that the failure list could be considered incomplete as it was reasoned that no study would have the appetite to produce a complete list.

It was however found that the reasons collected from empirical studies could be divided into two categories. Factors believed to have an impact on the outcomes of the development initiative but were, under the control of the software development team or, outside the control of the software development team. As such, the former was referred to as “internal variability” and the latter as “political impacts”.

Table 2 Failure reason roll up

	Problem	Impact	Categorisation	Reference
• Commercial pressures or cost/value benefit. • Approved project budget less than budget estimated by the project team	Commercials	Cost/Value, Management	Management	(Charette, 2005; Henderson, 2006; Kappelman, McKeeman and Zhang, 2006)
• Lack of IT Management, lack of executive management commitment with no-one to champion the project. • Lack of top management support or commitment to the project. • Key stakeholders have not signed the project charter. • No due diligence on vendor(s) and team members	Commitment	Execution, stakeholder management, process maturity	Management	(Schmidt *et al.*, 2001; The Standish Group, 2001; Smith, 2002; Kappelman, McKeeman and Zhang, 2006; Al-Ahmad *et al.*, 2009; Gulla, 2011; Krigsman, 2012a)
• Business decisions made by senior management are commands to be executed, which results in problems and issues being addressed by team members because there is no open communication channel to executives. Budget, schedule, scope, and quality all mandated from outside the project team. Delivery date affected development process. Lack of or inadequate change management	Communication, Control, Maturity	Execution	Management	(Kappelman, McKeeman and Zhang, 2006; Verner, Sampson and Cerpa, 2008; Young, Brady and Nagle, 2009; Thejendra, 2014)
• External factors, such as vendor and regulatory issues	External	Control	Management	(Thejendra, 2014)
• No executive ownership. • No business case for the project. • Return on investment assessment is not performed. • No real demand for the products that were to be delivered by the project. • Delivery decision not made with appropriate requirements info. • Inadequate impact analysis was done to determine the impact the project could have had on the organisation and business processes	Ownership, Planning, Value, Impact	Cost/value, Execution, Organisation	Management	(Kappelman, McKeeman and Zhang, 2006; Verner, Sampson and Cerpa, 2008; Young, Brady and Nagle, 2009; Frederiksz, Hedeman and Heemst, 2010; Grey, 2011)
• No buy-in or support from the project stakeholders. • No written commitment for the project outside of the project team. • Team commitment due to conflicting interests or cultures. • Lack of top management support. • Project team members have a weak commitment to the project scope and schedule. • Project resources have been assigned to a higher priority project. • Key team member turnover after project kickoff. • Low level of customers/user involvement. • Involved customers/users did not stay throughout the project. • Customers/users not involved in schedule estimates. • High customer/user turnover. • Customers/users did not make enough time for requirements gathering. • Users are not willing to cooperate. Sponsor commitment did not last throughout the project. • Key stakeholders do not review and sign off deliverables on a timely basis.	Commitment	Execution, Process	Organisation	(The Standish Group, 1995; Smith, 2002; Kappelman, McKeeman and Zhang, 2006; Westland, 2007; Verner, Sampson and Cerpa, 2008; Al-Ahmad *et al.*, 2009; Krigsman, 2012a)

	Problem	Impact	Categorisation	Reference
• Project stakeholder decision delays have caused due dates to be missed. Lack of Executive Support				
• Conflict between user departments. • No stakeholder investment or alignment or stakeholder conflict regarding unrealistic and mismatched expectations. • Cultural conflict among organizations involved. • No project communications plan or resources devoted to managing and communicating project expectations. • Little support from senior management	Conflict, Commitment	Communications, Expectation management, Team	Organisation	(May, 1998; Schmidt *et al.*, 2001; Charette, 2005; Henderson, 2006; Kappelman, McKeeman and Zhang, 2006; Frederiksz, Hedeman and Heemst, 2010; Bloch, Blumberg and Laartz, 2012; Krigsman, 2012b)
• IT operations infrastructure and network infrastructure problems have a major impact on project team productivity. • Project manager did not control the project. • PM not given full authority to manage project	Environments, Maturity	Execution	Organisation	(Kappelman, McKeeman and Zhang, 2006; Verner, Sampson and Cerpa, 2008)
• The client did not need the solution any longer • Lack of business focus	Focus	Cost/value, Execution, Organisation	Organisation	(The Standish Group, 1995; Smith, 2002; Henderson, 2006; Bloch, Blumberg and Laartz, 2012)
• Project team members are overscheduled. • No clear or unified vision and objectives	Focus	Team, Vision	Organisation	(Hayes, 2002; Kappelman, McKeeman and Zhang, 2006)
• Communication breakdown among project stakeholders. Insufficient organisational structure. • Stakeholders have different and unrealistic expectations. • Senior management negatively impacted project. • Unstable organisation environment (such as changes in senior management or restructuring)	Maturity	Communications, Control, Cost/value, Environment	Organisation	(Kappelman, McKeeman and Zhang, 2006; Verner, Sampson and Cerpa, 2008; Young, Brady and Nagle, 2009; Thejendra, 2014)
• Projects are approved based on intuitive, personal or political reasons, instead of conducting the necessary risk assessment. • No contingency budget for known risks and rate of changes	Maturity	Cost/Value	Organisation	(Kappelman, McKeeman and Zhang, 2006; Lock, 2007)
• Organizational politics and culture clashes. • Personal interests are not taken into consideration. • Staff not appreciated for working long hours. • Staff not rewarded for working long hours. • Project team member(s) have low morale. • Schedule affected team members' lives	Maturity	Morale	Organisation	(Lientz and Rea, 2003; Kappelman, McKeeman and Zhang, 2006; Verner, Sampson and Cerpa, 2008; Thejendra, 2014)
• Subject matter experts are overscheduled: retain all prior duties yet expected to provide substantial participation to the project. • Users or technical support team feel threatened by a project to replace their legacy system. • Team members have undefined roles and responsibilities. • Team conflicting interests	Maturity	Team	Organisation	(Kappelman, McKeeman and Zhang, 2006; Al-Ahmad *et al.*, 2009)

	Problem	Impact	Categorisation	Reference
• Lack of Resources. • More developers increase execution. • Inappropriate team organisation. • Poorly defined roles, authority and responsibilities. • Resource shortages (money, time, expertise). • The project was not adequately staffed. • PM changed during the project. • Staff added late to meet the aggressive schedule. • Inappropriate team structure or an improper definition of roles and responsibilities. • Insufficient or inappropriate team staffing. • Lack of Resources	Resources	Execution, Team	Organisation	(The Standish Group, 1995; Schmidt *et al.*, 2001; Hayes, 2002; Smith, 2002; Verner, Sampson and Cerpa, 2008; Al-Ahmad *et al.*, 2009; Buckler, 2010; Frederiksz, Hedeman and Heemst, 2010; Kaur and Sengupta, 2011; Krigsman, 2012a; Rajkumar and Alagarsamy, 2013)
• Weak managers. • Poor or inadequate leadership. • Project managers are not given training for developing their project management skills which result in the same mistakes being repeated. • Project sponsors lack experience. • Project team members do not have required knowledge/skills	Skill	Control, Execution	Organisation	(Charvat, 2003; Kappelman, McKeeman and Zhang, 2006; Young, Brady and Nagle, 2009; Kaur and Sengupta, 2011; Thejendra, 2014)
• Insufficient critical project communication. • Team misalignment due to improper communication failing to act as a team. • Poor communication. • Failure to manage end-user expectations	Communication	Execution, Expectation management	PM	(Schmidt *et al.*, 2001; Charvat, 2003; Charette, 2005; Dorsey, 2005; Henderson, 2006; Westland, 2007; Gulla, 2011; Kaur and Sengupta, 2011; Oracle Corporation, 2011; Bloch, Blumberg and Laartz, 2012; Dalal and Chhillar, 2012; Rajkumar and Alagarsamy, 2013)
• Significant goal, scope, or schedule requirements change immediately after project kickoff. • Poor control during project delivery. • Poor quality control. • Lack of management of changing user requirements and scope during execution. • Objective and goals are unarticulated, vague or blurred and often change	Control	Execution, Quality, Scope	PM	(May, 1998; Schmidt *et al.*, 2001; Charette, 2005; Henderson, 2006; Jones, 2006; Kappelman, McKeeman and Zhang, 2006; Melton, 2007; Al-Ahmad *et al.*, 2009; Young, Brady and Nagle, 2009; Buckler, 2010; Kaur and Sengupta, 2011; Bloch, Blumberg and Laartz, 2012; Dalal and Chhillar, 2012; Rajkumar and Alagarsamy, 2013)
• Inaccurate - unrealistically low bids and budget requests and an underestimation of time requirements.	Estimation	Budget, Quality	PM	

	Problem	Impact	Categorisation	Reference
• Customers/users did not have realistic expectations				
• Insufficient attention is given to the management of cash flow and fund provision	Focus	Management	PM	(Lock, 2007)
• Poor management of scope. • Inadequately dissected and poorly scoped projects. • Project scope changes and scope creep. • Changing Requirements & specifications. Ineffective change control because of evolving business and user requirements	Management	Scope	PM	(The Standish Group, 1995; Smith, 2002; Charvat, 2003; Young, Brady and Nagle, 2009; Buckler, 2010; Frederiksz, Hedeman and Heemst, 2010; Oracle Corporation, 2011)
• PM abandoned plan under pressure. • Unrealistic Expectations. • Stakeholder concerns and interests are neglected	Maturity	Execution, Expectation Management, Stakeholder	PM	(The Standish Group, 1995; Smith, 2002; Lock, 2007; Verner, Sampson and Cerpa, 2008)
• No planning and estimation documentation. • Enthusiastic/optimistic estimations were taken literally and became execution time. • Cost and project schedule estimates are too optimistic, and they are not revised as the project progresses and new information becomes available. • Major new risks are identified after the project kickoff. • No robust project delivery plan. • Schedule deadline not reconciled to the project schedule. • Aggressive schedule affected team motivation. • Testing time was sacrificed to avoid late delivery.	Planning	Budget, Execution, Morale, Quality	PM	(Charvat, 2003; Kappelman, McKeeman and Zhang, 2006; Lock, 2007; Melton, 2007; Verner, Sampson and Cerpa, 2008; Buckler, 2010)
• Incomplete project risk assessment is conducted. • Risks not identified at the start of the project. • Risks not incorporated into the project plan. • Risks not reassessed, and controlled. • Insufficient risk management; this involves identifying and mitigating anything which may impact upon the project	Planning	Risk	PM	(Nelson, 2005; Verner, Sampson and Cerpa, 2008; Grey, 2011)
• Insufficient project sizing. • Poor estimating or scheduling; this involves scoping the project and estimating the time and effort required	Planning	Scope	PM	(Nelson, 2005; Grey, 2011)
• Ineffective planning and task assignment and direction	Planning	Team	PM	(May, 1998; Dorsey, 2005; Henderson, 2006; Gulla, 2011; Nawi, Rahma and Ibrahim, 2014)
• No project status progress process. • Earned value systems not in place or used to control program. • No change control process. • No project charter document at an early stage of the project. • No risk analysis documentation and process. • No performance and reliability requirements and metrics tracking process.	Process	Communications, Control, Execution	PM	(The Standish Group, 1995; Smith, 2002; Kappelman, McKeeman and Zhang, 2006)

	Problem	Impact	Categorisation	Reference
• No documented milestone deliverables and due dates. • Undefined project success criteria, • Lack of User Involvement				
• No robust and understandable project scope. • Poor project scoping. • Unclear / misunderstood scope and objectives	Quality	Scope	PM	(Schmidt *et al*., 2001; Lock, 2007; Melton, 2007; Al-Ahmad *et al.*, 2009; Grey, 2011)
• Inadequate estimation techniques or estimated by non-developers or overly optimistic developers	Skill	Budget	PM	(Dorsey, 2005; Buckler, 2010; Rajkumar and Alagarsamy, 2013)
• Incorrect and Optimistic Status Reporting with a lack of success and failure criteria. Inadequate budgets and unrealistic timelines. • No defined checkpoints and structure. • No effective project plan. • Project not re-planned after requirements changes. • Project managers have poor training. • Project manager(s) have never managed a project of this scale before. • Deliverable due dates missed during the first 10 percent of the project schedule. • Shortchanged quality assurance; this comes about when projects come under pressure and thus testing, and training are often cut	Skill	Execution	PM	(Charette, 2005; Nelson, 2005; Jones, 2006; Kappelman, McKeeman and Zhang, 2006; Verner, Sampson and Cerpa, 2008; Frederiksz, Hedeman and Heemst, 2010; Krigsman, 2012a; Thejendra, 2014)
• Customer/user expectations not managed. • Unrealistic client and business expectations. • Ineffective stakeholder management; this involves identifying stakeholders and their influence and interest in the project and then managing their expectations accordingly	Skill	Expectation management	PM	(Hayes, 2002; Nelson, 2005; Verner, Sampson and Cerpa, 2008)
• Project problems are identified too late. • Weak project manager. • Project manager(s) cannot effectively lead the team and communicate with clients. • Early project delays are ignored — no revision to the overall project schedule. • Lack of effective project management skills	Skill	Management	PM	(Schmidt *et al*., 2001; Kappelman, McKeeman and Zhang, 2006; Young, Brady and Nagle, 2009)
• Unrealistic or inaccurate scheduling using task time as schedule time. • Insufficient utilisation of planning tools. • No or poor planning. • Plans are not used to guide the project or are not executed correctly. • Inappropriate PMM is followed. • Lack of Planning. • Insufficient planning; this happens when project managers jump straight into delivery. • Key project stakeholders do not participate in major review meetings. • Poor capacity planning	Skill	Planning, Process	PM	(The Standish Group, 1995; Hallows, 1998; Smith, 2002; Hayes, 2002; Charvat, 2003; Nelson, 2005; Jones, 2006; Kappelman, McKeeman and Zhang, 2006; Grey, 2011; Kaur and Sengupta, 2011; Bloch, Blumberg and Laartz, 2012; Rajkumar and Alagarsamy, 2013; Thejendra, 2014)

	Problem	Impact	Categorisation	Reference
• Failure to align with constituents and interested parties and a late realisation of warning signals with a refusal to recognise/admit the project are in trouble. • Responsive project planning	Skill	Quality	PM	(May, 1998; Charette, 2005; Jones, 2006; Gulla, 2011; Oracle Corporation, 2011; Bloch, Blumberg and Laartz, 2012)
• Poor user input. • Failure to gain user commitment or inadequate user involvement. • No user involvement and commitment from project initiation	User involvement	Requirements	PM	(May, 1998; Schmidt *et al.*, 2001; Al-Ahmad *et al.*, 2009; Young, Brady and Nagle, 2009; Frederiksz, Hedeman and Heemst, 2010; Kaur and Sengupta, 2011; Rajkumar and Alagarsamy, 2013)
• Developers not involved in estimates • Project underestimated	Commitment	Team	Process	(Verner, Sampson and Cerpa, 2008)
• Inadequate Quality Control. • Change control not monitored effectively	Control	Quality	Process	(Jones, 2006; Verner, Sampson and Cerpa, 2008; Kaur and Sengupta, 2011)
• Sloppy development practices	Maturity	Quality	Process	(May, 1998; Schmidt *et al.*, 2001; Charette, 2005; Dorsey, 2005; Al-Ahmad *et al.*, 2009; Gulla, 2011; Bloch, Blumberg and Laartz, 2012; Dalal and Chhillar, 2012)
• PM had no input into estimates. • Lack of change, risk, financial and performance management and measurement	Method	Budget, Control	Process	(Charette, 2005; Verner, Sampson and Cerpa, 2008; Gulla, 2011; Oracle Corporation, 2011)
• Inappropriate development methodology. • No clear methodology is defined and followed. • No project management methodology. • Avoidance of, lack of or suitability of effective project management methodology	Method	Development, Execution	Process	(Schmidt *et al.*, 2001; Dorsey, 2005; Kappelman, McKeeman and Zhang, 2006; Westland, 2007; Verner, Sampson and Cerpa, 2008; Al-Ahmad *et al.*, 2009; Gulla, 2011; Oracle Corporation, 2011)
• No reviews at the end of each phase. • No method for requirements gathering. • No quality and acceptance criteria. Lack of delivering benefits and value to the organisation	Practice, Process	Communications, Scope, Quality	Process	(Melton, 2007; Verner, Sampson and Cerpa, 2008; Frederiksz, Hedeman and Heemst, 2010)
• Misalignment within the internal organisation. • In practice, project management theory is not executed correctly. • Intended project management strategy is inappropriate. • Problems due to scores of customer/users involved	Skill	Execution	Process	(Charvat, 2003; Lock, 2007; Verner, Sampson and Cerpa, 2008; Grey, 2011)
• Difficulty in determining the input and output of the system. • A large number of interfaces to other system required.	Complexity	System	System	(Kappelman, McKeeman and Zhang, 2006)

	Problem	Impact	Categorisation	Reference
• Project involves implementing a custom or beta version of hardware or software				
• Users cannot get involved because of lack of understanding of new system capabilities	Requirements	User involvement	System	(Kappelman, McKeeman and Zhang, 2006)
• Large and multifaceted with lots of integration points. • Inability to handle the project's technical complexity	Size	Complexity	System	(Schmidt *et al.*, 2001; Charette, 2005; Al-Ahmad *et al.*, 2009; Bloch, Blumberg and Laartz, 2012)
• Project stakeholders have not been interviewed for project requirements. • No clear and robust business case or project definition. • Poor requirements definition and gathering. • Deliverables to be produced are poorly defined. • No clear understanding of the user and technical requirements and what must be done. • Poor requirement set due to badly defined or misunderstanding the user needs or failed to extract requirements appropriately. • The project did not have real requirements at any stage. • The scope of the project was not well-defined • Scope changes during the project. • Functional, performance, and reliability requirements and scope are not documented. • Failure to gather requirement via joint application design. • No documented analysis of business strategy alignment. • Incomplete Requirements	BA Skill	Scope	Technology	(The Standish Group, 1995; Schmidt *et al.*, 2001; Smith, 2002; Charette, 2005; Kappelman, McKeeman and Zhang, 2006; Westland, 2007; Lock, 2007; Melton, 2007; Verner, Sampson and Cerpa, 2008; Young, Brady and Nagle, 2009; Frederiksz, Hedeman and Heemst, 2010; Grey, 2011; Kaur and Sengupta, 2011; Rajkumar and Alagarsamy, 2013)
• Poor technical designs and practices. • Lack of technology experience. • No team member experience with the chosen technology	Dev Skill	Quality	Technology	(Hayes, 2002; Kappelman, McKeeman and Zhang, 2006; Thejendra, 2014)
• Too high a reliance on technology. • Introduction of new or immature technology, skills do not match the job, lack of design, technical experience	Maturity	Quality	Technology	(Hallows, 1998; May, 1998; Schmidt *et al.*, 2001; Charette, 2005; Dorsey, 2005; Al-Ahmad *et al.*, 2009; Gulla, 2011; Bloch, Blumberg and Laartz, 2012; Dalal and Chhillar, 2012)
• Inappropriate tooling. • The team did not have adequate experience with tools and techniques. • Technology Illiteracy. • Incorrect steps to reproduce and improper fault assignment or a poor quality of testing	Team Skill	Execution, Quality	Technology	(The Standish Group, 1995; Smith, 2002; Dorsey, 2005; Verner, Sampson and Cerpa, 2008; Kaur and Sengupta, 2011; Dalal and Chhillar, 2012; Rajkumar and Alagarsamy, 2013)
• Poor testing of product developed	Test skill	Quality	Technology	(Charvat, 2003)

	Problem	Impact	Categorisation	Reference
• No single requirements repository for requirements • Size of project affected requirements elicitation	Tools	Control	Technology	(Verner, Sampson and Cerpa, 2008)
• Insufficient evaluation of technology	Arch Skill	Quality	Technology	(Grey, 2011)

This section provides an overview of the human factor in software development but with a specific focus on cognitive bias. A level of understanding is provided in terms of the theoretical concept of cognitive bias, but specifically as it relates to experts in the software development industry. This section may prove to be interesting to all reader types but the description may add the most value to pragmatic readers.			

2.4 The Human factor and cognitive bias

In attempting to define the meaning of success, Thomas and Fernández (2008) suggested that "success and failure are difficult to define and measure since they mean different things to different people" (Thomas and Fernández, 2008). The authors noted that perception was primary when success or failure in terms of social accomplishment was considered. People play a significant and essential part in developing software systems, and as such, *reasons for failure or success* of software development projects were found to be *subjective* (Heeks & Bhatnagar, 1999) and resulted in an *assortment of divergent perspectives*, all *based on interest* (Karch, 2011).

Rosqvist et al. (2003) in defining a model where expert judgement could be used as a driver for software quality noted that "the assessment of the compliance of a software product with stated requirements is, in many cases, subjective and dependent on the assessor's past experience" (Rosqvist, Koskela and Harju, 2003). The authors suggested that experts had their own "mental model" of what was believed to be the quality of the system based on artefacts they obtained during the development process. "*This suggests that the evaluation of software quality may be based on expert group decision-making rather than empirical evidence alone*" (Ibid.). Rosqvist, Koskela and Harju (2003) concluded that the challenges for methods using expert judgement included empirical evaluation of expert probability judgement, as well as the principle of impartiality, where expert biases could not be questioned (Ibid.).

Ouchi (2004) suggested that expert judgement was complex and subjective by nature. The author noted that "there has been no formally established methodology for treating expert judgment" and in referencing Winkler et al. (1992) and von Winterfeld (1998) pointed out that "in recent years, there has been increasing effort in establishing a more systematic approach to eliciting expert opinion" (Ouchi, 2004).

Yudkowsky (2008) suggested that "human beings use methods of thought—heuristics—which quickly return good approximate answers in many cases; but which also give rise to systematic errors called biases" (Yudkowsky, 2008).

Skjong and Wentworth (2000) in a paper on expert judgement in risk assessment noted that heuristics, a theoretical framework based on the work of Tversky and Kahneman (1974), was used to explain risks in terms of "intuitive mental rule-of-thumb" for judging everyday events. The authors further noted that even though this approach was used in daily life, it was also employed by decision-makers and experts when consulted on risk-related assessments and stated that "heuristics may yield quick, intuitive insights, but may also result in dramatic lapses of logic" (Skjong and Wentworth, 2000). Skjong and Wentworth (2000) explained that heuristics could be grouped into six categories, including structured bias, suggesting that experts were influenced by the manner a problem was structured before it was presented to them and motivational bias, suggesting that experts were influenced based on the stake they had in the outcome of the analysis. Burgman et al. (2006) suggested that "people distort interpretation of decision criteria and the evaluation of information to justify a predetermined choice, even when engaged in evaluating information in an attempt to be objective" (Burgman *et al.*, 2006). The authors noted that people were inclined to settle on an outcome that they favour, but required superficially defensible evidence to justify their conclusions in an attempt to maintain an illusion of objectivity. The authors suggested that individuals required defensible evidence not to appear unquestionable, but rather that they "don't realise that they are accessing a biased subset of the available information" (Ibid.).

In addition to motivational bias, Baddeley, Curtis and Wood (2004) suggested cognitive bias as an additional type of individual prejudice (Baddeley, Curtis and Wood, 2004). The authors noted that cognitive biases were more problematic than motivational biases as they develop from the incorrect processing of information and as such are not under conscious control. Also, cognitive biases were found to be due to using heuristics as a "common-sense device" to make a quick decision in an uncertain situation (Ibid.). The authors in referencing Tversky and Kahneman (1974) noted that "At least four types of heuristics

that produce cognitive bias are commonly employed: availability, anchoring and adjustment, representativeness, and control" (Ibid.)(Baddeley, Curtis and Wood, 2004). Availability was described as a heuristic for assessing the probability of an event regarding the ease with which the event was recalled. The authors suggested that this approach could introduce bias due to the importance of particular events, rather than their frequency. Anchoring and adjustment were explained as a heuristic where an initial estimate was based on an estimation of a probability that was then revised when new information was available. The authors noted that this approach was biased based on the importance of the initial or anchor value. Control was defined as the illusion of control on the outcome people have in situations they cannot control and representativeness, where people estimate probability based on the similarity of two events. Baddeley, Curtis and Wood (2004) suggested that representativeness heuristics were biased in that "the belief that when a series of trials have all had the same outcome then the opposite outcome is more likely to occur in the following trials, since random fluctuations seem more representative of the sample space" (gambler's fallacy) (Ibid.).

This section concludes the literature review by restating some of the important aspects highlighted in the text such as the history of the software industry, the software crisis and how it relates to modern software development, reasons for software success and failure as well as cognitive bias, thus suggesting that it is appropriate for all reader types.





2.5 Summary

The literature review revealed that there were several opinions about the state of software engineering, especially when the software crisis was considered. The review tested the general opinion of whether the industry was in crisis and found that due to the interpretation of crisis in terms of software engineering, the crisis as it was defined in 1968 may have morphed from its initial incarnation but persisted over the years. In uncovering the state of the industry, it was found that authors in the field of software engineering had a negative sentiment towards the state of affairs, with several authors suggesting a software-fuelled apocalypse in the very near future.

The literature review continued in unpacking software development outcomes in terms of failure and success and found that even though measures existed to some degree, they appeared to be misused. In addition, an extensive investigation was done to understand what reasons contributed towards the success and failure of software development projects. The literature review incorporated both formal and informal sources and revealed more than three hundred reasons why software projects fail. It was however noted that the list that was uncovered was by no means complete as there was a vast amount of available literature on reasons for software development failure in the existing bodies of knowledge. It was found that extensive research into understanding factors that influence the success and failure of a software development project had not been able to provide conclusive answers as to why some software development projects succeed, while others fail. It was suggested that a research study dedicated to consolidating all failure factors might never be able to conclude with a consolidated failure factor list. It was however found that that existing research suggested project management failure as the largest contributor towards the failure of software development projects, but it could be argued that this phenomenon was largely related to the fact that the area of project and program management was the largest contributor to uncovering reasons for software failure.

In understanding software outcomes, the literature review revealed several classification frameworks capable of classifying failure factors in terms of the primary contributors to software failure. In understanding these classifications, a research categorisation was defined that suggested that failure reasons could be categorised as internal to the software development team, meaning a team had control over the "influence" and external to the team, meaning that the team had no control over the "influence".

Finally, the literature review investigated what was considered primary to the study, the human factor and what effects expert judgement had on software engineering. The review provided insight into cognitive bias introduced into answers constructed from expert judgement. The study incorporated work that was done on the original contributions of Tversky and Kahneman on heuristics and found several classifications of bias that were introduced when decisions were based on heuristics. The literature suggested that there were severe risks in using heuristics or expert judgement when deciding and suggested several frameworks that could be used to limit the impact of bias, should expert judgement be the only form of data available.

There was, however, no work that could be found that highlighted the impact expert judgement had on the outcomes of software development outcomes, meaning that the research associated with this dissertation is well-suited to fill this gap in research.

3 Theoretical foundation

This section provides a background to the theoretical foundation of the research and describes the overall research approach as it relates to theory creation in quantitative research. This section is focused on establishing a theoretical viewpoint and as such is considered most valuable to academic readers.





3.1 Preamble

A worldview has the potential to influence the practice of research and as such, should be identified. The term worldview could be approached from several perspectives, including paradigms, epistemologies or ontologies, or even broadly defined research methodologies. Creswell (2014) defined a worldview as a "basic set of beliefs that guide action" and suggested that this concept was a "general philosophical orientation about the world and the nature of research that a researcher brings to a study" (Creswell, 2014). This chapter describes a worldview or philosophy that underpins a theory, epistemology and design for conducting the research. Its inclusion was intended to inform the reader of the rigour and suitability of the methodology in relation to the existing bodies of knowledge the outcome is intended to augment. As such, the research is based on a quantitative study, supported by methods, theory and an appropriate design capable of providing an understanding of the perceptions of factors that influence software development outcomes between experts practising in different roles, generations and tenure.

Creswell (2014) suggested that "Certain types of social research problems call for specific approaches" (Creswell, 2014). The author further stated that "if the problem calls for (a) the identification of factors that influence an outcome, (b) the utility of an intervention, or (c) understanding the best predictors of outcomes, then a quantitative approach is best" (Ibid.). In addition, Creswell (2014) suggested that quantitative analysis was the best approach to test a theory or explanation. Sale, Lohfeld and Brazil (2002) suggested that quantitative research was associated with a positivist or post-positivist paradigm and was characterised by empirical studies (Sale, Lohfeld and Brazil, 2002, pp. 44–45). The authors state that for Quantitative research, "all phenomena can be reduced to empirical indicators which represent the truth" and "the ontological position of the quantitative

paradigm is that there is only one truth, an objective reality that exists independent of human perception" (Ibid.). Abeyasekera (2005) suggested that "Quantitative methods of data analysis can be of great value to the researcher who is attempting to draw meaningful results from a large body of qualitative data". The author further noted that "Quantitative analysis approaches are meaningful only when there is a need for data summary across many repetitions of a participatory process" (Abeyasekera, 2005). For this research, a Quantitative study was favoured as the rigour associated with this paradigm, had the faculty to highlight quality about the importance of the research phenomenon as well as the ability in generalising its findings across the software industry.

This section provides a complete understanding of the research phenomenon in terms of theory. This section provides a description of the assumptions made across the areas of work-related complexities, ways of working and finally social class. In addition, a theoretical premise is defined describing the research theory in relation to several propositions. A primary proposition is established and dependent and independent variables are uncovered for investigation. This section provides additional understanding in terms of the theory proposed for the research and as such is considered most valuable to academic readers.			

3.2 Research theory

The theory defined in this section was developed to understand the research problem and to assist in developing the research questions and hypothesis.

3.2.1 Background assumptions:

Evaluating and describing assumptions made to formulate a theory are not merely important in attempting to define the theory, but additionally, provide insight into phenomena that cannot be observed or were intentionally excluded from empirical evaluation. Moe (2017) suggested that for social science, "we make assumptions about the nature of human beings, social reality or a particular phenomenon" (Moe, 2017). The following assumptions were made in developing the theoretical view.

Work-related complexities

Work-related complexities define the apparatuses employed in the primary function of experts in a development team, such as the development languages or frameworks used by developers to produce source code. These apparatuses have inherent complexities associated with not just utilising them to complete tasks, but in learning and understanding them to a level where proficiency cannot be questioned. It was assumed that these complexities were equal for different apparatuses used between similar and differing roles in a development team.

Way of working

Different development teams utilise different approaches to complete their development tasks. Methodologies introduce various complexities development teams have to deal with, such as learning, understanding and implementing practices associated with the selected methodology. In addition, not all methodologies are equal, and in some cases, development teams may elect to use proprietary methods, bespoke methods or no method at all. As such, it was assumed that development teams are at a stage where the selected approach, methodology or practice was being used in accordance with its design and its appropriateness was not in question.

Social class

The foundation of this theory is social class, but the dimensions introduced by this element had the potential of over-inflating the research scope with little value. As such, only certain dimensions were selected as priorities for investigation, such as role, designation and level of experience. It is worth noting that other dimensions such as level of education had the potential of increasing the understanding of the phenomenon but incorporating this dimension would have introduced different aspects marginally related to the research problem.

3.2.2 Theoretical premise

The research study suggests that experts draw from their perspectives and knowledge to describe outcomes of software development initiatives, but perspectives create knowledge and knowledge augments perspective. As perspectives are based on social class, a theoretical premise was established that social class had a defined relationship to recognising and specifying outcomes of software development initiatives. Goodfriend (2017) suggested that "theories are an essential part of the framework used to organise specific social phenomena within the social sciences" (Goodfriend, 2017). Figure 4 (*Theoretical view*) below illustrates the theory and explains each entity and proposition associated with the theory.

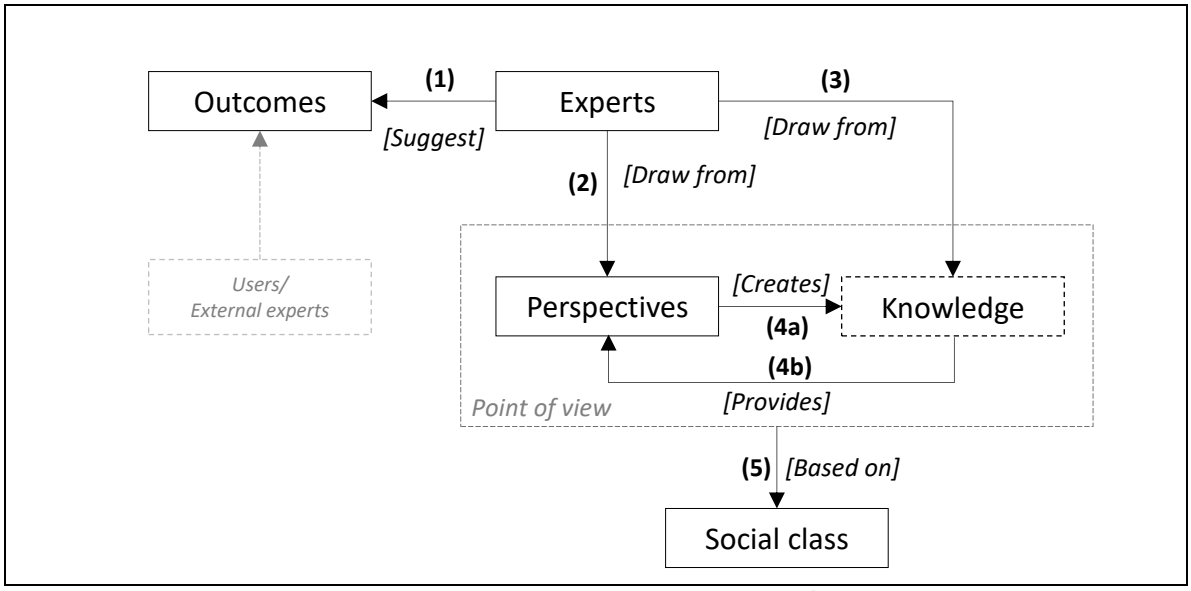


Figure 4: Theoretical view[6]

[6] Author's own work

Concepts

The incorporation of different concepts, both abstract and concrete provides different filters with which to view the phenomenon. Concepts are either physical entities, such as a person, or abstract, such as social class or point of view. However, abstract concepts are complex and in terms of this study are approached with a specific focus on the problem, rather than a generalist view. These concepts are defined as follows:

Social class

Social class is an abstract concept that is defined by the Oxford dictionary as "a division of a society based on social and economic status" (Oxford University Press, 2017b). Society can be classified into a hierarchy of social classes (Moe, 2017), based on education, role or designation, experience level, income, wealth, attainment or membership in a subculture or social network. It can be argued that each contributor categorised under social class contributes toward the construction of reality and as such could be incorporated as independent variables into the study. However, age, role or designation and experience level are favoured over other contributors due to their narrow focus and their potential for comparison and are thus viewed as the primary filter for the study.

Point of view

Point of view is an abstract concept and is highlighted to demonstrate the inferred relationship between perspectives and knowledge. Knowledge is embedded in the understanding of perception but is explicitly included to illustrate its direct relationship to experts as a physical concept. In terms of the theory, point of view is defined as the cumulative understanding of perspectives.

Perspectives

Perspective is an abstract concept and is described by the Oxford dictionary as "a particular attitude towards or way of regarding something; a point of view" and "true understanding of the relative importance of things; a sense of proportion" (Oxford

University Press, 2017b). Perspectives describe an attitude towards a phenomenon or point of view substantiated by knowledge, observations, personal experience, emotions and worldviews or word-of-mouth. Perspectives are personal and represent a reality to the individual framing them (Crotty, 1998). Crotty (1998) suggested that "each one's way of making sense of the world is as valid and worthy of respect as any other" (Ibid.).

Knowledge

Knowledge is an abstract concept and is defined by the Oxford dictionary as "facts, information, and skills acquired through experience or education; the theoretical or practical understanding of a subject" or "the sum of what is known". A philosophical reasoning suggests that knowledge is "true, justified belief; certain understanding, as opposed to opinion" (Oxford University Press, 2017b). As such, knowledge is viewed as self-established facts created by subjectivity or data.

Experts

The Oxford dictionary defines an expert as "a person who is very knowledgeable about or skilful in a particular area" and "having or involving a great deal of knowledge or skill in a particular area" (Oxford University Press, 2017b). The concept of "Experts" as defined in Figure 4 (*Theoretical view*) above is a physical concept and in terms of the theory is considered a professional capable of providing authoritative reasoning.

Outcomes

An outcome is an abstract concept and is defined by the Oxford dictionary as "the way a thing turns out; a consequence" (Oxford University Press, 2017b). This definition suggests that the way things turn out is a consequence but considered consequence as "a result or effect, typically one that is unwelcome or unpleasant" (ibid.). The basis of the definition is founded on causality and as such, outcomes are viewed as cause and effect and their relationship to the completion of a specific task. In terms of the research, a specific view is favoured that defines outcomes as the result of the software development initiative being successful or unsuccessful and the reasons associated with such a result.

Secondary Propositions

The illustration provided in Figure 4 (*Theoretical view*) provides a graphical representation of concepts relevant to the theory. Each concept has a potential relationship with other concepts, either directly or indirectly using a direct association with another idea. Relationships are required to provide a certain amount of truth, as the combination of all truths creates the theory. Each direct correlation is provided in proposition form to guide the theory.

Secondary proposition 1

Outcomes, in terms of the reasons associated with the result of a software development initiative being successful or unsuccessful, are specified by professionals capable of providing authoritative reasoning. As such, a direct and reasonable relationship can be established between outcomes and experts.

Alternate reasoning: It should be noted that reasons associated with the result of software development outcomes could also be provided by other entities not included in the theory. Physical concepts such as the end-users of the system who are responsible for providing the requirements of the system could provide insight regarding the functional correctness of a system as well as non-functional requirements such as ease of use. In addition, external experts, such as consultants can provide insight into the health of the system often related to the technical implementation and non-functional requirements, such as performance and scalability of a solution. These concepts are not included in the theory as they are explicitly excluded from the scope of the investigation. The focus of the theory is set on experts internal to the development team.

Secondary proposition 2

Professionals capable of providing sound reasoning cement their point of view in a "particular" manner towards what is being reasoned about. Experts draw from their perspectives, and as such, a direct and reasonable relationship can be established between experts and perspectives.

Secondary proposition 3

Similarly, professionals capable of providing sound reasoning rely on their theoretical or practical understanding, justified belief or certain understanding to form a point of view. Experts draw on their knowledge, and as such, a direct and reasonable relationship can be established between experts and knowledge.

Secondary proposition 4

Both perspectives and knowledge form part of a point of view. Knowledge is considered an attribute of perspectives, resulting in a default relationship between these concepts. It is, however, important to explicitly state the proposition in terms of a direct relationship between knowledge and perspectives in forming this theory.

A "particular" attitude towards something provides practical understanding, justified belief or certain understanding and, practical understanding and justified beliefs provide a point of view of something else. As such, a bidirectional and reasonable relationship can be established between perceptions and knowledge.

Secondary proposition 5

Education, role or designation, experience level, income, wealth, attainment or membership as well as a magnitude of other circumstances sculpt society. These conditions all contribute toward the construction of reality and provide similar and different perspectives used by individuals within a society. Perspectives are based on the social classes that 'construct' them and as such, a direct and reasonable relationship can be established between social class and point of view.

Primary proposition

Should all relationships established in secondary propositions one to five be valid, a plausible and indirect relationship can be established between social class and outcomes. As such, it is argued that professionals capable of providing authoritative reasoning, propose reasons associated with the result of software development initiatives being

successful or unsuccessful and these reasons are drawn from their "particular" attitude towards the situation as well as their practical understanding and justified beliefs. Their point of view is however based on their construction of reality rooted in their social class.

As such, it is argued that experts in diverse social classes may or may not provide different interpretations of factors that influence the outcomes of software development initiatives. This proposition served as the basis for the research question and hypothesis.

3.2.3 Variables

A variable is defined as anything that has a quantity or quality that varies. A dependent variable is a variable that the researcher is most interested in. An independent variable is a variable believed to affect the dependent variable (Moe, 2017).

Dependent variable

The research theory suggests that experts may or may not provide different interpretations of factors that influence the outcome of software development initiatives. The effects, in terms of failure or success, remain largely static and it is argued that its definition in terms of scope, time and budget are well understood by experts in the field of software engineering. However, causes or factors are subjective, varied and numerous (Karch, 2011), but using factor classifications, such as "internal variability" and "political impacts" to categorise factors, collected from existing literature, would eliminate variability. As such, of factors that influence software development outcomes are considered an ideal dependent variable to use in the research study.

Independent variables

The research theory suggests that expert opinion is founded in social class. It is however noted that only certain aspects that contribute towards social class can be considered in the study, due to the study's narrow focus and generalization potential. These include age, role or designation and experience level. These contributors are wide-ranging, and their inclusion in the research study is considered valuable in explaining the diversity of

experts. Thus; age, role or designation and experience level are found to be potential independent variables.

However, even though the narrow focus of these variables make them potential candidates, their fine-grained nature can introduce control complexities when comparability is considered. As such, each potential independent variable is refined to simplify comparability and increase control. This results in the ideal independent variables of generation (age), role (role or designation) and tenure (experience level).

Role (IV1): Primary

Role or designation as an independent variable is a simple concept, but its inclusion in this study is not without its challenges, such as the combined understanding of the differences between roles and designations. Yilmaz et al. (2012) suggested that designations were "job roles" individuals held in terms of the specific processes a development team followed, but "a role, on the other hand, is a series of expectations from an individual based on the team-based activities that are defined in a social context or a situation" (Yilmaz, O'Connor and Clarke, 2012). The authors uncovered nine different "job roles" used in traditional software development, thirteen "job roles" used in ISO/IEC 12207, seven "job roles" used in Extreme Programming (XP), six "job roles" used in Scrum and fifteen "job roles" used in Feature-Driven Development (FDD). The authors concluded that "we confirmed that several roles presented in older methods are *[sic]* emerged with a different name, with similar responsibilities in newer approaches" (Ibid.).

However, within the exhaustive lists provided by Yilmaz et al. (2012), different variations of homogenous responsibilities can be found. The software industry uses these interchangeably to promote positions, such as software developer, software programmer and software engineer to describe a position for a person responsible for producing source code. Equally, requirements engineer, technical writer and business analyst all refer to a person responsible for requirement elicitation.

It is evident that designation or "job role" has the potential to introduce complexities into the study as its granularity has the potential of reducing comparability. However, roles are found to relate more to the responsibilities of the individual and as such, when simplified, are considered an ideal independent variable for the study.

Generation (IV2): Secondary

In understanding experiences in the workplace between different ages and generations, Pitt-Catsouphes et al. (2009) suggested that people of similar ages could be divided into different generations. The authors stated that "key societal experiences (such as economic circumstances, historical events, and dominant cultural values) have the potential to affect the many ways that a majority of the members of these groups view the world and find meaning in their experiences (Pitt-Catsouphes, Matz-Costa and Besen, 2009). In their study, Pitt-Catsouphes et al. (2009) created six age groups and attached generation labels to each cluster as an independent variable to the research (*Table 3 Pitt-Catsouphes et al. generation make up*)

Table 3 Pitt-Catsouphes et al. (2009) generation make up

Generation	Traditionalists	Older boomers	Younger boomers	Older generation Xers	Younger generation Xers	Generation Y/ Millennials
Start date	-	1946	1955	1965	1972	1981
End date	1945	1954	1964	1971	1980	-

McCrindle (2010) suggested that characteristics associated with generation groups were not due to the influence of life stages but rather because people at various ages lived through the same events, creating different world-views at different ages (McCrindle, 2010). McCrindle (2010) found that markers for adulthood were delayed for Generation Y's when compared to other generations, and loyalty was a dominant characteristic for Boomers. The author noted that "while age influences behaviour and attitudes, greater

impacts are made by the culture in which one lives out one's youth, as well as social markers – significant events during one's formative years" (Ibid.). McCrindle (2010) research measured the Australian population using the generation make up highlighted in Table 4 (*McCrindle generation make up*) below.

Table 4 McCrindle (2010) generation make up

Generation	Federation generation	Builders	Boomers	Generation X	Generation Y	Generation Z	Generation Alpha
Start date	1901	1925	1946	1965	1980	1995	2010
End date	1924	1945	1964	1979	1994	2009	-

Pitt-Catsouphes et al. (2009) suggested that "chronological age" was often used as a measure of age-related human development and that it had become common practice to use generation mapping when discussing age groups, as visualising generations increased the understanding of a population (Pitt-Catsouphes, Matz-Costa and Besen, 2009). Existing research such as this revealed that different generations reason differently. As such, generational labelling is considered an ideal independent variable as ages can be separated into logical groups, allowing for control over the variability of this dimension.

Tenure (IV3): Secondary

Pitt-Catsouphes et al. (2009) suggested that "when considering age and work, it can be a bit difficult to untangle what is related to what because age is often connected to many aspects of our lives. For example, there is often a correlation between age, career-stage, life course, and tenure; however, it is important to keep the distinctions clear" (Pitt-Catsouphes, Matz-Costa and Besen, 2009).

For this study, tenure refers to experience in terms of the number of years an individual has contributed towards a specific role. Tenure relates to age and career stage, and as such, it can be argued that people with shorter tenure have less experience than those with longer tenure. Also, in the software industry, tenure is often used to explain seniority or

level of experience. As such, in this form, tenure is deemed an ideal independent variable for this study.

This section provides a complete reasoning in terms of selecting and validating the research methodology and approach as it relates to existing practices in research. The section covers four distinct areas such as the method, the methodology, the theoretical perspective and finally the epistemology. The contents of this section combine literature, theory and practical understanding as a background to the research methodology defined in the following section and as such is considered most valuable to academic readers.





3.3 Methodology selection & validation

The research problem suggests that in the absence of formal approaches aimed at collecting empirical evidence on the failure or success of software development projects, experts in the field of software development have no choice but to base their opinions on matter-of-judgement, rather than matter-of-fact when evaluating factors that influence software development outcomes. Its purpose, along with the primary research question is presented in such a way, to enable the collection of evidence capable of providing an understanding on the similarities between experts in different social classes and their perceptions of factors that influence software outcomes. On their own accord, the nature of the problem, the purpose of the research and the research question, to some degree, hint towards the chosen research methodology, however, justification is required in order to inform the study by answering the research question and by validating the importance of the research.

Crotty (1998) suggested that the importance of research outcomes could only be established if the research approach incorporated thinking in relation to the proposed method, governing methodology, theoretical stance and epistemology. The author further stated that each of these elements could inform one another, and describing the relationships between them could provide a mechanism for validating the approach (Crotty, 1998).

Each of these four elements are separately discussed in terms of the following questions:

- What **methods** are proposed to assist in collecting research data?

- Of the selected methods, which **methodologies** govern the proposed methods?
- What are the **theoretical perspectives** underpinned by the methodologies in question?
- What **epistemology** informs the theoretical perspectives?

Figure 5 (*Quantitative approach*) below illustrates the interrelationship of the methods, methodologies, theoretical perspectives and epistemology used in this research.

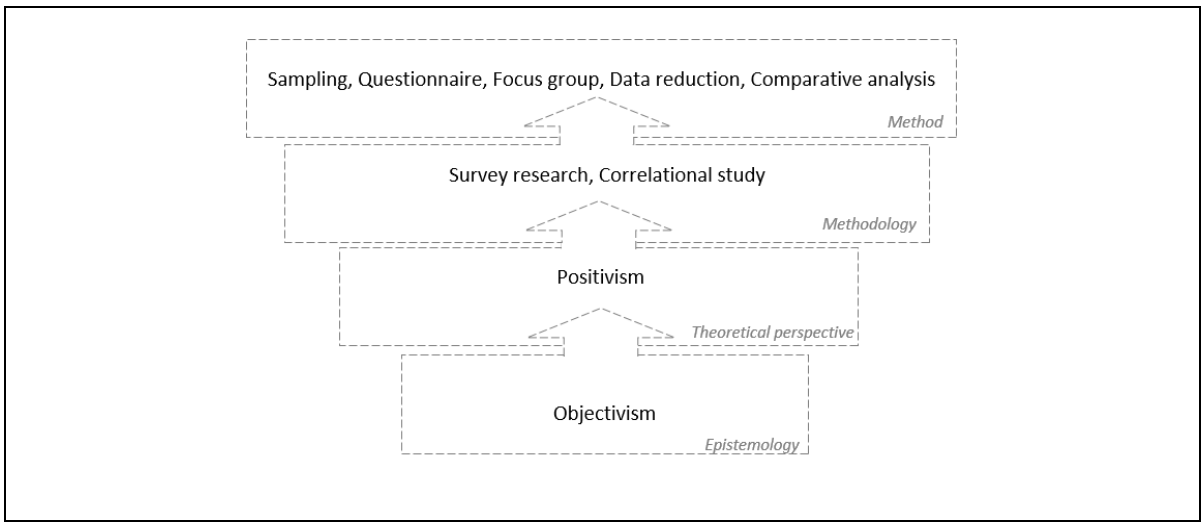


Figure 5 Quantitative approach[7]

3.3.1 Method

A method is defined as "a particular procedure for accomplishing or approaching something, especially a systematic or established one" (Oxford University Press, 2017b). It is suggested that methods describe the mechanisms, procedures or techniques used to gather data related to the research questions and hypothesis. An exhaustive list of methods has been used over the years, creating a wealth of knowledge for research, but the importance of incorporating techniques capable of augmenting the research is vital. The following methods were evaluated and proposed as appropriate for use in terms of the research problem, purpose, questions and hypothesis.

[7] Adapted from figure 1 in (Crotty, 1998)

Questionnaire

Bijker et al. (1987) suggested that the use of questionnaires rested "on the assumption that interesting and useful knowledge can be transmitted to members of society and the researcher alike in discrete bits capable of being written down, classified, and counted" (Bijker, Hughes and Pinch, 1987). Kothari (2004) noted that questionnaires were appropriate for the description, collection, analysis and interpretation of conditions that existed in a population. The author pointed out that the instrument was "only concerned with conditions or relationships that exist, opinions that are held, processes that are going on, effects that are evident or trends that are developing" (Kothari, 2004).

It is however noted that no research instrument is without its challenges and advantages and when selecting an appropriate instrument both advantages and disadvantages should be considered.

Challenges

When choosing the primary research instrument, it is important to understand the difficulties associated with that particular method. It is evident that these complexities may present risks during data collection and mitigation of these risks should be considered as part of the selection process. The following challenges are discussed in order to highlight the difficulties associated with selecting questionnaires as the primary research instrument.

The annoyance factor: Questionnaires have been used over the years as a research instrument to collect data from a populace. Their effectiveness in consistently collecting data allowed them to become a primary tool not just for academia, but also for industry. The over-use of questionnaires introduced data collection challenges due to a prospective respondent's irritation at having to deal with the unethical collection of data by corporations and scholars alike. Over time, the instrument grew to be associated with the stigma of "spam surveys". Poynter (2016) noted that individuals received too many questionnaire requests from far too many sources that were often of a general-purpose nature, were unsolicited or had no relevance to the person receiving it. The author wrote

that "customers have become so annoyed and alienated by spam surveys that many people have withdrawn some or all of their cooperation with companies" (Poynter, 2016). Gallant (2016) suggested that "many common survey practices alienate or at the very least annoy the people researchers are hoping to persuade to participate in their research projects" (Gallant, 2016).

As such, it is expected that difficulties in the collection of data using questionnaires would be encountered, and the instrumentation would have to incorporate mitigation strategies in terms of focusing on the relevance of the questionnaire to the proposed population, as well as the ethicality of its design and execution. In addition, it is expected that due to the challenges described above, the collection process would have to deal with scores of incomplete or partial submissions. Hurry (2014) stated that "it is not unusual to find incomplete questionnaires as a result of deliberate and accidental omissions" (Hurry, 2014).

Researcher bias: It can be argued that in social studies, the researcher is as much a contributor to the research as is the data collected to describe a phenomenon. This, therefore, introduces the risk of researcher bias into the study. Hurry (2014) suggested that "the structure of the questionnaire can too easily reflect the way the researcher sees the issues, both in terms of questions chosen and the way the questions (and possible answers) are phrased" (Hurry, 2014). In a study of research bias, Quelhas et al. (2011) found that biases such as "Ambiguous question", "Faking good", "Belief vs. behaviour", "Neutral opinion", "Faulty scale" and "Technical jargon" had the potential of leading to different answers in the research sample (Quelhas *et al.*, 2011). The authors suggested that "random errors are related to the variability inherent to a phenomenon that is being measured" and they directly affect the questionnaires' precision and replicability (Ibid.).

Additional data: Closed response formats have the ability to constrain the inferred meaning of research questions (de Leeuw, Hox and Dillman, 2008). This technique is primary to survey research and quantitative designs and allows the researcher to collect research data consistently. However, the nature of questionnaires designed to collect closed-ended answers limits the collection of additional data potentially required to further

explain or understand the phenomenon under investigation. The researcher has only one chance to collect the relevant information from the research sample and as such would be unable to probe for additional details once the questionnaire is completed. As such, should additional data be required, the researcher could consider employing different data collection techniques, such as focus groups to limit the chance of alienating the research sample by introducing additional questionnaires.

Advantages

There are, however, several advantages that influence the selection of questionnaires as a primary research instrument.

Reach: Questionnaires have the ability to reach scores of people, especially if internet and online surveying tools are considered as a distribution vehicle. The research population is spread across South Africa and collecting data from this population could be time-consuming and unreliable if other data collection approaches, such as the postal services are considered. The internet provides a platform whereby the questionnaire can be distributed across the country, with limited effort. Online surveying tools would make the distribution of questionnaires and the collection of responses simple and would decrease the time taken to collect data from the population.

Analysis: When structured correctly, questionnaires are easy to analyse. However, this would only be possible if the questions posed in the questionnaire are structured to incorporate simple closed-ended questions. It is suggested that a correctly structured questionnaire would have the potential to provide the answers required for analysis in a consistent manner.

Evaluator bias: Questionnaires provide an ability to administer data collection consistently. Consistency is key when evaluator bias is considered, and as all questions are posed in the same way, evaluator bias could be limited.

Simplicity: Over the years, questionnaires have become a common instrument used in collecting information from a population for both scientific and commercial studies. As

such, people are familiar with the concept and feel more comfortable responding to questionnaires than taking part in formal interviews. In addition, the closed-ended nature of a well-designed questionnaire would allow the collection of data from a population to be straightforward (de Leeuw, Hox and Dillman, 2008).

Focus group

Questionnaires are favoured as a data collection technique in quantitative analysis due to the consistency and repeatability advantages the instrument offers. However, it is noted that questionnaires are poor at collecting additional data to understand or explain a phenomenon once the data collection process is completed. Over the years, social research has adopted several different approaches to deal with the shortcomings of questionnaires, such as access to additional data. One such technique is known as discussion groups or focus groups; a practice imported from market research studies. Focus groups provide "in-depth information from a number of individuals simultaneously, making it a time efficient method of gathering data" (Macdonald and Headlam, 2008). Elliot and Timulak (2005) suggested that focus groups were considered a popular alternative form of qualitative interviews as it allows "access to a large number of possible views and a replication of naturalistic social influence and consensus processes" (Elliott and Timulak, 2005).

A certain amount of insight is required in validating the outcomes. As such, it is evident that focus groups could be an appropriate mechanism for evaluating the research findings, as a panel of experts from both academia and industry would be able to act as a sounding board, in order to reflect on what was discovered in the research. Also, this approach could be effective in helping to recognise any biases that may have been inadvertently introduced into the study.

Sampling

In terms of this research, sampling is considered a primary technique required to increase the generalisation aspect of the study, as it has a direct relationship to the software development population. In terms of the proposed primary research instrument, limited upfront control would be possible in designing the population sample, and as such, it is viewed that intervention in terms of sampling, would be required before and after the data collection process.

The sampling plan requires two distinct research samples; a defining sample consisting of software development experts working together as a software team, and the general population composed of experts in the industry. As such, quota sampling techniques are required to create a representation of the overall population. This technique could potentially inform the makeup of the general sample required for analysis. As such, stratified sampling is considered a valuable technique that could be used for data analysis and the combination of these approaches may increase the generalisation of the research outcomes across the local software development industry.

Comparative analysis

The process of data analysis involves scrutinising data samples to identify trends or patterns that may exist within the data. Macdonald and Headlam (2008) defined analysis as “a collection of methods used to process large amounts of data and report overall trends” (Macdonald and Headlam, 2008). The authors suggested that the primary function of analysis was to interrogate sets of data to find relationships between different sets (Ibid.). Wilson and Stern (2001) suggested that data analysis should be performed in three phases such as the exploratory data analysis phase, the deriving the main findings phase and the archiving phase. The authors considered the exploratory data analysis phase necessary, as analysis, even if the data collection stage was not completed, could give the researcher an upfront view of the value embedded in the collected data (Wilson and Stern, 2001).

Vosloo (2014) noted that "comparative analysis provides an alternative for statistical analysis, which bears closer relation to the positivist views" (Vosloo, 2014). This research aims at understanding whether there are differences in how factors that influence software outcomes are interpreted between experts in different social classes, and as such, comparative analysis techniques of summarising and comparing datasets that represented each variable or a combination of variables could be appropriate for proving or disproving the hypothesis and research theory.

Data reduction

Data reduction is a technique of reducing the amount of data a researcher must work with, either during the instrumentation phase or the data analysis phase. In terms of this research, data reduction is suggested as a primary technique in the instrumentation phase, specifically for refining factors that influence software development outcomes as a dependent variable. It is noted that the factors for success and failure of software development project outcomes, are too fine-grained and an aggregation would be better suited to illustrate the research phenomenon and to create the research instrument. As such, a primary step in the instrumentation phase requires factor analysis to aggregate failure and success factors, into existing factor classifications and then into the two most important research categories of "internal variability" and "political impact".

3.3.2 Methodology

In software engineering, the term methodology is understood as a well-defined process that incorporates practices aimed at solving common problems in a repeatable manner. This viewpoint remains valid when research methods are considered. Crotty (1998) defined methodologies as "the strategy, plan of action, process or design lying behind the choice and use of particular methods and linking the choice and use of methods to the desired outcomes" (Crotty, 1998). Quantitative research designs incorporate several methodologies, including survey research and correlational research to name a few. Quantitative methodologies aim at scientifically collecting data and as such are considered at least post-positivist approaches. However, not all methodologies can be regarded as

appropriate for all research problems, as the nature of the research may not benefit from the design of all methods.

In terms of this research, *correlational* and *survey methodologies* are considered appropriate as each of these are representative, either partially or wholly in terms of the selected methods described above.

Survey research

Survey research incorporates instruments such as interviews, focus groups and questionnaires in describing data associated with a population. Creswell (2014) suggested that survey research "provides a quantitative or numeric description of trends, attitudes, or opinions of a population by studying a sample of that population" (Creswell, 2014). Survey research describes trends and as such is not capable of providing causality of a research phenomenon (Keele, 2010). The main advantage that survey research brings to this study is the ability to demonstrate sample generalisation, due to the population reach that this approach provides (Creswell, 2014). In addition, the cornerstone of this investigation is to test whether different opinions are provided by experts founded in various social groups and this methodology is known for its ability to uncover opinions.

As such, a cross-sectional survey design is considered appropriate, as it can examine the perceptions provided by the research sample. The approach could then be sub-divided into different social classes for comparison, as a mechanism for measuring the differences in attitudes towards factors that influence software development outcomes.

Correlational study

Creswell (2014) defined correlational research as a mechanism to "describe and measure the degree of association (or relationship) between two or more variables or sets of scores" (Creswell, 2014). Keele (2010), however, noted that "the primary disadvantage with this design is that no conclusions can be made regarding causality, just that there is a relationship between the tested variables" (Keele, 2010).

The research theory suggests that social class has a defined relationship in recognising and specifying outcomes of software development initiatives. The primary hypothesis states that software development experts in different groups of a software development team, may or may not have differing views of “internal variability” and “political impacts”, as the factors that influence bespoke software development project outcomes in South Africa.

As such, a correlational design, using comparative analysis is considered an appropriate mechanism to compare the dependent variable in terms of the data collected for each independent variable or sets of variables.

3.3.3 Theoretical perspectives

Crotty (1998) suggested that a theoretical perspective is the “philosophical stance informing the methodology and thus providing a context for the process and grounding its logic and criteria” (Crotty, 1998). Social science theory is the art of condensing and shaping common knowledge and interconnecting ideas in order to interpret social phenomenon (Neuman, 2013). Harrington (2005) suggested that social theory “can be defined as the study of scientific ways of thinking about social life” and involves “ideas about how societies change and develop” (Harrington, 2005). These perspectives focus on the understanding of behaviour in terms of individual meaning and actions grounded in cultural interpretations. In considering the research problem, the study suggests that software development experts are diverse and as such, may have different views when outcomes are considered, due to the different social classes they belong to. The nature of this research is to re-evaluate expert views while focusing on various variables that define these experts.

However, this study aims at explaining the importance of empirical data collection rather than subjective reasoning when software development outcomes are measured, and as such underpin the universal principles of observable facts, suggesting a study in natural science rather than social science. As such, it is viewed that the evaluation of the theory and hypothesis should incorporate methodologies and practices that underpin a study in positivism.

Positivism

Positivism is a philosophy that stresses empiricism and highlights the importance of objectivity as well as the need to study observable components. This research supports the belief that through following accepted methodologies and the practices they embed, the study would be able to empirically prove or disprove a significant difference in opinions between experts of different social classes in terms of the factors that influence software development outcomes. The process would follow a controlled and structured approach, by specifying a clear and concise research topic, theory and hypothesis; and collecting and analysing research data while remaining detached from the research participants to increase objectivity.

However, when using a survey research methodology, specifically questionnaires, the research requires input from participants in the software industry. In addition, it is viewed that experts in the form of a focus group could provide additional benefits to the study as this approach is considered an appropriate mechanism for evaluating the research findings. It can be argued that participants selected for the research sample are disconnected from the study, and their objectivity could be assumed, although their disconnected nature limits the observability requirements set by the rules of positivism. Also, positivism historically frowned upon human observation as objects for research and argued that "no realistic account of social facts will assume that people always act rationally" (Götschl, 1995).

Various arguments have been made over the years to validate human contributions towards positivist studies. Macdonald and Headlam (2008) suggested that positivism was "a paradigm that assumes human behavior [*sic*] is determined by external stimuli and that it is possible to use the principles and methods traditionally employed by the natural scientist to observe and measure social phenomena" (Macdonald and Headlam, 2008). Vosloo (2014) agreed by arguing that positivism "entails a belief based on the assumption that patterns (trends), generalisations [*sic*], methods, procedures, cause-and-effect issues are also applicable to the social sciences" (Vosloo, 2014). The author further suggested that people, in terms of the objects of social science, were suitable for the implementation of social methods (Ibid.). In addition, Hurry (2014) stated that "questionnaires are

typically seen as reflecting a 'positivistic' research orientation, seeking to capture an 'objective', 'real' world 'out there'" (Hurry, 2014).

It is argued that in terms of this study, empirical data collection and analysis is important for demonstrating differences in expert opinion based on social class. The phenomenon, by its very nature, may introduce challenges related to an interpretivist opinion, but the focus on objectivity as a positivist design premise allows for a disconnect between the social world and the research being done.

As such, the following positivist perspective is adopted to guide the research:

> *Data is everywhere and easily accessible; either through existing literature or by sampling the relevant population. This research, by using trusted methodologies such as survey and correlational research approaches, would be able to leverage techniques such as questionnaires to collect relevant information that can empirically and factually prove or disprove that there is a significant difference in how experts from different social classes view the factors that influence software development outcomes.*

3.3.4 Epistemology

Crotty (1998) suggested that an "epistemology is concerned with providing a philosophical grounding for deciding what kinds of knowledge are possible and how we can ensure that they are both adequate and legitimate" (Crotty, 1998). The author noted that objectivist epistemology provided the meaningful reality of 'what it is to know' and suggested that knowledge was considered "objectified in the people we are studying". By incorporating a positivist opinion that leverages appropriate methods, an objective truth could be discovered (Ibid.).

Even though this research is ultimately founded in objectivism, the importance of the constructionist undertone cannot be underestimated, as it offers an opinion that defines the problem in terms of the context of the study. As such, constructionism is discussed in

terms of the way the world exists and objectivism in terms of the way the world should be (Ibid.).

Constructionism

Reality is constantly changing, and in terms of this research, it is argued that the evolution of software development methodologies was due to shifts in the realities created and re-created by the inventors of these methods. Their existence and the differences in each method provides subjective proof that there are multiple truths in play when these practices are considered. Sale et al. (2002) suggested that "on an epistemological level, there is no access to reality independent of our minds, no external referent by which to compare claims of truth" (Sale, Lohfeld and Brazil, 2002). Furthermore, the research proposes that experts base their findings of factors that influence software outcomes on their perceptions of truth which may be socially constructed. Software development as an abstract entity exists across several realms, such as engineering, science and business. Each of these influences the entity by imposing 'particular' constraints upon the practice. As such, software is considered to be "where the technology of computing meets social relationships, organizational [*sic*] politics, and personal agendas" (Ensmenger, 2010). Ensmenger (2010) wrote that "all technologies are to a certain extent social constructions, but in the case of software, the social dimensions of technology are particularly apparent" (Ibid.).

Bruner et al. (2008) suggested that "learning is a constructive process in which the learner is building an internal illustration of knowledge, a personal interpretation of experience" (Bruner, Vygotsky and Feuerstein, 2008). The authors further noted that academic growth originated from sharing perspectives as well as the shifting of internal representations, as representations were continually open to modification. Creswell (2014) stated that "individuals develop subjective meanings of their experiences, meanings directed toward certain objects or things." (Creswell, 2014) Creswell further suggested that subjective meaning was socially negotiated.

Research into theories associated with the social construction of reality is well documented in academia. The epistemology suggests that the development of a jointly constructed understanding of the world forms the basis for shared assumptions about reality. Crotty (1998) suggested that "social constructionism emphasises [sic] the hold our culture has on us: it shapes the way in which we see things (even the way in which we feel things!) and gives us a quite definite view of the world" (Crotty, 1998). The validity of social constructionism is not in question, and this study provides valuable insight into understanding the state of software engineering. As such, it is viewed that its arguments are applicable for use in this research.

Objectivism

"An epistemology, we have already seen, is a way of understanding and explaining how we know what we know" (Crotty, 1998). Epistemology describes the establishment of valid knowledge and how it can be obtained. It is believed to provide a theory of knowledge and a view of reality, underpinning a theoretical perspective and methodology (Ibid.).

Experts are defined as "having or involving a great deal of knowledge or skill in a particular area" (Oxford University Press, 2017b). In terms of this research, it is suggested that individuals in software engineering are often illegitimately 'promoted' to the status of expert out of necessity, rather than the actual grounding expert status normally requires. This necessity is due to the limited availability of skilled software professionals, or 'the software crisis' as it is commonly referred to. Ensmenger (2010) suggested that the software crisis has persisted over the years and wrote that "the Y2K crisis, H1-B visa debates, and recent concerns about the loss of programming jobs to India and Pakistan are only the most recent manifestations of the industry's apparent predilection for apocalyptic rhetoric" (Ensmenger, 2010).

This research, through a positivist approach suggests that, by illustrating that experts from different social classes view factors that influence software development outcomes differently, it could prove that the reality in relation to the evolution of software

development practice is independent of the consciousness the impact of expert judgement has on software development outcomes. The viewpoint, by its very nature, fortifies objectivism as the epistemology of the study.

This section provides a summary of the theoretical foundation by restating some of the important aspects highlighted in the text such as the research theory and method selection and validation, handling concepts such as the method, methodology, theoretical perspective and epistemology, thus suggesting that it is appropriate for all reader types.			

3.4 Summary

The study aims to understand whether there is a significant difference in how experts in different roles, tenure and generations view factors that influence software development outcomes. As such, a theory is created that suggests that experts in diverse social classes may or may not provide different interpretations of factors that influence software development initiatives.

In understanding how the research should be approached, different methods, methodologies and theoretical perspectives are provided. An epistemology that incorporates both constructionism and objectivism is provided, that suggests that social constructionism is appropriate, but only in understanding that in terms of this research, the evolution of software development methodologies may be found in it. The epistemology notes that the approach suggested for defining, collecting and interpreting the data gathered for the research, is better suited to an Objectivist perspective. Suitability is founded in a positivist approach that can illustrate that experts from different social classes view factors that influence software development outcomes differently, thus proving, that the reality in relation to the evolution of software development practice is independent of the consciousness of the impact it has on software development outcomes.

4 Research methodology

This section provides the primary research question as well as more granular view defined as three supporting research questions and as such is considered most valuable to academic readers.





4.1 Research questions

4.1.1 Primary question (RQ1)

This research argues that expert opinions and perspectives contribute to establishing the factors that influence software development outcomes, especially when formal approaches aimed at collecting empirical evidence are not present. This has the potential of underpinning software development practices and approaches in bias, as experts innovate in the realm of software development methodology. There is no research available aimed at investigating the impact that different expert perspectives have on the evolution of software development methodology, and as such, the research set out to understand how factors that influence software development outcomes in South Africa are recognised. Specifically, the study attempts to determine the relationship between different groups present in a software development teams, and their perspectives of factors in terms of "internal variability" and "political impacts", as "influences" on software development outcomes. As such, the following research question and hypothesis are proposed:

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts in different groups of a software development team?*

Null hypothesis 1 (H01): There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts in different groups of a software development team.

Alternate hypothesis 1 (H1): Software development experts in various groups of a software development team have differing views of "internal variability" and "political

impacts" as the factors that influence bespoke software development outcomes in South Africa.

4.1.2 Supporting questions

Secondary questions are required to provide input into answering the primary question. Secondary questions are posed to provide insight into the different groupings suggested by the primary research question and includes determining the relationship of the perspective of factors that influence software development outcomes and (RQ2) different roles, (RQ3) different generations and (RQ4) mixed tenure of the software development team. As such, the following sub-questions are presented.

Research Question 2 (RQ2)

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts in different roles?*

Null hypothesis 2 (H02): There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts in different roles.

Alternate hypothesis 2 (H2): Experts in various roles in a software development team have differing views of "internal variability" and "political impacts" as the factors that influence bespoke software development outcomes in South Africa.

Research Question 3 (RQ3)

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts from different generations?*

Null hypothesis 3 (H03): There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts from different generations.

Alternate hypothesis 3 (H3): Experts from various generations in a software development team have differing views of "internal variability" and "political impacts" as the factors that influence bespoke software development outcomes in South Africa.

Research Question 4 (RQ4)

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts of different tenure?*

Null hypothesis 4 (H04): There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts with different tenure.

Alternate hypothesis 4 (H4): Experts with different tenure in a software development team have differing views of "internal variability" and "political impacts" as the factors that influence bespoke software development project outcomes in South Africa.

This section provides a complete description of the design of the research investigation. Based in the pre-work done in the methodology selection and validation, the research design section incorporates the formation of factors that influence software outcomes, the research instrumentation, the data collection approach, the data analysis approach, the data interpretation approach and finally the method in which the research result is disseminated. As such, this section is considered most valuable to academic readers.			

4.2 Research Design

Section 3.3 (*Methodology selection & validation*) above aims at establishing which methodologies could be considered appropriate for this research, by understanding the theoretical perspective and epistemology suggested for this study. This exercise suggests that survey and correlational research methods are valuable in describing the research from a positivist perspective. In a study to demystify the selection of research approaches, Keele (2010) proposed a model to assist in selecting the correct type of quantitative research design. This model is used to validate the viewpoints provided in Section 3.3 (*Methodology selection & validation*), and Figure 6 (*Research design decision tree*) below illustrates Keele's model (Keele, 2010) employed in the evaluation.

Correlational design to quantitative research is considered appropriate as the study relies on describing the significance of relationships between perceptions on software development cause and effect and different groups within the software development team, such as role, generation and tenure. The software development industry in South Africa enjoys large numbers, and as such, it is expected that the research will have to deal with a substantial amount of data.

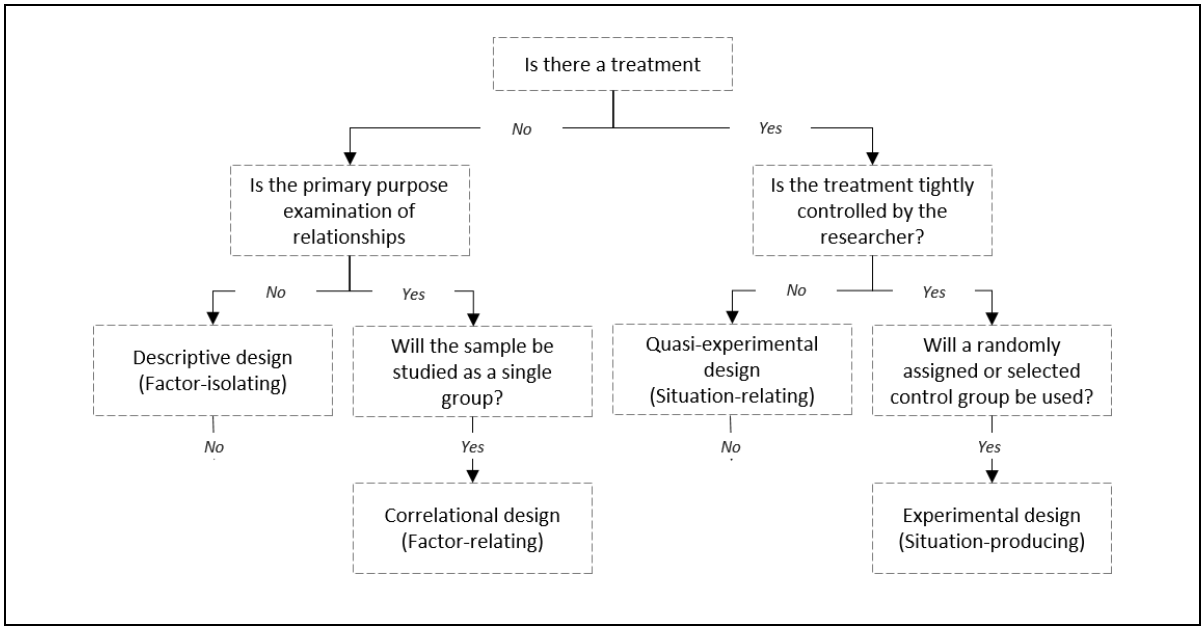


Figure 6 Research design decision tree[8]

Wellman (2005) suggested that correlational research was well suited to dealing with relationships and significant data but would be unable to describe why these relationships exist. The author stated that correlational research "attempts to discover or establish the existence of a relationship/ interdependence between two or more aspects of a situation" (Welman *et al.*, 2005). The research problem suggests that, in the absence of formal approaches, expert's opinions and perspectives are recorded as evidence into software development outcomes, and the research questions are designed to investigate whether there is a difference in opinions between different groups of the software development team. As such, there is no need to establish why these relationships exist, only that they do exist.

[8] Adopted from Quantitative Designs (Keele, 2010)

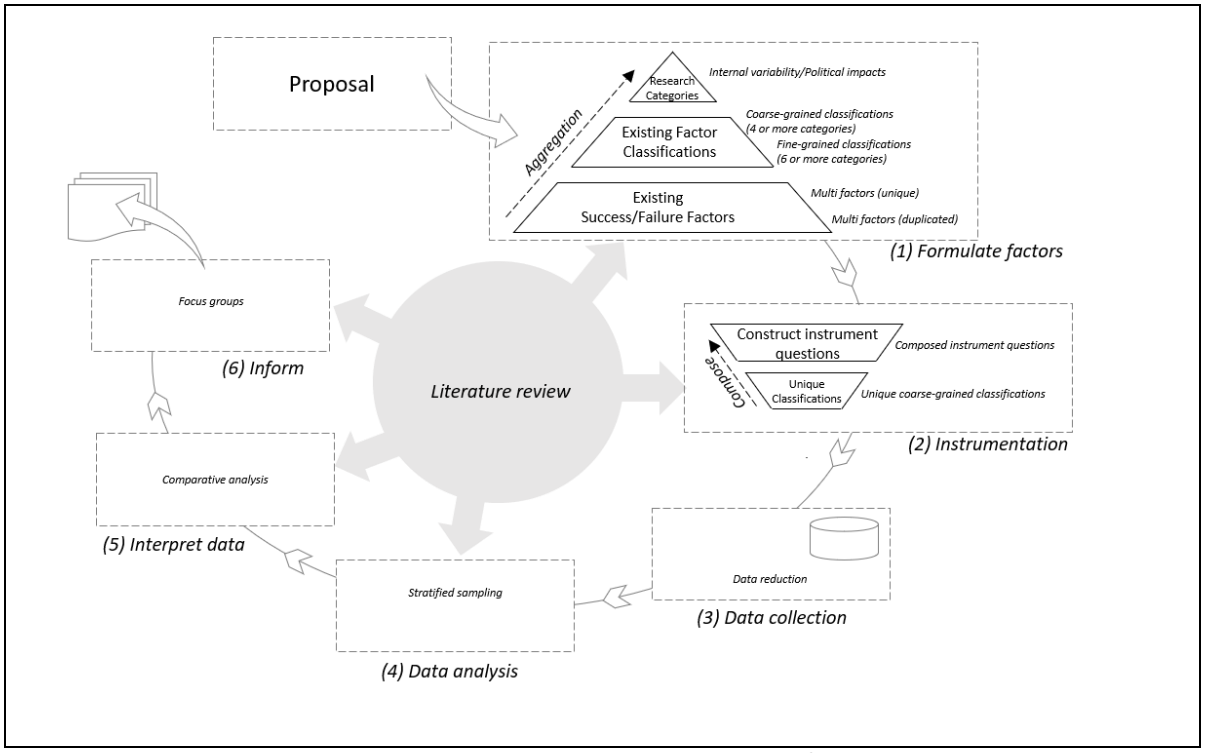


Figure 7 Research process[9]

The correlational design proposed for this research has to be extended to incorporate all the methods defined for the study. As such, the design is approached from a perspective that allows formulating factors that influence software outcomes, creating the instrument, collecting and analysing the data, as well as interpreting and informing the findings to the greater software development audience. This design is presented in Figure 7 (*Research process*) above. Each step is described separately in terms of this representation.

4.2.1 Formulate factors that influence software development outcomes

While defining the research theory, it was uncovered that an extensive effort would be required to formulate the dependent variable, factors that influence software development outcomes. It is argued that the level of granularity, for factors responsible for software development successes and failures, are too fine-grained and factor categorisations would provide an appropriate coarseness to describe the variable, especially when categories

[9] Author's own work

such as “internal variability” and “political impacts” are considered. As such, factors require aggregation into factor categories to achieve an adequate level of granularity. The process of formulating factors that influence software development outcomes consists of collecting factors and categories from the existing body of knowledge, then aggregating these into categories that ultimately represented the two primary factor classifications of “internal variability” and “political impacts”. This process is intended to move from fine-grained factors to coarse-grained categories.

Factor and categorisation collection

It can be argued that collecting factors for the failure and success of software development initiatives, is less valuable if they are to be gathered from the software development population, than testing well-understood and pre-defined factor aggregations against a population, as the former has the potential to produce similar concepts based on different interpretations. As such, data in terms of factors and categorisations collected from existing literature is considered more appropriate and could simply be aggregated into “internal variability” and “political impacts”.

The design extends the literature review to incorporate the collection of the factors for software development successes and failure from both formal and informal literature sources. Moreover, existing factor classifications and categorisation must be evaluated and documented as part of the literature review to form the basis for aggregating these factors into the two primary research categories.

Coding

It is clear that the process of aggregation deals with data from a qualitative perspective. As such, coding techniques, commonly used in qualitative analysis, are introduced to assist in the process. Saldaña (2010) suggested that code “is more often a word or short phrase that symbolically assigns a summative, salient, essence-capturing, and/or evocative attribute for a portion of language-based of visual data” (Saldaña, 2010). Lockyer (2004) suggested that coding was “a systematic way in which to condense extensive datasets into smaller

analyzable units through the creation of categories and concepts derived from the data" (Lockyer, 2004). Bourque (2004) added that coding was "the process by which verbal data are converted into variables and categories of variables using numbers, so that the data can be entered into computers for analysis" (Sage, 2004).

Saldaña (2010) provided an extensive list of coding practices used in both quantitative and qualitative research. For data aggregation, Descriptive Coding techniques are regarded as an appropriate approach for the first cycle or initial coding. Saldaña (2010) suggested that Descriptive Coding or Topic Coding was a technique that "assigns basic labels to data to provide an inventory of their topics" (Saldaña, 2010). Saldaña (2010) further explained that Descriptive Codes could be assigned a more detailed sub-code, such as "Parent" and "Child" to assist in forming a hierarchy of "siblings" and suggested that this approach led to a categorised inventory of the data's content (Ibid.). In addition to Descriptive Coding, a Focus Coding technique is considered appropriate for the second cycle approach in creating the most salient categories required for the research. Saldaña (2010) suggested that Focused Coding was suitable for studies that required the development of the main categories or themes from the data as this technique was capable of searching for the most frequent or significant initial codes (Ibid.).

Factor aggregation

To formulate factors that influence software development outcomes as a dependent variable, factors and factor categories collected as part of the literature review need to be aggregated into the two primary research categories of "internal variability" and "political impacts". Before aggregation can be attempted, a reduction of factor data is required to eliminate duplication within the initial dataset. Data reduction to remove duplication is performed by defining a set of labels for each factor and assigning a weight to each tag. Summing the values of labels provides an instance count of each, meaning that labels with higher values will have a greater degree of duplication.

Once data reduction is completed, the aggregation of factors is started. Data reduction is done by using factor mappings where labels are added to each categorisation. With labels

assigned to both factors and categorisations, factors are then grouped into relevant categories. In addition to reducing data in terms of factor to category, additional aggregation is required in which categories are aggregated into coarser-grained categories. This is done by allocating abstract labels to each set, then grouping the finer-grained set in terms of the coarse-grained set. Once completed, the coarse-grained categories are assigned to a research category of either "internal variability" and "political impacts". Figure 8 (*Factor aggregation*) illustrated below, provides a visual representation of this process.

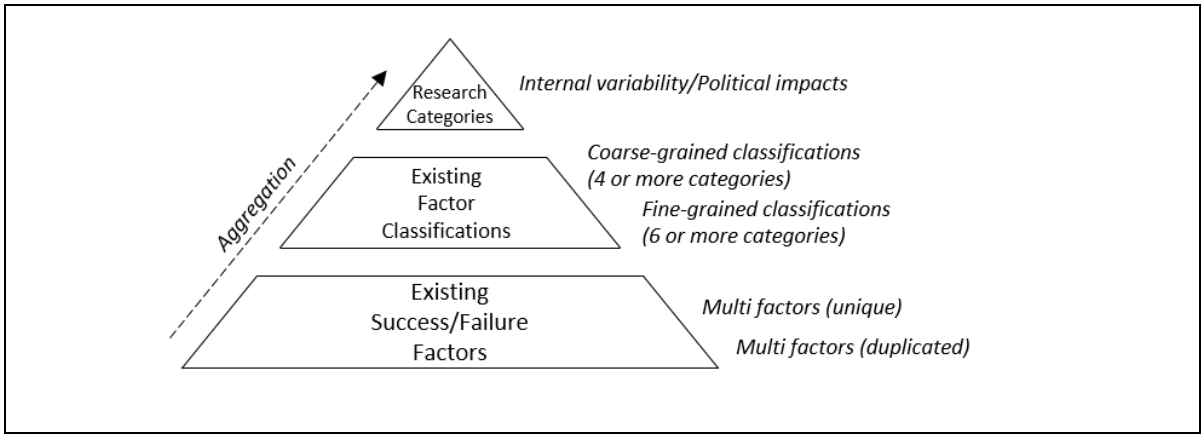


Figure 8 Factor aggregation[10]

Factor aggregation result

Using Google Scholar, a simple internet content search using a search term such as "software failure" produces a fraction more than twenty-three thousand results. The search result alone provides evidence underpinning an opinion that many individuals have attempted to unpack the reasons why software development projects continue to fail.

Why do software development initiatives fail? During research, this study uncovered more than 209 distinct reasons for software failure from formal empirical studies done over the years. It is noted that this is not an exhaustive list and due to the nature of human

[10] Author's own work

perception, no study would be able to successfully compile a definitive list containing an all-encompassing view of software failure contributors.

In defining the primary software outcome categories, "political impact" is ultimately underpinned by top management and organisational factors, especially when focus is considered. As such, this category includes reasons such as commercial pressures, lack of commitment, communication, control, maturity, ownership and unrealistic cost/value propositions to name a few.

In terms of "internal variability", project management, system complexity, skills, technology and process factors underpinned this category. These themes are described as problems relating to communication, control, focus and maturity, planning, risk mitigation, quality, skills, user involvement, system size, commitment and inappropriate use of methodology to name a few.

This study does not question the validity of any reason in terms of the impacts they may have had on the outcomes of software development initiatives but argues that even though there appears to have been value in uncovering them, no guarantee can be found that these reasons are not founded in contributor bias. As such, their value in terms of this research is not to explain why software development projects fail or succeed, but in re-using them as a perceived reality for software outcomes and testing whether experts reason differently about them.

The process of factor aggregation revealed a total of 209 factors. Annexure B (*Literature review summary on reasons for software project failure*) provides an account of the outcomes of the factor aggregation process.

4.2.2 Instrumentation

Neuman (2013) suggested that "designing a study requires making many decisions about the type of case or sample to select, how to measure relevant factors, and what research technique (e.g., questionnaire, experiment) to employ" (Neuman, 2013). An appropriate instrument is required to facilitate data collection. Section 3.3 (*Methodology selection &*

validation) suggests that survey research using questionnaires are an appropriate methodology and instrument to collect data from a population, as a correlational approach will provide the collection of research data in a consistent manner. However, consistency could only be achieved if the content of the instrument is structured in such a way as to promote it. Also, it is regarded paramount that the foundation of the questionnaire is based on the collection of data in relation to the dependent variable as well as the independent variables defined as role, tenure and generation. The process of formulating factors that influence software development outcomes provides the basis for the types of data required by the study, and the instrument is regarded as a vehicle appropriate for collecting this data. The following sub-process is suggested in developing an appropriate instrument for the study.

Objectives

Moe (2017) suggested that instrument objectives "are part of the research study, anchoring what you hope to find or achieve with the methodology" (Moe, 2017). The author further noted that the importance of documenting the efforts employed to achieve a solution in the face of the research problem has the potential to bind the instrumentation and focuses on the data collection in a manner that allows the research to incorporate only that which was needed (Ibid.). The purpose of the research is to understand whether experts in different roles, generations and tenure reason differently when considering factors that influence software development outcomes. As such, it stands to reason that the instrument must achieve the following objectives:

- *Use an instrument that can mature as more data is collected, should it be required.*
  The selected instrument must have the ability to mature over time, should there be a need. Consistency is considered a primary requirement, but it is viewed that consistency would only be achieved if the instrument is properly reviewed and tested. This objective makes provision for testing and morphing the instrument from initial design to the final version that is used to collect data from the research sample.
- *Use an instrument that facilitates consistent data collection.*

As consistency is considered a primary requirement, the design of the instrument has to incorporate consistency in terms of the way that questions are presented as well as the manner in which data is collected from the research sample. Consistency is considered key to replication, without which the data could be questioned.

- *Use an instrument that facilitates ease of data collection.*
  Ease of data collection is crucial as it directly relates to the management of data across the study. In addition, the potential of introducing data anomalies is considered a risk, should the data collection process be complex.
- *Use an instrument that is readily available to the research sample.*
  It is noted that securing and using an appropriate research sample would be difficult. Also, over time different instrument types have alienated people with arguably limited numbers willing to assist in research studies. As such, the instrument would have to make it easy for a potential respondent to provide the information required.
- *Use an instrument that can capture as much of the relevant population as possible.*
  The research is bounded within the borders of South Africa but accessing the software engineering population is challenging. In addition, several decisions are made that have the potential of reducing the response sample. As such, the instrument must facilitate capturing the audience in such a way that the bounded context of South Africa is represented while limiting the risk introduced by limiting data due to sampling techniques.
- *Pose questions that are relevant to the research sample.*
  It is noted that research respondents are lost when they are asked to participate in a study they feel has no relevance to themselves. As such, the instrument must explain what it aims at achieving and why the participants are chosen.
- *Pose questions that the research sample can associate with.*
  It is noted that the selection of an appropriate research sample is paramount. The independent variables provide a quasi-measure that assist in formulating and growing the population. However, the instrument must ensure that the questions that are posed, are designed to allow respondents to associate themselves with it.
- *Pose questions in a manner that is easy to understand by a research sample.*

It is viewed that the inability of a potential respondent to understand the aim of a question, would either alienate the respondent or, result in the respondent abandoning the data collection exercise. An addition risk includes the potential that respondents may provide answers that do not reflect their actual opinions. As such, the research questions have to be created in a manner that a respondent would find easy to relate to and understand. In terms of question design, simplicity is considered key to collecting relevant information.

- *Pose questions in a manner that is easy to answer by the research sample.*

  It is viewed that, in addition to easing the understanding, facilitating the answering of a question by providing a mechanism that is simple to use, would allow respondents to partake in the data collection exercise without frustration. As such, the answering mechanism should be simple, consistent and appropriate.

- *Provide an answering mechanism that facilitates data aggregation and analysis, without the need for further analysis.*

  In addition to reducing the respondents' frustration while answering questions, the answering mechanism should be created in such a way, so as to facilitate data aggregation and statistical analysis. As such, the questions are required to be closed-ended, to reduce the need for an additional qualitative analysis exercise after the data is collected.

- *Pose questions that encapsulate rational reasons for failure and success in software development projects.*

  When considering relevance, and incorporating the independent variables of role, tenure and generation, the dependent variable of factors that influence software development outcomes must be the focus of the data being collected. Questions must be created in such a way, so as to capture the meaning of influencing factors by leveraging the outcomes of the aggregation process used to form it.

- *Pose questions that can group respondents into different groups defined by role, tenure and generation.*

  The instrument is required to capture data about the population. This metadata is required to group respondents during the analysis phase, and as such, questions related

to the independent variables of role, tenure and generation have to be included in the instrument.

- *Pose questions that incorporate cause and effect in terms of both favourable and unfavourable outcomes.*

  The research attempts to investigate the phenomenon of factors that influence software development outcomes. Cause and effect in terms of this research are viewed as both favourable and unfavourable outcomes, and as such, the instrument should facilitate collecting data in both realms of successful and failed software development projects.

When considering any form of research, ethics are a primary concern. A research instrument should achieve its objective in a manner that does not jeopardize morality (Moe, 2017). As such, the instrument design and data collection mechanism underpin morality as a supreme principle. Annexure F (*Ethical Considerations*) provides a description of an ethical standard created for this research and its incorporation into this study provides an irrefutable focus on professional practice.

Instrument selection

Creswell (2014) suggested that a questionnaire "provides a quantitative or numeric description of trends, attitudes, or opinions of a population by studying a sample of that population" (Creswell, 2014). Section 3.3.1 (*Method*) proposes that survey research using questionnaires is an appropriate instrument for collecting data from a relevant population. The proposition of this instrument is based on the advantages the instrument offers, such as consistency, reach and simplicity. It is argued that questionnaires can consistently collect data from a population distributed across a wide geography. Coherence and reach are considered important aspects as both contribute towards proving generalisation of the study's outcomes. In addition, simplicity in terms of creating, distributing, collecting and analysing data is considered a key factor in this research. It is, however, important to note that the instrument would be a cross-sectional questionnaire, rather than a longitudinal questionnaire. Neuman (20130) suggested that "Cross-sectional research gathers data at one time [*sic*] point and creates a kind of 'snapshot' of social life" (Neuman, 2013). The

author noted that a shortcoming of cross-sectional questionnaires was that they examined information for many unrelated cases at a point in time and as such was unable to capture social processes or change (Ibid.). For this research, there is little need to understand why the phenomenon exists, but purely whether it did or did not. Even though respondents are asked to focus on software development projects that failed or succeeded during their careers, it is suggested that the data is collected at the time the research is done and that the respondent's perspectives of the related outcomes are necessary at that very moment.

Instrument content development

Neuman (2013) suggested that instrument content development was a crucial step in conducting a research study, as this approach would narrow down the broad-spectrum, that the question and hypothesis proposed, into specific questions a research population could answer (Neuman, 2013). This step is designed to build the instrument in such a way, so as to facilitate the collection of relevant data from a population, in order to describe their perspectives of factors that influence software development outcomes. The sub-process (*Figure 9 Question composition*) is intended to compose instrument questions by elaborating on the coarse-grained categories uncovered in the factor aggregation process. An Axial Coding technique is employed to re-assemble categorisations uncovered in the aggregation process in order to create the required research questions. Saldaña (2010) suggested that the purpose of Axial Coding was to “strategically reassemble data that were ‘split’ or ‘fractured’ during the initial coding process” (Saldaña, 2010). The author noted that the practice of grouping coded data had the ability to reduce the number of the original codes while re-labelling them into conceptual categories, in this case, questions (Ibid.).

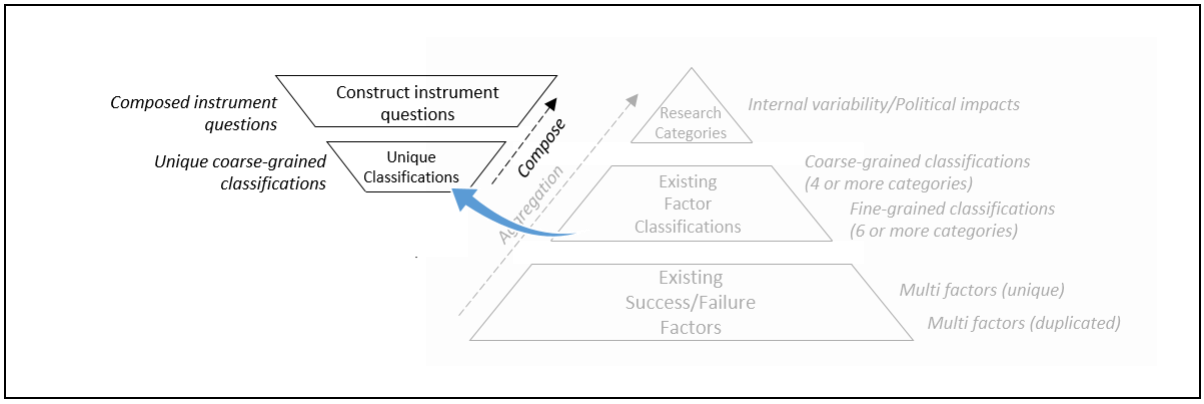


Figure 9 Question composition[11]

The questions posed in the questionnaire incorporate both internal variables (demographic information) as well as dependent variables (factor category questions) used to collect data from the population. The questionnaire is created with closed-ended questions using a Magnitude Coding technique to allow respondents to provide answers to the questions posed. The coding technique uses numbers 1 to 5 where 5 is considered most-important and 1, least-important.

Instrument layout

The content developed as part of instrumentation incorporates a total of four questions aimed at capturing a respondent's demographic information, as well as two sets of sixteen questions each aimed at capturing the respondent's perspectives of factors that influence software development outcomes. In addition to this, the questionnaire contains a participant information sheet (*Annexure C Participants information sheet*) consisting of an introduction to the study, the aims of the study, as well the researcher's ethical consideration of a respondent taking part in the study.

[11] Author's own work

Demographic information

As suggested by in the Independent variable section of the Theoretical premise (*Section 3.2.2*) described in this study, generation, role and tenure are appropriate in defining the social class of each respondent. As such, each of these are incorporated into the questionnaire in order to provide a demographic representation of those that will be surveyed.

In terms of generation, respondents are asked to select their age range. The options provided in the questionnaire include the ages defined in table 5 (*Age group to generation label mapping*) below.

Table 5 Age group to generation label mapping

Age group	18 - 22	23 - 30	31 - 40	41 - 50	51 – 65
Generation label	Generation Z	Early-Mid Generation Y	Late Generation Y - Early Generation X	Mid - Late Generation X	Late Generation X - Early Boomers

Respondents are also required to provide their tenure in terms of their professional experience in their current role. The possible answers used in the questionnaire include the options specified in Table 6 (*Tenure answer options*) below.

Table 6 Tenure answer options

Questionnaire answer	**Description**[12]
Less than 1 year	Learner level
At least 1 year but less than 2 years	Junior level
At least 2 years but less than 7 years	Junior to intermediate level
At least 7 years but less than 12 years	Intermediate to Senior level
12 years or more	Senior level

Finally, respondents are asked to select a role from a predefined list that best describes their role or position. As suggested in the Role (IV1) section of the Theoretical premise (*Section 3.2.2*) outlined in this study, the software industry uses role labels in different ways to describe the primary responsibility of an expert. Role labels include software developer, software programmer and software engineer to describe the position of a person responsible for producing source code or, requirements engineer, technical writer and business analyst to describe a person responsible for requirements elicitation. As such, the roles considered as answers for this study include only the basic and most commonly used roles. The available answers to the questionnaire include the options defined in Table 7 (*Role answer options*) below.

[12] The levels suggested is not academically founded, but is used in the software industry as a common grouping for software practitioners.

Table 7 Role answer options

Role	Description
Developer	Describes a person responsible for producing source code used in creating a software system. This role incorporates all aspects of source code, including databases, back-end system, application services and user interfaces.
Business Analyst	Describes a person responsible for providing a translation between a business entity and a 'developer' or development team, that describe the functional requirements of the software system, how it's used and the business value it would provide.
Project Manager	Describes a person responsible for guiding, managing or maintaining a plan, approach or perspective, that a combination of 'developers' and 'business analysts' may use in developing a software system. This role provides guidance and oversight to a development team and facilitates interaction between the team and the business.
Architect	Describes a person responsible for providing an approach, blueprint or quality perspective that defines the quality attributes and non-functional requirements of a software system used in binding the business requirement and technical implementation into a quality software system.
Other	The role, other, describes a person with a role similar to one described above, but with responsibilities different to those defined above, such as software tester.

The "Other" option is included to cater for respondents that are unable to map their respective roles to the available answers. The approach allows the researcher to determine if the response should be included in the analysis, or if the answer given could be translated into one of the defined roles, which would then result in the response being included for analysis.

Factors that influence software development outcomes

Factors that influence software development outcomes are considered from two different viewpoints, namely success or failure. Questions created for both viewpoints are equal in terms of what is being asked, but the questions are structured such that they can be used in conjunction with the presentation of the viewpoint.

For success, respondents are asked to focus on a software development project they were previously involved in, where the outcome was accepted as a success by all concerned. Similarly; for failure, respondents are asked to focus on another software development project they were involved in, but where the outcome was accepted as a failure by all concerned. Success and failure are presented in different sections and contained sixteen questions each. Annexure D (*Question design*) provides a description of how each questionnaire question was formed, and Annexure E (*Questionnaire*) provides the final version of the questionnaire.

Piloting

Creswell (2014) suggested that the testing or piloting of a questionnaire "is important to establish the content validity of scores on an instrument and to improve questions, format, and scales" (Creswell, 2014). A piloting phase is conducted to evaluate the questionnaire, its structure and answering mechanism. It is equally important to validate the appropriateness of the questions posed to participants, by comparing respondent answers to their interpretations of the question. Validation is done by selecting a testing group consisting of one expert per role demographic, asking them to complete the questionnaire and rate the appropriateness of each question. In addition, the testing group could provide written explanations to describe their perspective in terms of the questions structure, simplicity, understandability, ease of answering, relevance to their role and overall appropriateness of the questionnaire. Anomalies are recorded, and the data is used to adapt questions where required. The updated survey is again presented to the test group for final validation. The final version of the questionnaire is provided as Annexure E (*Questionnaire*) in the accompanying documents.

Sample design

Kothari (2004) suggested that sample design could be approached from two perspectives, probability and non-probability sampling. The author stated that with "probability samples each element has a known probability of being included in the sample but the non-probability samples do not allow the researcher to determine this probability" (Kothari, 2004).

The intended population are residents of South Africa. The software engineering population consists of individuals with diverse socioeconomic backgrounds but are all employed full-time as experts in the software development industry. The research focused on experts in software development teams suggesting a fundamental requirement that questionnaire respondents had to be full-time employees. As such, it is suggested that participants had to be between the ages of 18 and 65, as individuals under the age of 18 were not considered professional contributors in the industry.

The sample design was found to be crucial as it had a direct relationship to the overall generalisation of the research. In calculating the research sample size, a questionnaire response rate was revealed that, if it could be secured, could be considered appropriate for generalisation. Abeyasekera (2005) suggested that "if results are to be generalized [*sic*] to a wider population, sample sizes and methods of sampling are key issues that need consideration" (Abeyasekera, 2005). However, Neuman (2013) stated that "two key ideas to remember about representative samples are that: (1) self-selection yields a nonrepresentative [*sic*] sample and (2) a big sample size alone is not enough to make a sample representative" (Neuman, 2013).

For this research, the following sample designs were considered appropriate for collecting research data.

Single- or multistage sampling

Single- or multistage sampling is a probability sampling technique (Kothari, 2004). Kothari (2004) suggested that "cluster sampling involves grouping the population and then selecting the groups or the clusters rather than individual elements for inclusion in the sample" (Ibid.). Creswell (2014) suggested that in cases where the elements of a population were unavailable, sampling design techniques such as single-stage or multi-stage was valuable in uncovering the population (Creswell, 2014). The author further stated that in a "multistage or clustering procedure, the researcher first identifies clusters (groups or organizations [*sic*]), obtains names of individuals within those clusters, and then samples within them" (Ibid.). In terms of this research, an extensive list containing 1200 experts is available. This list contains a population of software experts employed full-time in the South African software industry. As such, a multi-stage approach is not required as this list is considered appropriate for sampling.

Quota sampling

Kothari (2004) suggested that quota sampling was often used in conjunction with stratified sampling (Kothari, 2004). In understanding the methods required to perform the research, it was uncovered that the overall research sample would require two types of samples. The first called a "defining sample", consisting of software development experts working together as a software development team and the second called the "general sample" consisting of experts in the industry.

It is suggested that the defining sample could inform the makeup of the general sample and as such enable stratified sampling during analysis. It is viewed that the incorporation of both approaches has the potential to increase the generalisation of the research outcomes. The essence of this method is visible in the fundamental design of the quota sampling technique.

Neuman (2013) defined quota sampling as "A non-random sample in which the researcher first identifies general categories into which cases or people will be placed and then selects

cases to reach a predetermined number in each category". The author suggested that "well-designed quota sampling is an acceptable nonprobability substitute method for producing a quasi-representative sample" (Neuman, 2013). The author noted that; relevant categories amongst the population had to be captured, such as age or gender to describe the diversity of the population. Then, the research would attempt to determine the number of cases per category (the quota) and then fix the number of situations in each category at the start of the data collection process (Ibid.).

The quota sampling technique has some advantages and disadvantages to consider. This technique provides a guarantee that the sample would have some diversity. However, quota sampling is only able to capture some aspects of the population's diversity and ignores others. Neuman (2013) suggested that quota sampling introduced the risk that "the fixed number of cases in each category may not accurately reflect the proportion of cases in the total population for the category" (Ibid.).

The defining sample is considered nothing more than a subset of the overall research sample, but what distinguishes this group from the general sample is that it would require experts working in the same team and on the same software development initiatives. Kothari (2004) noted that "The size of the quota for each stratum is generally proportionate to the size of that stratum in the population" (Kothari, 2004). However, the value this sample adds to the research is its ability to specify the makeup of the general sample required for the stratified sampling during analysis.

Stratified sampling

Kothari (2004) suggested that "if the population from which a sample is to be drawn does not constitute a homogeneous group, then stratified sampling technique is applied to obtain a representative sample" (Kothari, 2004). The author further noted that the population was 'stratified' into non-overlapping strata (subpopulations) and sample items were selected from each (Ibid.). It is viewed that this technique is valuable in understanding and describing the fragmentation of the population, but not for selecting the research sample. It is however considered that in explaining the research outcomes in

terms of data analysis, stratified sampling is beneficial in demonstrating relationships between the independent variables suggested by the research theory.

Sample size

The sample design provides a clear understanding of how the population could be incorporated to provide data for the study. Even though it could be considered a representative population, the design alone is limited in fulfilling the overall generalisation requirement, and an appropriate sample size is required. Sale et al. (2002) suggested that when considering generalisation, larger sample sizes were needed to allow for statistical methods capable of ensuring that a representative perspective could be attained (Sale, Lohfeld and Brazil, 2002). Neuman (2013) suggested that a predetermined sample size was not required when non-probability techniques were used in designing the sample, as the approach was aimed at first learning more about the population. However, the author stated that probability samples are a "preplanned approach based on mathematical theory" (Neuman, 2013).

As single-stage sampling is considered a primary sampling technique, an ideal sample size is required. In Addition, to facilitate the quota sampling technique, two separate sample groups are created once the sample size is determined. The first is the defining sample consisting of software development experts working together as a software team, and the second, the general population composed of experts in the industry. Data is collected from the first group, and the fragmentation present in the defining group is used to calculate the numbers required for the first group based on the initial sample size presented in Annexure A (*Sample size calculation reasoning*).

1. Calculated sample size

In computing the required sample size, the following question is created to provide guidance:

> *How large should a selected sample be in order to provide a 95% (**z** = 1.96) confidence interval with a margin of error (**e**) of 5, assuming a total population (**N**) of 250,000 with a normal distribution (**p**) of 50%?*

Table 8 Sample size calculation attributes

Attribute	**Symbol**	**%**	**Value**
Margin of error	*e*	5%	0.05
Confidence interval	*z*	95%	1.96
Planning value (*Estimated)*	*N*	100%	250,000
Response distribution	*p*	50%	0.5
Sample size			384

The following algorithm is used to determine the required sample size:

Table 9 Sample size algorithm[13]

$$\boldsymbol{Sample\ size} = \frac{\frac{\boldsymbol{z^2} \times \boldsymbol{p(1-p)}}{\boldsymbol{e^2}}}{\boldsymbol{1} + \left(\frac{\boldsymbol{z^2} \times \boldsymbol{p(1-p)}}{\boldsymbol{e^2 N}}\right)}$$

[13] (de Leeuw, Hox and Dillman, 2008)

Table 10 Sample size calculation

$$Sample\ size = \frac{\frac{1.96^2 \times 0.5 \times (1 - 0.5)}{0.05^2}}{1 + \left(\frac{1.96^2 \times (0.5 \times (1 - 0.5))}{0.05^2 \times 250000}\right)}$$

$$Sample\ size = \frac{\frac{3.8416 \times 0.25}{0.0025}}{1 + \left(\frac{3.8416 \times 0.25}{0.0025 \times 250000}\right)}$$

$$Sample\ size = \frac{384.16}{1.000154}$$

$$Sample\ size = 384.1008$$

Annexure A (*Sample size calculation reasoning*) provides the full rationale on the numbers selected for margin of error (e) and estimated population size (N).

Population size

The population size is referred to as the total number of relevant individuals that could be used in the study. In terms of this research, the population size of the overall software engineering industry in South Africa is unknown, and as such, an estimated value of 250,000 is used.

Margin of error

The margin of error refers to the margin of mistakes that could be tolerated by a study and still be representative. It is suggested that lower margins require larger sample sizes. In calculating the required sample size, a standard of 5% is used as the margin of error.

Confidence level

Confidence level refers to the amount of uncertainty the study can tolerate. This measure represents the certainty that the sample accurately reflects the population, within the margin of error. For this study, a margin of error of 5% is chosen and as such a standard 95% confidence is considered appropriate.

Response distribution

The nature of this study is founded on uncertainty. Its purpose, however, is to investigate whether there are differences in opinions between different experts when factors that influence software development outcomes are considered. As such, it is suggested to use a 50% response distribution as this allows for the largest sample size.

Recommended sample size

In completing the recommended sample calculation, it is revealed that based on the parameters, a sample size of 384 respondents is required to demonstrate the generalisation of the research outcomes. However, it is understood that collecting data from the population would be challenging and as such, a response rate of no more than a third of the population is expected. Thus, it is decided to increase the target population to 1,200 to allow for a non-response rate of 66% while still allowing for the generalisation of the study.

2. Quota sample

Data had to be collected from a sample size smaller than that considered as a representative make up for teams in the software industry, in order to understand the makeup of the general sample. This group is called the defining sample as it could provide insight into how to collect data from a general sample in terms of the independent variable suggested for the study.

Once this process was completed, the fragmentation of this sample was analysed to understand what the makeup of the general sample had to be. The initial data collection surveyed 38 professional, working in the software engineering industry, on the same team and on the same software development projects. These experts were asked to complete the questionnaire in the same manner as planned for the general sample but were tracked separately as analysis in terms of the quota sample's demographic information was required before data collection against the general sample could start.

In terms of the defining sample, the analysis revealed that the fragmentation in terms of the primary independent variable (Role) provided an appropriate sample makeup. It was also found that including the secondary independent variable of generation and tenure, fragmented the quota sample too much, and incorporating them into defining the general sample would require an exponential increase in the overall response rate. As such, it is suggested that role alone be used to provide the makeup of the general sample. Table 11 (*Defining sample make up*) below, provides the fragmentation of the defining sample used in formulating the quota sample required for data collection from the general sample.

Table 11 Defining sample makeup

Role	**Count**	**Percentage of makeup**
Project manager	4	11%
Architects	2	5%
Business analysts	8	21%
Developers	24	63%
Total	38	

3. Required general sample:

The general sample consists of the population captured by the instrument and excludes responses collected for the defining sample. The selected instrument by its very nature has no control over who completes it but has control in terms of the selected audience. By focusing on the population makeup suggested by the defining sample, guidance is given in terms of focus in collecting data from the research population.

Based on the fragmentation proposed by the defining sample as well as the recommended sample size specified in Table 8 (*Sample size calculation attributes*) above, the questionnaire had to capture the respondents based on the fragmentation numbers provided in Table 12 (*General sample fragmentation requirement*) below.

Table 12 General sample fragmentation requirement

Role	Required responses
Project manager	36
Architects	18
Business analysts	73
Developers	219
Total	346

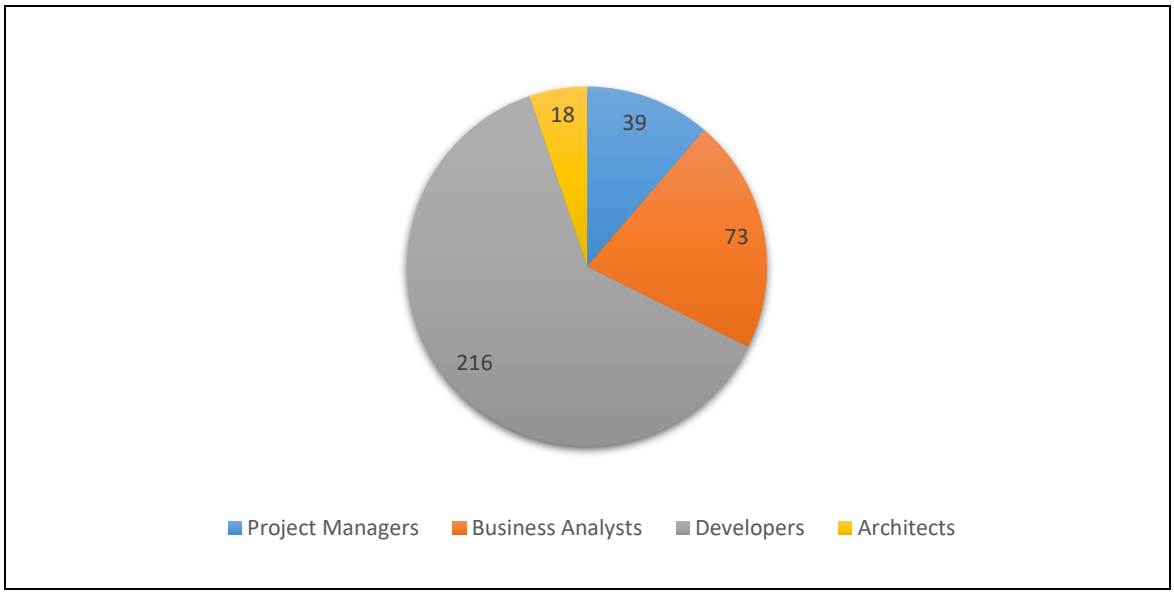


Figure 10 General sample makeup

4.2.3 Data collection

Kothari (2004) suggested that there were "several ways of collecting the appropriate data which differ considerably in context of money costs, time and other resources at the disposal of the researcher" (Kothari, 2004). It is known that data collection from a research population would be time-consuming and complex. It is believed that the local South African software industry is embodied by a large community of professionals, all potential candidates for the study, but accessing an audience of this magnitude would be challenging. In addition, the sampling techniques proposed for this study could reduce the numbers significantly, jeopardising the overall generalisation of the study.

As such, online surveying tools were employed to assist with the dissemination and collection of the questionnaires. It is noted that a primary benefit of using questionnaires is their reach, especially when online surveying tools are used as the vehicle for distribution and collection. These benefits are not limited to, but include:

- The flexibility in terms of the ability to further refine the questionnaire after it's release. Should anything require amending, such as the order in which the questions are posed, the questionnaire could easily be changed without introducing risk into the entire data collection process.

- The ability to pre-evaluate the sample. A pre-evaluation of the collected data would provide insight into the selected sample, allowing the researcher to augment the sample while the data is being collected, rather than waiting for all feedback to be consolidated in order to make decisions about the population.

- The accuracy in collecting data. As participants submit their answers directly into a system, accuracy is increased as there is no need to translate the given answers between mediums.

- Data exportability. The ability to export the data from an online surveying tool into a local database or data analysis tool is the most attractive benefit. Most online surveying tools have some capability of producing an analytical view of the collected data. In most cases, the analysis cannot be wholly defined to align with the research perspective and as such would be unable to replace research specific statistical analysis of the data. As such, it is considered paramount that a surveying tool must have the ability for a researcher to export the data from the instrument without introducing any change to the dataset.

Questionnaires were released in paper and electronic form. Paper collections were considered ad-hoc and only used where access to the electronic version was not available. The electronic format was facilitated by a proprietary data collection service. This service hosted the questionnaire and provided several different mechanisms, such as email, web

links and social networking channels, focusing on the collection of research data. The approach proved valuable in collecting data from a distributed and diverse participant base. In addition to the former, the data collection service allows for a "kiosk mode" where an electronic device such as a tablet could be used to collect manual responses. The "kiosk mode" facilitated collecting data from participants in random groups, such as community events.

4.2.4 Data analysis

Neuman (2013) suggested that the process of data analysis usually requires the use of computer software to assist in creating statistical measures as these provided a condensed picture of the data (Neuman, 2013). The process defined in this research includes sampling the population, applying coding to the datasets and then establishing the trends visible in the data. The following sections explain each of these in more detail.

Population

As discussed above, an initial collection of data is done on a smaller scale, referred to as the defining sample. This sample represents a software development team working together as a team on the same development project. This approach was favoured as the defining sample had the ability to define what the general sample would consist of in terms of the independent variable of the study. Wilson and Stern (2001) suggested that analysis was typically completed in three stages, including the exploratory analysis, deriving the meaning and archiving of the data. The authors noted that exploratory analysis was used "to get an idea of what there is" to assist in understanding how to work with the data (Wilson and Stern, 2001). As such, once the initial defining sample data is collected, an exploratory analysis is done on this dataset in order to understand the makeup of this sample. This understanding provides insight into what data is required from the general sample and based on the calculated sample size; a quota is established that specifies the necessary numbers for each role in the study.

Strategy

This research suggests that in the absence of formal approaches aimed at collecting empirical evidence on failure or success of software development projects, experts are forced to base the case and effect of software development outcomes on their perspectives, and proposes that the research aims at uncovering whether experts in different roles, generation and tenure have different perspectives of factors that influence software development outcomes. As such, a strategy to analyse the data associated with the research has to include measurements capable of showing a significance, probability and association between experts in different roles, generation and tenure.

The quota sample used in defining the general sample was incorporated to allow "post-stratified" sampling on the data. Neville (2007) suggested that stratified sampling included "sampling within particular sections of the target groups" (Neville, 2007). Kothari (2004) suggested that "if the population from which a sample is to be drawn does not constitute a homogeneous group, then a stratified sampling technique is applied so as to obtain a representative sample" (Kothari, 2004). The author further noted that the population was 'stratified' into non-overlapping strata (subpopulations) and sample items were selected from each (Ibid.). It is viewed that this technique is valuable in understanding and describing the fragmentation of the population, but not for selecting the research sample. It is however considered that when explaining the research outcomes in terms of data analysis, stratified sampling is beneficial in demonstrating relationships between the independent variables suggested by the research theory. When considering a stratum that defines the roles within the software development industry, stratified sampling is found to be valuable. This approach allowed for analysis across all independent variables and assisted in accepting or rejecting hypothesis H2 to H4.

The following strata were proposed to test each hypothesis:

H2 Stratum

This stratum allows for the comparison of data against the different roles included in the study. It is suggested that the comparison of roles be used as the primary measure as a representative sample could then be tested (*Table 13 H2 Stratum below*). This stratum could, however, be applied to generations and tenure as well, but it is evident that it would have to use less of the sample for comparison in order to cater for the lowest common denominator. For example: if the lowest numbers are collected from the Late Generation X – Early Boomers, then this figure is to be considered the highest number of responses that can be used for analysing data based on the generation group. Equally, should the number of responses collected for a senior level presented in the tenure group be the lowest, then this number would constrain the total number of replies used across all tenures. This approach is required, and generation and tenure are not catered for in the quota sample. Table 14 (*H3 Stratum*) and Table 15 (*H4 Stratum*) below, provide strata for the generation and tenure strata respectively.

Table 13 H2 Stratum

Project managers	Architects	Business Analysts	Developers

H3 & H4 Strata

It is noted that the inclusion of the categories, generation and tenure would reduce the overall analysis sample due to the quota sample incorporating role as a primary categorisation. As stated above, the independent variables of generation and tenure are marginally related to the formation of software development teams, and as such, when comparing these variables, an equal sample size from each category would have to be used. This approach has the effect of reducing the maximum number of respondents in each category to the lowest number of all categories.

Table 14 H3 Stratum

Generation Z	Early – Mid Generation Y	Late Generation Y – Early Generation X	Mid – Late Generation X	Late Generation X – Early Boomers

Table 15 H4 Stratum

Learner level	Junior level	Junior to intermediate level	Intermediate to senior level	Senior level

Measurement

Measure of Association

Neuman (2013) suggested that the measure of association was considered "a single number that expresses the strength, and often the direction, of a relationship" (Neuman, 2013). The author further noted that it had the effect of summarising information about multiple variables and their relationships with each other into a single number. Neuman (2013) suggested that several measures existed, but that most followed a "proportionate reduction in error", testing the independence of one variable in relation to another to establish its level of association (Ibid.). The purpose of this research is to establish whether there is a difference (relationship) in terms of how experts view factors that influence software development outcomes.

Macdonald and Headlam (2008) noted that measurement models were divided into one of two families that included parametric and nonparametric tests. The authors suggested that in contrast to nonparametric tests, models from the parametric family were more appropriate when the data were sampled from a population that followed a normal distribution, signifying that the data was less inclined to produce "unusually extreme values" (Macdonald and Headlam, 2008).

In considering this research, it is understood that the existence of a relationship between the dependent variable and independent variables would explain the data to some degree,

but additional measures are required to demonstrate the research phenomenon. Also, the research objective is to understand if there is a correlation between factors that influence software development outcomes and the opinions of experts on such, and not to explain why the relationship existed. Kothari (2004) suggested that "there are several methods of determining the relationship between variables, but no method can tell us for certain that a correlation is indicative of a causal relationship" (Kothari, 2004). As such, it is understood that incorporating relationship summary techniques such as cross-tabulation, mean, median and mode as well as standard deviation are sufficient in describing variable relationships from a bivariate perspective. However, incorporating relationship exploration techniques, such as Pearson Correlation and Chi-square as well as significance testing techniques, such as the T-Test would benefit the research by augmenting the relationship information with a description of the strength between variables, the association between them and finally, the significance of difference, should there be any.

Colman and Pulford (2008) provided a model that could be used in choosing statistical procedures (*Annexure H Choosing an appropriate statistical procedure*) (Colman and Pulford, 2008). This model is used in determining the appropriateness of the statistical techniques selected for this study.

1. Relationship summary

Campbell and Campbell (2008) suggested that regression was a statistical technique used to "determine the linear relationship between two or more variables" (Campbell and Campbell, 2008). The authors noted that bivariate tables were often used as the simplest form of regression analysis as they merely highlighted the relationship between one independent variable and one dependent variable. The authors further stated that regression was limited in that it was unable to explain causation (Ibid.). When considering variables and the relationships between them, Kothari (2004) noted three perspectives that suggested relationships between variables. These included a single variable (univariate), two variables (bivariate) and more than two variables (multivariate) (Kothari, 2004). It is noted that the comparison of each independent variable against the dependent variable

would provide an appropriate level of comparison to test the research theory. As such, bivariate tables are favoured as the regression technique.

a. Cross-tabulations

Neuman (2013) suggested that cross-tabulation is the "process of placing data for two variables in a contingency table to show the percentage or number of cases at the intersection of variable categories" (Neuman, 2013). Macdonald and Headlam (2008) noted that "a cross-tabulation gives you a basic picture of how two variables inter-relate" and suggested that this approach had the ability to provide a powerful mechanism for testing relationships between variables (Macdonald and Headlam, 2008). This technique is incorporated as it could describe a summarised view of each independent variable in its most basic form. Cross-tabulation is considered the most basic form of bivariate tables and could provide a visual representation of the stratified samples with each category describing its relationship to the dependent variable. This technique is considered the primary visualisation technique as it is commonly used to explore relationships between multiple variables and could be visualised in tabular form (Neuman, 2013).

b. Mean, median and mode

Macdonald and Headlam (2008) suggested that mean commonly referred to as average but considered this generalisation of the term improper when considering "arithmetic mean", as averages could be measured regarding mean, median and mode (Macdonald and Headlam, 2008). Kothari (2004) suggested that this central tendency or statistical average was commonly used to summarise survey data and when considering averages, the combination of mean, median and mode could increase the overall understanding of large amounts of data as it reduces the view to a single value (Kothari, 2004). The author further noted that mean, median and mode were considered the most commonly used and popular measures for statistical averages (Ibid.). Neuman (2013) suggested that central tendency (usually mean) was another type of bivariate table, capable of reducing large amounts of data (Neuman, 2013). In terms of the study and the large amount of data required for analysis, mean is considered a useful perspective when comparing each independent

variable against the dependent variable. Mean, in its native form can reduce each role, tenure or generation to a single number that can be compared to other roles, tenures and generations.

c. Standard deviation

Macdonald and Headlam (2008) suggested that standard deviation was "a descriptive statistic used to measure the degree of variability within a set of scores" (Macdonald and Headlam, 2008). The authors further stated that when comparing sets of data that may or may not have the same mean but different ranges, the standard deviation would be able to measure the "spread of the data about the mean value" (Ibid.). In terms of this research, the standard deviation is considering the first step in discovering and describing the research phenomenon that suggested that there may or may not be a difference in opinion of factors that influence software development outcomes between different roles, generations and tenure.

2. Relationships exploration and significance testing

In addition to defining relationships between variables, a certain amount of effort is required to understand these relationships. The process of exploring relationships includes techniques such as Pearson's Correlation and the Chi-square test for association. These techniques are considered valuable in augmenting the understanding of relationships between variables in terms of its strength and testing the research hypothesis.

a. Pearson Correlation

The Pearson correlation model is regarded as a common parametric measure and describes a test of the strength of association between variables. Macdonald and Headlam (2008) suggested that "a correlation of +1 means that there is a perfect positive linear relationship between variables" and "a correlation of -1 means that there is a perfect negative linear relationship between variables" (Macdonald and Headlam, 2008). Kothari (2004) noted that the Pearson's Coefficient of Correlation or simple correlation was "the most widely used method of measuring the degree of relationship between two variables" (Kothari,

2004). The author noted that the coefficient, however, assumes that there was a linear relationship between variables, variables were causally related, and one was dependent and the other independent and several independent causes operated in both variables to produce a normal distribution (Ibid.). In terms of this research, Pearson's Correlation is considered appropriate as its assumptions could be validated by other analysis techniques used in the study, such as mean, median, mode and standard deviation.

b. T-Test

In addition to the Pearson's Chi-square test for association, the one-sample T-Test is considered a mechanism capable of testing the significance of relationships between variables. Macdonald and Headlam (2008) suggested that the "T-Test assesses whether the means of two groups are statistically different from each other" (Macdonald and Headlam, 2008). Creswell (2014) suggested that T-Tests were common in testing research hypothesis and noted that T-Tests were used in "experimental designs with categorical information (groups) on the independent variable and continuous information on the dependent variable" (Creswell, 2014). Borden (2016) confirmed this approach by suggesting that T-Tests provided "information about the results of the comparisons between the two means" (Borden, 2016). Also, DeCoster (2006) suggested that T-Tests were appropriate as the simplest hypothesis tests when comparing means and noted that the "One-sample T-Test assumes that the variable you are testing follows a normal distribution" (DeCoster, 2006). This assumption could be managed in terms of the additional analysis techniques used in the study, and as such, the T-Test is considered a primary mechanism for describing the acceptance or rejection of the research hypothesis.

4.2.5 Interpret data

Neuman (2013) suggested that the process of interpreting data in terms of a quantitative study is of utmost importance as it combines the data analysis and research knowledge with an existing theory in order to answer the research question. The author noted that in considering alternative interpretations of the data as well as data associated with part studies, the researcher would be able to " draw out wider implications of what we have

learned" (Neuman, 2013). This process is included in the research design in order to extend the knowledge base by providing a clear and concise understanding of the research phenomenon in terms of how software development methodologies evolve. Vosloo (2014) suggested that "comparative analysis provides an alternative for statistical analysis, which bears closer relation to the positivist views" (Vosloo, 2014). In terms of this research, a positivist approach suggests that by illustrating that experts from different social classes view the factors that influence software development outcomes differently, it can be proven that the reality in relation to the evolution of software development practice and the meaning it has for the software industry is independent of the consciousness of the impact it has on software development outcomes.

In providing a consolidated understanding of the data collected, the research theory and evidence found in existing bodies of knowledge, a strong case can be made about the flaws present in software methodology.

4.2.6 Inform

The final step in the research design suggests that the knowledge accumulated by all other processes would require validation and dissemination. In terms of the methods defined for this study, it is proposed that a focus group is considered to discuss the outcomes of the research. This approach is incorporated in order to act as a sounding board against which the interpretation could be validated. In addition, this method could recognise any biases that may have been inadvertently introduced into the study. The inform phase is divided into two primary activities, including expert discussion and conclude.

Expert discussion

The expert discussion activity includes a focus group populated by professional experts as well as academics in the field of software engineering. The process includes briefing the group on the aim of the research, proposing the theory developed for the study, presenting key findings and trends discovered by the study and finally facilitating a quasi-structured discussion on the topic. This approach allows participants to air their views on the outcomes and provide an opinion on the interpretation of the data and findings. The

discussion is recorded and incorporated into the conclusion of the dissertation. The approach provides insight into un-tested limitations, validity and future research on the topic.

Conclude

The conclude activity incorporates the expert perspectives collected from the focus group and provides a discussion in terms of validation and critique of the research outcomes. This approach provides a richer explanation capable of increasing understanding and testing the research theory.

This section provides a summary of the research methodology selected for the research and as such is considered appropriate for all reader types.			

4.3 Summary

In addition to the research theory defined in chapter 3, a research question is posed to understand what the difference in recognising factors that influence bespoke software development outcomes in South Africa are between software development experts in different groups of a software development team.

For the research design, a correlational design to quantitative research is proposed as the study relies on describing the significance of relationships between perceptions of factors that influence software development outcomes and different groups in the software development teams. The design incorporates multiple steps for defining appropriate research instruments, collecting data from a generalizable population, statistically analysing the data and finally interpreting and validating the research findings in an ethical manner.

The following chapter provides the results of the research in terms of the processes defined in this chapter.

5 Data analysis

The purpose of any research is to investigate and realistically report on a phenomenon. This research, like many others, attempts to gain a deeper understanding of a phenomenon that warrants investigation. In its most simplistic form, the phenomenon under investigation is to understand if an expert's social class has an impact on perceptions of software development outcomes. A theory is created that suggests the possibility that this phenomenon exists, and in defining a way to research this theory, several processes are incorporated into the study. This data analysis process follows a data collection process where the relevant population is sampled and asked to contribute to a survey study, designed to elicit information that could be used to test the theory. The purpose of the data analysis process is to unpack the data that is collected and using statistical measures, uncover information in a formal and unbiased manner.

In terms of the research methodology, the data analysis design suggests that measures of association are beneficial during analysis, as the ultimate purpose of the study is to test for the existence of differing perspectives between different categories. As such, data analysis incorporates several approaches to firstly gain a superficial understanding of differences by comparing the means of each category and then lastly, perform in-depth statistical analysis to formally uncover the significance of associations using T-Tests and Pearson's correlation.

This section provides a description of the bounded context or boundary selected for the research as it relates to the research question. The description explains the reasoning for the inclusion of only certain dataset into the research analysis process. As such, this section is considered most valuable to academic readers.

5.1 Data boundary

The data analysis phase focuses on a subset of data that represents the core of the research question and its associated hypotheses. It is found that a decomposition of the research question and hypotheses results in a multitude of different subsets of data. Also, these subsets could be decomposed even further into datasets that represent software outcomes in terms of successes and failures. The aggregation of software failure and success factors,

as well as factor categorisations into the primary research categories of "internal variability" and "political impacts", fragments the data even further. The fragmentation results in a total of 135 differing viewpoints, each with its own dataset and each capable of explaining the research phenomenon from a different perspective (*See Analysis boundary in Annexure I research Data Analysis for more detail*).

As such, the data analysis process is bounded to only three datasets that represent the research question and hypotheses in its natural form. These include the three independent variables of role, generation and tenure against the most basic form of the dependent variable, software outcomes. Software outcomes is a consolidation of the combined view of "internal variability" and "political impacts" as well as successful and failed software development projects. Analysis of each of the granular datasets was done but is not included in this dissertation.

This section provides a description of the guidelines selected for data analysis and provides details relevant to hypothesis and relationship testing and as such, is considered most valuable to academic readers.



5.2 Measurement guidelines

The data analysis phase uses several statistical measures. These include measures to create an initial opinion (relationship summary) of the data such as mean, median, mode and standard deviation as well as measures to explore the significance of the association between variables (relationship exploration), such as the common T-Test, Student's probability test and Pearson's Correlation Coefficient. When considering the relationship summary, no guidance is available, and the summary is based on the perspective of the reader. As such, these are merely considered claims made by the researcher about the data. However, when considering relationship exploration, several industry standards are available to guide the outcomes of the data. This section explains these guidance rules as they are used in the study.

5.2.1 Hypothesis testing

Hypothesis testing is considered an appropriate way to test the validity of the claims made during relationship summary. DeCoster (2006) noted that this technique was commonly used in statistical analysis to measure the significance of relationships between variables (DeCoster, 2006). For this study, the T-Test, as well as the Student's probability test, are included to measure relationships between each of the independent variables compared to the dependent variable. Borden (2016) suggested that T-Test were common mechanisms to analyse differences between two means as this approach was capable of distinguishing between an actual difference and chance (Borden, 2016). Both the common T-Test and Student's probability test results in a P-Value that represent the association between the means of two variables or its significance level. Borden (2016) suggested that the "significance level tells you if the difference observed between the means was greater than would be expected by chance" (Borden, 2016). The P-Value is then used to test the null hypothesis of the association between the two variables. For example, consider the null hypothesis of the means of two variables: when subtracting the mean of one variable from another, the result would be zero (H0: $\mu_{\text{variable a}} - \mu_{\text{variable b}} = 0$).

Quantitative study is anchored in statistical significance. Coughlan, Cronin, and Ryan (2007) noted that statistical significance assisted in ruling out the threat to validity, where "results could be due to chance rather than to real differences in the population" (Coughlan, Cronin and Ryan, 2007). The authors suggested that statistics is about probability and quantitative studies typically identify with the lowest level of significance and as such, only indicate that value is significant if the probability value was equal to or less than 0.05, meaning that the likelihood of chance is less than 5%.

For this research, an industry standard is adopted that suggests that when the P-Value of a relationship is less than 0.01, then strong evidence exists that a significant difference can be found between the means of the two variables. Equally, when the P-Value of a relationship is less than or equal to 0.05, sufficient evidence exists suggesting a significant difference between the means of the two variables. However, should the P-Value of the

relationship be more than 0.06, it can be argued that a difference between the means exists for the relationship, but it cannot be considered statistically significant.

As such, for hypothesis testing, any P-Value of 0.05 and less would result in the null hypothesis being rejected and any P-Value of 0.06 and more would lead to the null hypothesis being accepted.

5.2.2 Relationship testing

The Pearson correlation method is a parametric correlation test that depends on a distribution of data. Macdonald and Headlam (2008) suggested that this approach was a "common measure of the correlation between two variables" (Macdonald and Headlam, 2008). This method is used to measure the linear dependence between each relationship or the strength and direction of a linear relationship between two variables. The Pearson correlation formula produces an R-Value that is between + 1 and –1, where numbers greater than 0 are considered positive relational and numbers smaller than 0, negative relational. The closer the number is to 0, the greater the variation is between data points. As such, more variations describe weaker relationships between datasets.

For this research, an industry standard is adopted that suggests that when the R-Value is 0, no linear relationship exists. Positive linear relationships exist when the value is greater than 0, and negative linear relationships exist when the value is smaller than zero. The strength of the relationships is measured per (+-)0.3 for weak, (+-)0.5, moderate and (+-)0.7, strong. In addition, (+-)0.5 is considered an appropriate cut-off mark for this research, suggesting that any value greater than +0.5 and smaller than -0.5 exhibits a significant linear relationship, where values between +0.5 and -0.5 exhibit an insignificant linear relationship.

This section provides a description of the sample used in the research and provides insight in terms of the sample makeup as it relates to the independent variables of role, generation and tenure. As such, this section is considered most valuable to academic readers.





5.3 Sample

The research incorporates both quota and stratified sampling techniques to create a research sample representative of a general software development population.

The research suggests that experts with different social classes reason differently when factors that influence software development outcomes are considered. The research aims at understanding this phenomenon by providing a theory that incorporates three variables capable of representing social class. These variables include role, generation and tenure. Each of these provides a different perspective capable of acting as a filter on the data collected for the research. In addition, each variable contains several non-homogenous groups described as the primary strata of the research. Kothari (2004) suggested that "if the population from which a sample is to be drawn does not constitute a homogeneous group, then stratified sampling technique is applied so as to obtain a representative sample" (Kothari, 2004). This approach is valuable in understanding and describing the fragmentation of the population as well as describing relationships between sets of data.

The research design suggests that an appropriate population size include 384 experts in the field of software engineering, but before data can be collected, a better understanding of the makeup of the general population is required. A quota sampling technique is incorporated to allow for the collection of data against a smaller sample, that is considered representative of the makeup of software development teams in the industry.

Table 16 Defining sample makeup

Role	Count	Percentage of makeup
Project manager	4	11%
Architects	2	5%
Business analysts	8	21%
Developers	24	63%
Total	38	

This group is called the defining sample as it can provide insight into how to collect data from a general sample in terms of the independent variable suggested for the study. The breakup of the defining sample reveals that the total number of respondents must align to a makeup that facilitates 11% project manager responses, 5% architect responses, 21% business analyst responses and 63% developer responses as suggested in Annexure A (*Sample size calculation reasoning)*. This breakdown provides a quota in terms of the total number of respondents of 384 and each role specified by the research, requiring a total of 40 project manager responses, 20 architect responses, 81 business analyst responses and 243 developer responses.

It is worth noting that the research population's sampling is based on the fragmentation of roles in a software development project only and tenure and generation numbers are not considered part of the quota samples for focused data collection.

Stratified sampling is used to create similar views of the independent variables. Once the data is collected, it can be divided into three strata, each representing the independent variables of role, tenure and generation. Data analysis is done in terms of these strata as they relate to sub-research questions 2, 3 and 4. The following section explains the stratification of the population in more detail.

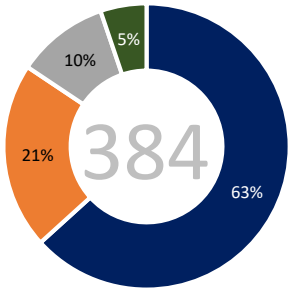
H2 Stratum: Role

The collection of responses in terms of predefined quotas provides structure and allows for the collection of the correct number of responses required by the research design. The H2 stratum is considered the primary perspective to investigate as it is both the primary independent variable as well as the guide in terms of defining the general research sample. The inclusion of role as a stratum allows analysing the research data from a specific vantage point that includes perspectives from developers, business analysts, project managers and architects. The stratum makeup is guided by the initial data collection exercise against a known defining sample that facilitates a quota required for each role. The stratum includes a requirement for 63% or 243 developers, 21% or 81 business

analysts, 10% or 40 project managers and 5% or 20 architects. Table 17 (*H2 Stratum make up*) below, illustrates the makeup of the H2 stratum.

Table 17: H2 Stratum make up

Role	Cnt	Makeup
Developer	243	63.28%
Business Analyst	81	21.09%
Project Manager	40	10.42%
Architect	20	5.21%



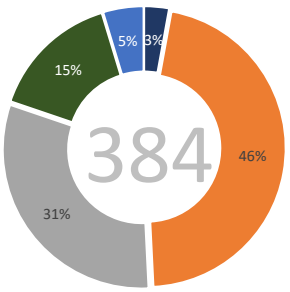
5.3.1 H3 Stratum: Generation

The H3 stratum is considered a secondary viewpoint and includes examining research data from the perspective of a generation. Even though this stratum is secondary to the role category, it is found to be important as it offers an interesting dimension in understanding the data. McCrindle (2010) suggested that "while age influences behaviour and attitudes, greater impacts are made by the culture in which one lives out one's youth, as well as social markers – significant events during one's formative years" (McCrindle, 2010). The author further noted that the characteristics associated with generation groups were not due to influences of life stages but rather because people at various ages live through the same events, creating different world-views at different ages (Ibid.).

The makeup of this stratum is not guided by the initial quota sampling exercise as it has the potential to over-inflate the research sample with little value. As such, the stratum is naturally formed as per the roles selected for the quota sample. A stratum could, however, be specified as the data collection process could collect respondents in all generational categories. Table 18 (*H3 Stratum make up*) below, illustrates the stratum makeup and the number of respondents collected against each category.

Table 18: H3 Stratum make up

Generation	Cnt	Makeup
Generation Z	11	2.87%
Early-Mid Generation Y	178	46.35%
Late Generation Y - Early Generation X	119	30.99%
Mid - Late Generation X	58	15.10%
Late Generation X - Early Boomers	18	4.69%



5.3.2 H4 Stratum: Tenure

The H4 stratum is considered a secondary viewpoint and includes examining research data from a tenure perspective. Tenure is regarded as an appropriate viewpoint to consider in data analysis, but its definition must be considered carefully to collect appropriately correct data. The questionnaire design chose to use years of experience rather than a tenure label to allow respondents to be truthful when answering the tenure question. This approach has the potential to reduce inaccuracies by allowing respondents to select a number instead of a tenure label.

For this research, respondents were asked to select a year range that represented their experience that was then translated into a tenure classification used in the stratum.

The tenure classification includes the following:

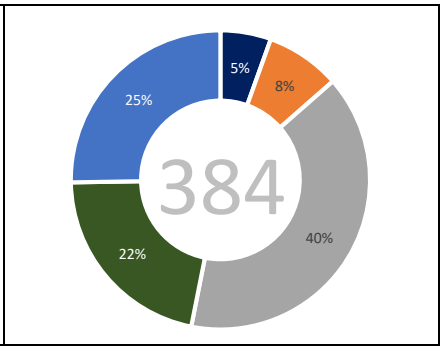
- For professional experience of less than 1 year, respondents are considered to be at a Learner level.
- For professional experience of at least 1 year, but less than 2 years, respondents are considered to be at a Junior level.
- For professional experience of at least 2 years, but less than 7 years, respondents are considered at a Junior to Intermediate level.
- For professional experience of at least 7 years, but less than 12 years, respondents are considered at an Intermediate to Senior level.

- For professional experience of more than 12 years, respondents are considered at a Senior level.

The makeup of the H4 stratum is not guided by the initial quota sampling exercise as it has the potential to over-inflate the research sample with little value. As such, the stratum is naturally formed as per the roles selected for the quota sample. A stratum could, however, be specified as the data collection process could collect respondents in all generation categories. Table 19 (*H4 Stratum make up*) below, illustrates the stratum makeup and the number of respondents collected against each category.

Table 19: H4 Stratum make up

Tenure	Cnt	Makeup
Learner level	21	5.47%
Junior level	31	8.07%
Junior to intermediate level	152	39.59%
Intermediate to Senior level	83	21.61%
Senior level	97	25.26%



This section provides a conclusive view of what the data analysis revealed. Each independent variable is described in terms of a dependent and covariant relationship. This section is considered a valuable read to both academic and pragmatic readers.

Note: For an in-depth view of the sample makeup, see Sample in Annexure I (Research Data Analysis).

5.4 Research findings

5.4.1 Introduction

This study argues that in the absence of formal approaches aimed at collecting empirical evidence on the outcomes of software development initiatives, expert opinions and perspectives are recorded and used as factual guidance in the evolution of software practice, as experts innovate in this space. As such, it is argued that methodologies are underpinned by bias, rather than fact. The research associated with this dissertation

provides a theory based on this phenomenon and incorporates a methodology, designed to investigate the relevance of the differences between experts in different roles, tenure and generation groups. In strictly and formally following the method, data is collected and statistically analysed. This section provides a high-level view of the outcomes of the data analysis of the research.

5.4.2 Description

This section provides a high-level view of what was discovered during the data analysis process. For a detailed analysis of the data, refer to Annexure I (*Research Data Analysis*).

The following sections describe each independent variable of role, generation and tenure in relation to the dependent variable of software development outcomes. Each section describes claims made about the relationships of the categories present in the internal variable by focusing on the difference in the means collected for each category. It then incorporates hypothesis testing to test for a significant statistical difference between each relationship.

Data source

The following sub-sections explain the data in the highest aggregated state. The data used represents a combination of failed and successful software development projects as well as "internal variability" and "political impacts" as factor categorisations. This combination represents an expert's perspective of software outcomes, regardless of the factors responsible for a positive or negative software development outcome. Please see Section 1.2 Analysis boundary in Annexure I for an explanation on the selection of appropriate data views. To increase understanding, each subsection is tagged with its associated view.

Role – View 25

It is suggested that experts in different roles reason differently when factors that influence software outcomes are considered. It is noted that designation or "job role" is perceived differently as several different designations have been used in the industry to describe

similar functions. As such, the study focuses on role in terms of responsibility, rather than an individual's designation. This approach reduces the complexity as respondents could associate themselves with a responsibility more easily than with a different designation. As such, role as an independent variable is limited to developer, business analyst, project manager and architect.

The following research sub-question (RQ2) is posed to describe the phenomenon:

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts in different roles?*

The following hypotheses are posed to allow testing of the question:

- H02: There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognized among software development experts in different roles.
- H2: Experts in different roles in a software development team have differing views of "internal variability" and "political impacts" as the factors that influence bespoke software development outcomes in South Africa.

In considering the role of an expert as a primary perspective for this research, a dataset is extracted that consolidates all research data into a single dataset that represents a combination of failed and successful software development projects as well as "internal variability" and "political impacts" as factor categorisations. This combination represents an expert's perspective on software outcomes, in terms of factors that influence software development. The independent variable of role is used as a lens to illustrate the relationship of experts in different roles as a primary viewpoint.

Dependence

When considering the means of each role, it can be argued that a difference in perspectives exists between each role. Figure 11 (*Role mean comparison*) below illustrates that the means for each role differ from 1.64% to 6.89%.

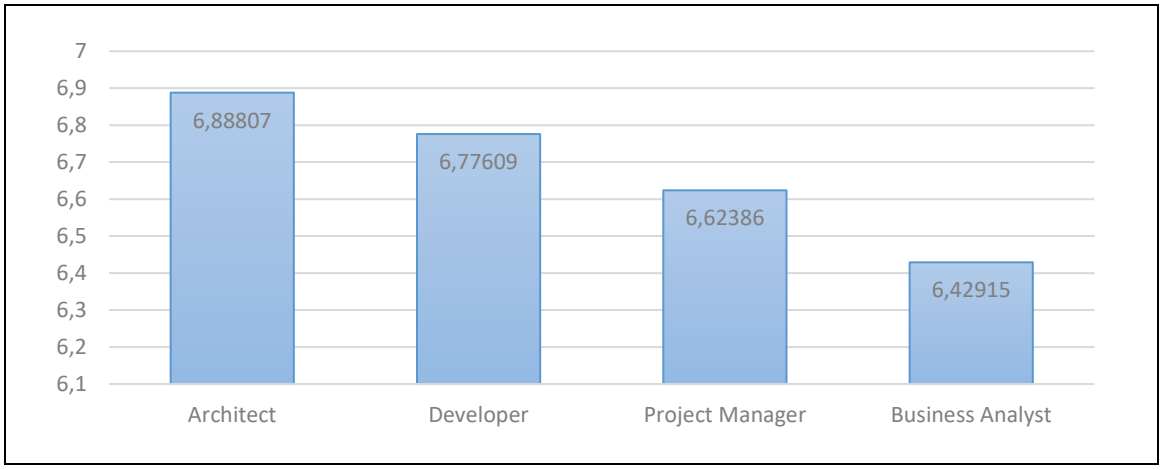


Figure 11: Role mean comparison

As such, the following claims are made based on the visible differences of means between each role.

- When considering architects and developers, there appears to be a difference of 1.64% between perspective
- When considering architects and project managers, there seems to be a difference of 3.91% between perspective
- When considering architects and business analysts, there seems to be a difference of 6.89% between perspective
- When considering developers and project managers, there appears to a difference of 2.27% between perspective
- When considering developers and business analysts, there seems to a difference of 5.25% between perspective
- When considering project managers and business analysts, there seems to a difference of 2.98% between perspective

From a pragmatic standpoint, these claims shed some light in terms of what the data is saying, but without statistical examination, little relevance can be proven. As such hypothesis testing is included to assist in describing the differences between roles, mentioned above. Hypothesis testing is done using the standard T-Test as well as the Student's probability test. The following null hypotheses are posed to cater for each relationship between the roles defined above as well as to globally represent the null hypothesis, H02.

When considering the outcomes of software development projects, there is no difference between the means of:

- developers and business analysts
- developers and project managers
- developers and architects
- business analysts and project managers
- business analysts and architects
- project managers and architects

In considering each role combination and the null hypotheses presented above, the following is found using the standard T-Test:

Table 20 Role P-Value

P-Value	A	B	C	D
A	-	0.0000	0.0211	0.0649
B	0.0000	-	0.0000	0.0000
C	0.0211	0.0000	-	0.0000
D	0.0649	0.0000	0.0000	-

Table legend: *A = Developer, B = Business Analyst, C = Project Manager, D = Architect*

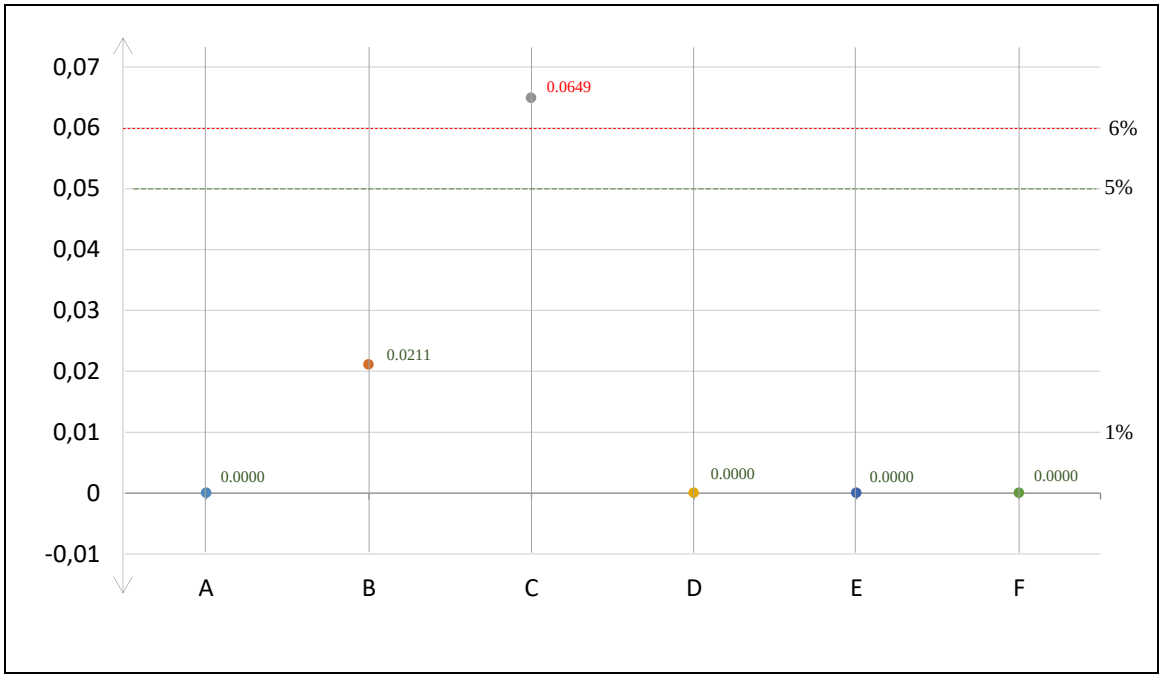


Figure 12: Role P-Value view

Graph legend: *A: Developer-Business Analyst, B: Developer-Project Manager, C: Developer-Architect, D: Business Analyst-Project Manager, E: Business Analyst-Architect, F: Project Manager-Architect*

Figure 12 (*Role P-Value view*) reveals that all relationships have a degree of difference between them and apart from the relationship of developer versus architect, each can be considered statistically significant.

Table 21 (*Role hypothesis testing outcomes*) provides a visual representation of the rejection of each null hypothesis presented in Table 20 (*Role P-Value*) above. Items marked in red (0) suggest that the null hypothesis is rejected and elements in green (1) suggests that the null hypothesis is not rejected.

Table 21 Role hypothesis testing outcomes

	Developer	Business Analyst	Project Manager	Architect
$H02_a$	0	0		
$H02_b$	0		0	
$H02_c$	1			1
$H02_d$		0	0	
$H02_e$		0		0
$H02_f$			0	0

Covariance

The role dimension proposes six different relationships, all related to a comparison of each role against others. In considering the covariance of each relationship, some evidence is found suggesting a correlation between roles, such as developer versus project manager. However, the strength of these are considered very weak. Figure 13 (*Role R-Value view*) provides a graphical representation of each relationship in terms of a positive (values greater than 0) or negative (values less than 0) linear relationship. The following outcomes are found:

- Relationship strength between Developer and Business Analyst = 0.0568. With an R-Value of 0.06, no linear relationship is established.
- Relationship strength between Developer and Project Manager = 0.1078. With an R-Value of 0.1, a marginally weak positive linear relationship can be found below 0.3.
- Relationship strength between Developer and Architect = 0.0111. With an R-Value of 0.01, no linear relationship is established.
- Relationship strength between Business Analyst and Project Manager = 0.0036. With an R-Value of 0.00, no linear relationship is established.
- Relationship strength between Business Analyst and Architect = -0.0509. With an R-Value of -0.05, no linear relationship is established.
- Relationship strength between Project Manager and Architect = 0.0165. With an R-Value of 0.01, no linear relationship is established.

It should be noted that a low correlational strength, such as $r = 0.3$ can still be considered statistically significant at $p < 0.01$.

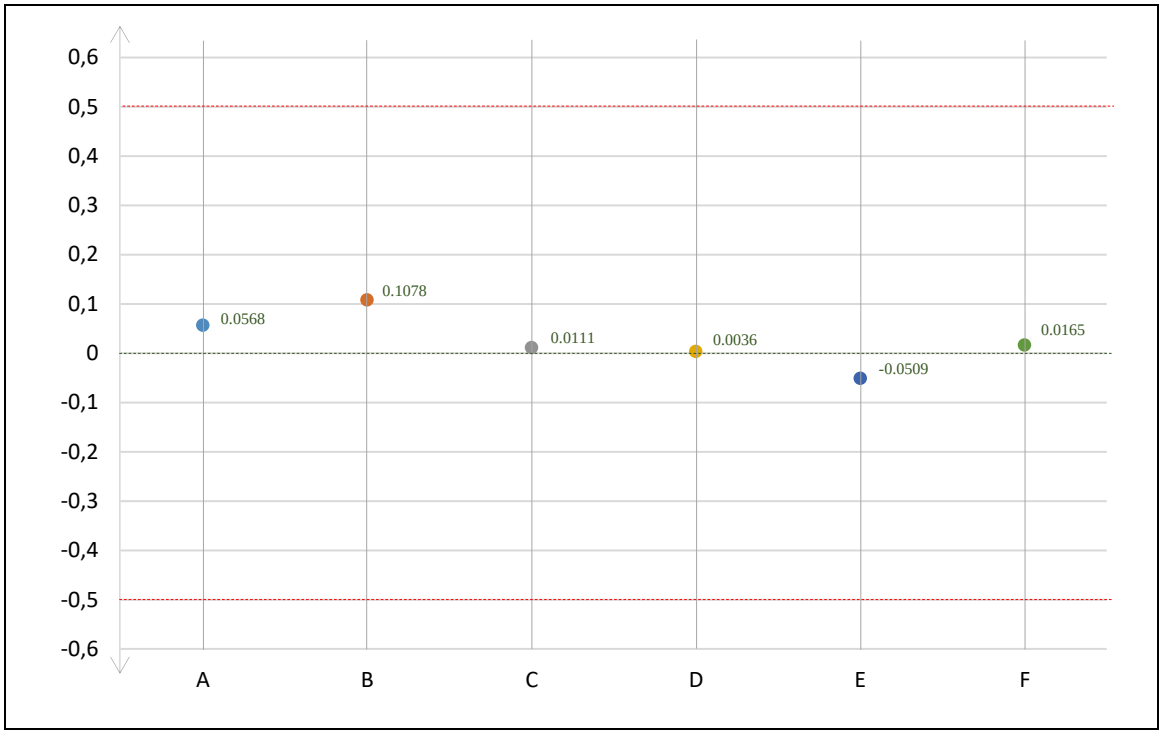


Figure 13: Role R-Value view

Graph legend: *A: Developer-Business Analyst, B: Developer-Project Manager, C: Developer-Architect, D: Business Analyst-Project Manager, E: Business Analyst-Architect, F: Project Manager-Architect*

Generation – View 27

It is suggested that experts in different generations reason differently when factors that influence software development outcomes are considered. When age is considered, it is found to be too granular to explain group dimensions. As much research has been dedicated to the study of differences between different generations, this research aims to introduce generation as a group perspective but defines generations in terms of relevant age groups. As such, generation as an independent variable is limited to the following:

- Age 18 – 22: Generation Z
- Age 23 – 30: Early-Mid Generation Y
- Age 31 – 40: Late Generation Y - Early Generation X
- Age 41 – 50: Mid - Late Generation X
- Age 51 – 65: Late Generation X - Early Boomers

The following research sub-question (RQ3) is posed to describe this phenomenon:

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts from different generations?*

The following null-hypotheses are posed to allow for the testing of this question:

- H03: There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts from different generations.
- H3: Experts in different generations in a software development team have differing views of "internal variability" and "political impacts" as the factors that influence bespoke software development outcomes in South Africa.

In considering generation as a secondary perspective for this research, a dataset is extracted that consolidates all research data into a single dataset that represents a combination of failed and successful software development projects as well as "internal variability" and "political impacts" as factor categorisations. This combination represents an expert's perspective on software outcomes, in terms of factors that influence software development. The independent variable of generation is used as a lens to illustrate the relationship of experts in different age groups as a secondary viewpoint.

Dependence

When considering the means of each generation, it can be argued that a difference in perspectives existed between each generation. Figure 14 (*Generation mean comparison*) below illustrates that the difference in means for each generation range from between 0.12% to 3.97%.

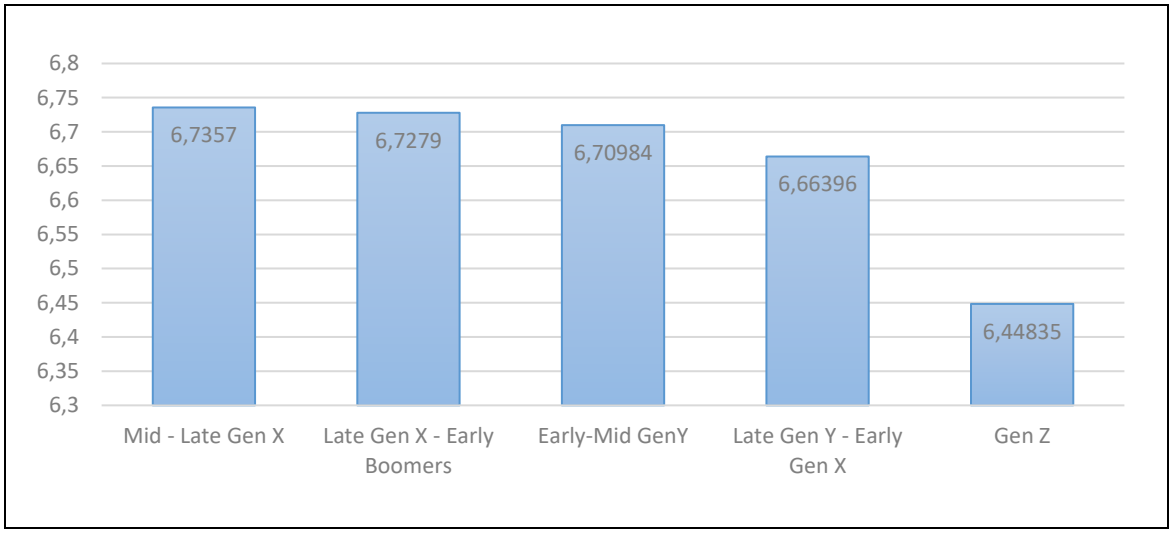


Figure 14: Generation mean comparison

As such, the following claims are made based on the visible differences of the means between each generation.

- When considering Generation Z and Early-Mid Generation Y, there appears to be a difference of 3.97% between perspectives
- When considering Generation Z and Late Generation Y - Early Generation X, there seems to be a difference of 3.29% between perspectives
- When considering Generation Z and Mid - Late Generation X, there seems to be a difference of 4.36% between perspectives
- When considering Generation Z and Late Generation X - Early Boomers, there seems to be a difference of 4.24% between perspectives
- When considering Early-Mid Generation Y and Late Generation Y - Early Generation X, there seems to be a difference of 0.69% between perspectives
- When considering Early-Mid Generation Y and Mid - Late Generation X, there seems to be a difference of 0.38% between perspectives
- When considering Early-Mid Generation Y and Late Generation X - Early Boomers, there seems to be a difference of 0.27% between perspectives
- When considering Late Generation Y - Early Generation X and Mid - Late Generation X, there seems to be a difference of 1.07% between perspectives

- When considering Late Generation Y - Early Generation X and Late Generation X - Early Boomers, there seems to be a difference of 0.95% between perspectives
- When considering Mid - Late Generation X and Late Generation X - Early Boomers, there seems to be a difference of 0.12% between perspectives

From a pragmatic viewpoint, these claims shed some light in terms of what the data is saying, but without statistical examination, little relevance can be proven. As such hypothesis testing is included to assist in describing the differences between the generations suggested above. Hypothesis testing is done using the standard T-Test as well as the Student's probability test. The following null hypotheses are posed to cater for each relationship between the generations defined above as well as to globally represent the null hypothesis, H03.

When considering the outcomes of software development projects, there is no difference between the means of:

- generation z (18 - 22) and early-mid generation y (23 - 30).
- generation z (18 - 22) and late generation y - early generation x (31 - 40).
- generation z (18 - 22) and mid - late generation x (41 - 50).
- generation z (18 - 22) and late generation x - early boomers (51 - 65).
- early-mid generation y (23 - 30) and late generation y - early generation x (31 - 40).
- early-mid generation y (23 - 30) and mid - late generation x (41 - 50).
- early-mid generation y (23 - 30) and late generation x - early boomers (51 - 65).
- late generation y - early generation x (31 - 40) and mid - late generation x (41 - 50).
- late generation y - early generation x (31 - 40) and late generation x - early boomers (51 - 65).
- mid - late generation x (41 - 50) and late generation x - early boomers (51 - 65).

In considering each generation combination and the null hypotheses presented above, the following is found using the standard T-Test:

Table 22 Generation P-Value

P-Value	A	B	C	D	E
A	-	0.0011	0.0003	0.0000	0.0000
B	0.0011	-	0.3288	0.3928	0.4172
C	0.0003	0.3288	-	0.1703	0.1592
D	0.0000	0.3928	0.1703	-	0.4370
E	0.0000	0.4172	0.1592	0.4370	-

Table legend: *A = Generation Z (18 - 22), B = Early-Mid Generation Y (23 - 30), C = Late Generation Y - Early Generation X (31 - 40), D = Mid - Late Generation X (41 - 50), E = Late Generation X - Early Boomers (51 - 65)*

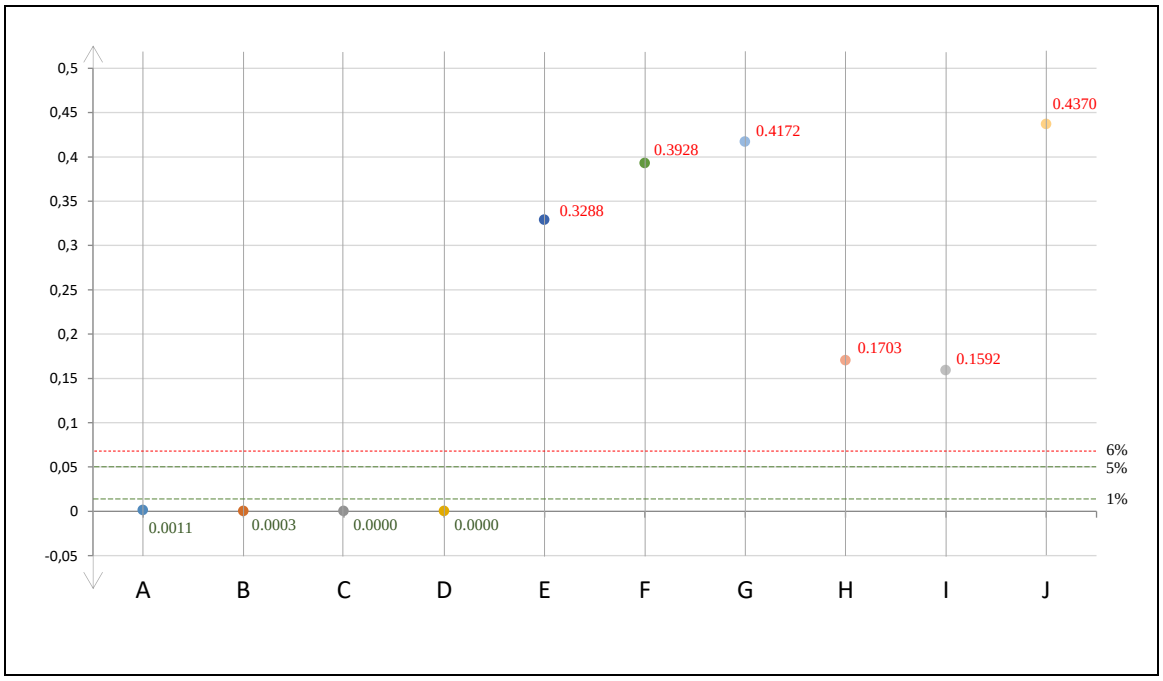


Figure 15: Generation P-Value view

Graph legend: *A = Generation Z -Early-Mid Generation Y , B = Generation Z -Late Generation Y - Early Generation X, C = Generation Z -Mid - Late Generation X , D = Generation Z -Late Generation X - Early Boomers , E = Early-Mid Generation Y -Late Generation Y - Early Generation X , F = Early-Mid Generation Y -Mid - Late Generation X , G = Early-Mid Generation Y -Late Generation X - Early Boomers , H = Late Generation Y - Early Generation X -Mid - Late Generation X , I = Late Generation Y - Early Generation X -Late Generation X - Early Boomers , J = Mid - Late Generation X -Late Generation X - Early Boomers*

Figure 15 (*Generation P-Value view*) reveals that all relationships have a degree of difference between them and apart from the relationships between early-mid generation Y versus late generation Y and early generation X, early-mid generation Y versus mid to late generation X, early to mid-generation Y versus late generation X to early boomers, late generation Y and early generation X versus mid to late generation X, late generation Y

and early generation X versus late generation x to early boomers, mid to late generation X versus late generation X and early boomers, each could be considered statistically significant.

Table 23 (*Generation hypothesis testing outcomes*) provides a visual representation of the rejection of each null hypothesis presented in Table 22 (*Generation P-Value*) above. Items marked in red (0) suggest that the null hypothesis is rejected and elements in green (1) suggest that the null hypothesis is not rejected.

Table 23 Generation hypothesis testing outcomes

	Gen Z	Early-Mid Gen Y	Late Gen Y - Early Gen X	Mid - Late Gen X	Late Gen X - Early Boomers
$H03_a$	0	0			
$H03_b$	0		0		
$H03_c$	0			0	
$H03_d$	0				0
$H03_e$		1	1		
$H03_f$		1		1	
$H03_g$		1			1
$H03_h$			1	1	
$H03_i$			1		1
$H03_j$				1	1

Covariance

The generation dimension proposes ten different relationships, all related to a combination of generation label against others. In considering the covariance of each relationship, some evidence is found suggesting a correlation between generations, such as early-mid generation Y versus late generation Y - early generation X. However, the strength of these are considered very weak. Figure 16 (*Generation R-Value view*) provides a graphical representation of each relationship in terms of a positive (values greater than 0) or negative (values less than 0) linear relationship. The following outcomes are found:

- Relationship strength between Generation Z versus Early-Mid Generation Y = -0.0863. With an R-Value of -0.08, no linear relationship is established.

- Relationship strength between Generation Z versus Late Generation Y - Early Generation X = -0.0053. With an R-Value of -0.00, no linear relationship is established.
- Relationship strength between Generation Z versus Mid - Late Generation X = -0.0782. With an R-Value of -0.08, no linear relationship is established
- Relationship strength between Generation Z versus Late Generation X - Early Boomers = -0.0157. With an R-Value of -0.02, no linear relationship is established.
- Relationship strength between Early-Mid Generation Y versus Late Generation Y - Early Generation X = 0.1613. With an R-Value of 0.16, a marginally weak positive linear relationship can be found below 0.3.
- Relationship strength between Early-Mid Generation Y versus Mid - Late Generation X = -0.0149. With an R-Value of -0.01, no linear relationship is established.
- Relationship strength between Early-Mid Generation Y versus Late Generation X - Early Boomers = 0.0315. With an R-Value of 0.03, no linear relationship is established.
- Relationship strength between Late Generation Y - Early Generation X versus Mid - Late Generation X = 0.0244. With an R-Value of 0.02, no linear relationship is established.
- Relationship strength between Late Generation Y - Early Generation X versus Late Generation X – Early Boomers = -0.1401. With an R-Value of -0.14, no linear relationship is established.
- Relationship strength between Mid - Late Generation X versus Late Generation X - Early Boomers = 0.0021. With an R-Value of 0.00, no linear relationship is established.

It should be noted that a low correlational strength, such as $r = 0.3$ can still be considered statistically significant at $p < 0.01$.

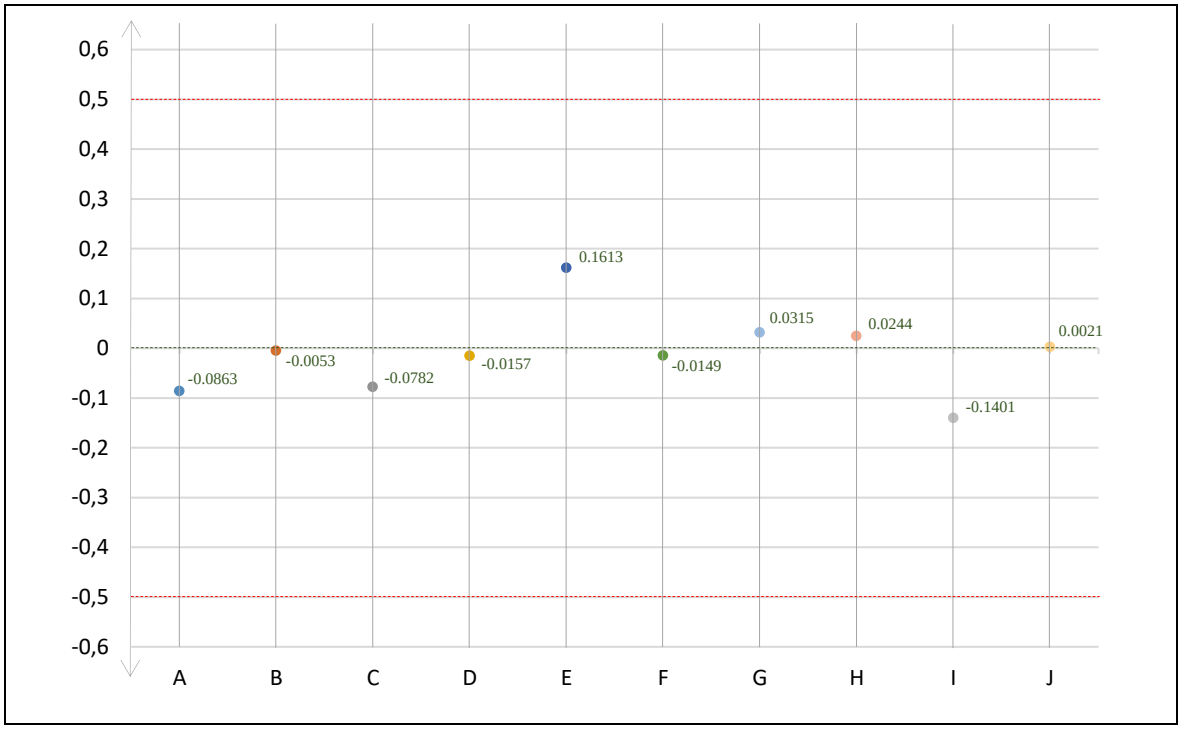


Figure 16: Generation R-Value view

Graph legend: *A = Generation Z -Early-Mid Generation Y , B = Generation Z -Late Generation Y - Early Generation X, C = Generation Z -Mid - Late Generation X , D = Generation Z -Late Generation X - Early Boomers , E = Early-Mid Generation Y -Late Generation Y - Early Generation X , F = Early-Mid Generation Y -Mid - Late Generation X , G = Early-Mid Generation Y -Late Generation X - Early Boomers , H = Late Generation Y - Early Generation X -Mid - Late Generation X , I = Late Generation Y - Early Generation X -Late Generation X - Early Boomers , J = Mid - Late Generation X -Late Generation X - Early Boomers*

Tenure – View 26

It is suggested that experts with different tenure reason differently when factors that influence software development outcomes are considered. When considering tenure as a group dimension, a clear distinction can be made between each level of experience. In terms of this study, level of experience is seen as disconnected from generation as experience is deemed to be in the role individuals execute within. In software engineering, it is found to be plausible that experts move between functions and even though they may be deemed to form part of one of the more mature generations, their tenure, or level of experience in their current role may be limited. As such, tenure as an independent variable is limited to the following:

- Less than 1 year: Learner level

- At least 1 year but less than 2 years: Junior level
- At least 2 years but less than 7 years: Junior to intermediate level
- At least 7 years but less than 12 years: Intermediate to Senior level
- 12 years or more: Senior level

The following research sub-question (RQ4) is posed to describe this phenomenon:

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts of different tenure?*

The following hypotheses are posed:

- H04: There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts of different tenure.
- H4: Experts with different tenure in a software development team have differing views of “internal variability” and “political impacts” as the factors that influence bespoke software development outcomes in South Africa.

In considering tenure as a secondary perspective for this research, a dataset is extracted that consolidates all research data into a single dataset that represents a combination of failed and successful software development projects as well as “internal variability” and “political impacts” as factor categorisations. This combination represents an expert's perspective on software outcomes, in terms of factors that influence software development. The independent variable of tenure is used as a lens to illustrate the relationship of experts with different levels of experience as a secondary viewpoint.

Dependence

When considering the means of each tenure, it can be argued that a difference in perspectives exists between each tenure. Figure 17 (*Tenure mean comparison*) illustrates that the means for each tenure differ from between 0.15% and 7.67%.

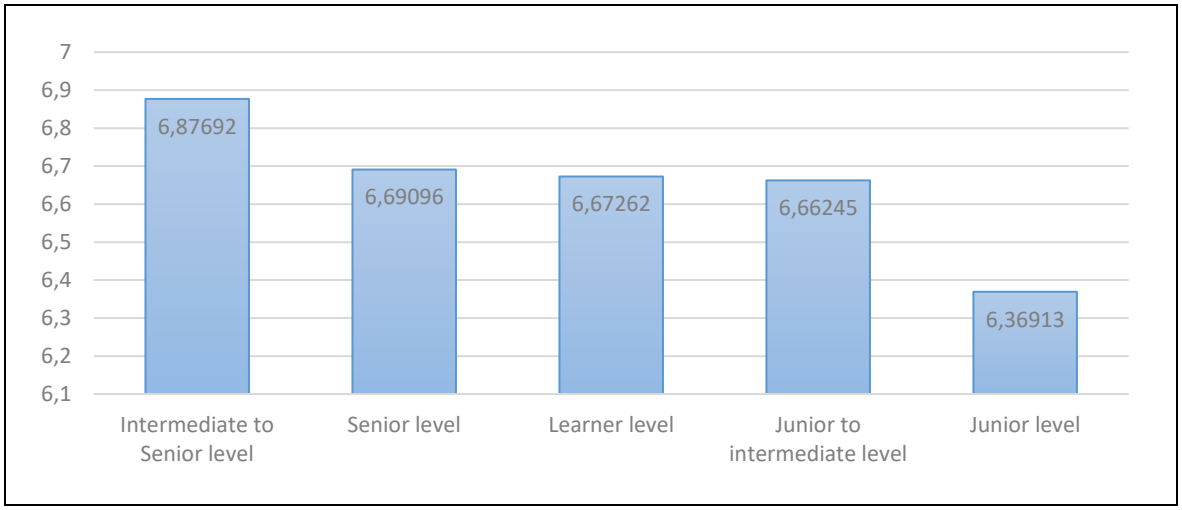


Figure 17: Tenure mean comparison

As such, the following claims are made based on the visible differences of means between each tenure.

- When considering Learner level - Junior level, there appears to be a difference of 4.65% between perspectives
- When considering Learner level - Junior to intermediate level, there seems to be a difference of 0.15% between perspectives
- When considering Learner level - Intermediate to Senior level, there seems to be a difference of 3.02% between perspectives
- When considering Learner level - Senior level, there appears to be a difference of 0.27% between perspectives
- When considering Junior Level - Junior to intermediate level, there seems to be a difference of 4.50% between perspectives
- When considering Junior level - Intermediate to Senior level, there seems to be a difference of 7.67% between perspectives
- When considering Junior level - Senior level, there seems to be a difference of 4.93% between perspectives
- When considering Junior to intermediate level - Intermediate to Senior level, there seems to be a difference of 3.17% between perspectives

- When considering Junior to intermediate level - Senior level, there seems to be a difference of 0.43% between perspectives
- When considering Intermediate to Senior level - Senior level, there seems to be a difference of 2.74% between perspectives

From a pragmatic perspective, these claims shed some light in terms of what the data is saying, but without statistical examination, little relevance can be proven. As such hypothesis testing is included to assist in describing the differences found between the tenure group. Hypothesis testing is done using the standard T-Test algorithm and Student's probability test. To cater for each tenure relationship, the following null hypotheses are posed, all cumulatively represented by null hypothesis 4, H04.

When considering the outcomes of software development projects, there is no difference between the means of:

- Learner level and junior level.
- Learner level and junior to intermediate level.
- Learner level and intermediate to senior level.
- Learner level and senior level.
- Junior level and junior to intermediate level.
  Junior level and intermediate to senior level.
  Junior level and senior level.
- Junior to intermediate level and intermediate to senior level.
- Junior to intermediate level and senior level.
- Intermediate to senior level and senior level.

In considering each tenure combination and the null hypotheses presented above, the following is found using the standard T-Test:

Table 24 Tenure P-Value

P-Value	A	B	C	D	E
A	-	0.0000	0.4571	0.0005	0.3997
B	0.0000	-	0.0018	0.0000	0.0000
C	0.4571	0.0018	-	0.0207	0.3992
D	0.0005	0.0000	0.0207	-	0.0152
E	0.3997	0.0000	0.3992	0.0152	-

Table legend: *A = Learner level, B = Junior level, C = Junior to intermediate level, D = Intermediate to Senior level, E = Senior level*

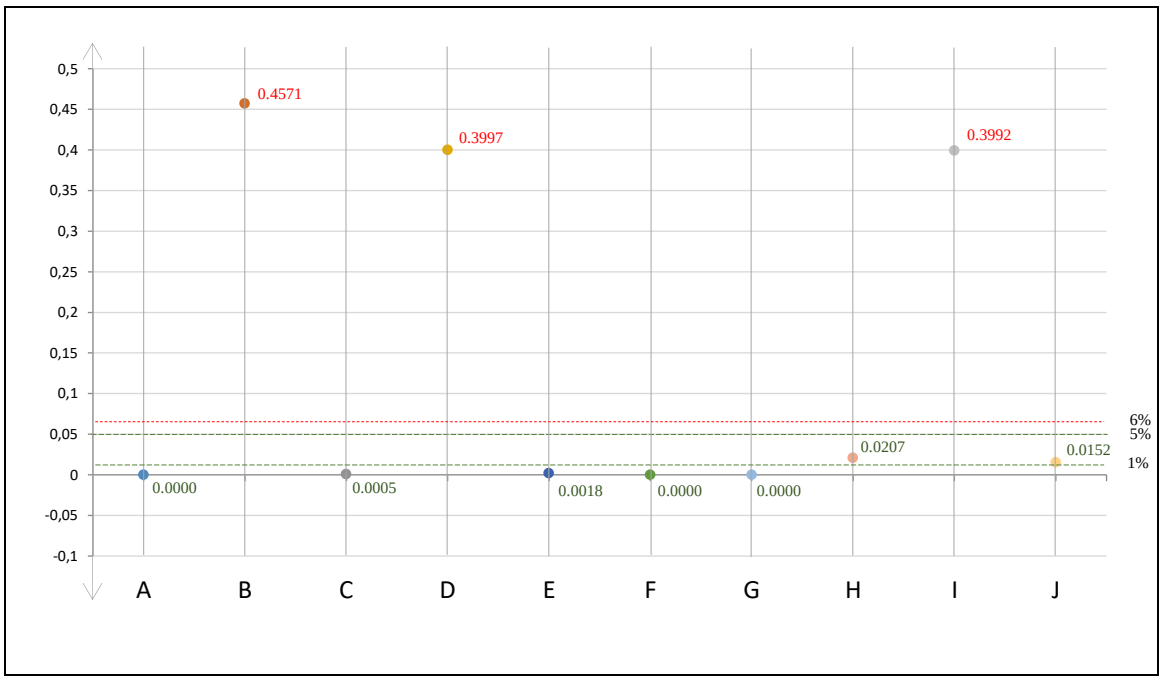


Figure 18: Tenure P-Value view

Graph legend: *A= Learner level - Junior level, B = Learner level - Junior to intermediate level, C = Learner level - Intermediate to Senior level, D = Learner level - Senior level, E = Junior Level - Junior to intermediate level, F = Junior level - Intermediate to Senior level, G = Junior level - Senior level, H = Junior to intermediate level - Intermediate to Senior level, I = Junior to intermediate level - Senior level, J = Intermediate to Senior level - Senior level*

Figure 18 (*Tenure P-Value view*) reveals that all relationships have a degree of difference between them and apart from the relationships between learner level and junior to intermediate level, learner level versus senior level and junior to intermediate level versus senior level, each can be considered statistically significant.

Table 25 (*Tenure hypothesis testing outcomes*) provides a visual representation of the rejection of each null hypothesis presented in Table 24 (*Tenure P-Value*) above. Items marked in red (0) suggests that the null hypothesis is rejected and elements in green (1) suggests that the null hypothesis is not rejected.

Table 25 Tenure hypothesis testing outcomes

	Learner level	Junior level	Junior to Int level	Int to Senior level	Senior level
$H04_a$	0	0			
$H04_b$	1		1		
$H04_c$	0			0	
$H04_d$	1				1
$H04_e$		0	0		
$H04_f$		0		0	
$H04_g$		0			0
$H04_h$			0	0	
$H04_i$			1		1
$H04_j$					

Covariance

The tenure dimension proposes ten different relationships, all related to a combination of tenure against others. In considering the covariance of each relationship, some evidence is found suggesting a correlation between generations, such as learner level versus a junior level. However, the strength of these are considered very weak. Figure 19 (*Tenure R-Value view*) provides a graphical representation of each relationship in terms of a positive (values greater than 0) or negative (values less than 0) linear relationship. The following outcomes are found:

- Relationship strength between Learner level - Junior level = 0.1484. With an R-Value of 0.15, a marginally weak positive linear relationship can be found below 0.3.
- Relationship strength between Learner level - Junior to intermediate level = 0.0146. With an R-Value of 0.01, no linear relationship is established.
- Relationship strength between Learner level - Intermediate to Senior level = 0.0420. With an R-Value of 0.04, no linear relationship is established.

- Relationship strength between Learner level - Senior level = -0.0389. With an R-Value of -0.04, no linear relationship is established.
- Relationship strength between Junior Level - Junior to intermediate level = 0.1178. With an R-Value of 0.12, a marginally weak positive linear relationship can be found below 0.3.
- Relationship strength between Junior level - Intermediate to Senior level = 0.0385. With an R-Value of 0.04, no linear relationship is established.
- Relationship strength between Junior level - Senior level = 0.0353. With an R-Value of 0.04, no linear relationship is established.
- Relationship strength between Junior to intermediate level - Intermediate to Senior level = 0.1426. With an R-Value of 0.14, a marginally weak positive linear relationship can be found below 0.3.
- Relationship strength between Junior to intermediate level - Senior level = 0.1058. With an R-Value of 0.11, a marginally weak positive linear relationship can be found below 0.3.
- Relationship strength between Intermediate to Senior level - Senior level = 0.0771. With an R-Value of 0.08, no linear relationship is established.

It should be noted that a low correlational strength, such as $r = 0.3$ can still be considered statistically significant at $p < 0.01$.

Graph legend: *A= Learner level - Junior level, B = Learner level - Junior to intermediate level, C = Learner level - Intermediate to Senior level, D = Learner level - Senior level, E = Junior Level - Junior to intermediate level, F = Junior level - Intermediate to Senior level, G = Junior level - Senior level, H = Junior to intermediate level - Intermediate to Senior level, I = Junior to intermediate level - Senior level, J = Intermediate to Senior level - Senior level*

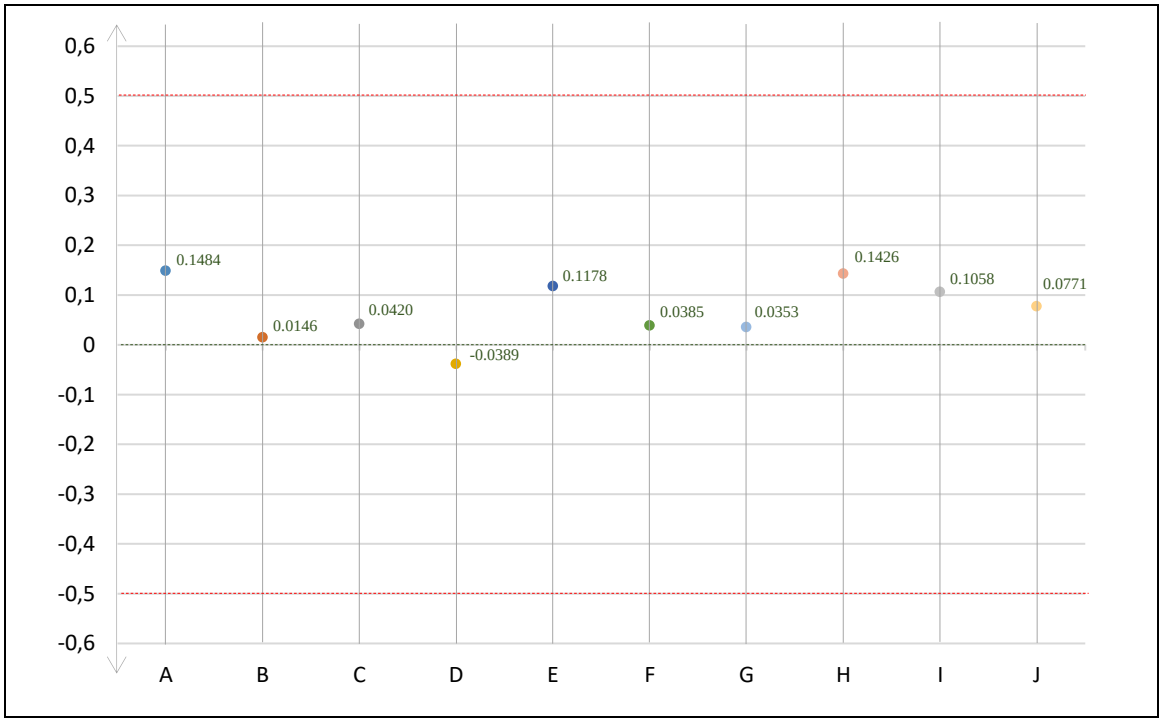


Figure 19: Tenure R-Value view

This section provides a summary of the data analysis process and discusses what was found during analysis. As such, this section is considered appropriate for all reader types.

5.5 Summary

The data analysis phase incorporates several statistical analysis techniques used to uncover whether a difference exists between experts with different role, tenure and generation labels. The analysis incorporates T-Testing and proposes several sub-hypotheses aimed specifically at testing the relationship between categories present in each variable for statistical significance. In considering each of these sub-categories in a consolidated view, the following information is produced:

In terms of a summary perspective and the accumulation of all hypothesis presented for each internal variable, the following is found:

- When considering role, four out of five hypotheses are rejected, resulting in a total of 80% rejection rate of all associated null hypotheses.
- When considering generation, four out of ten hypotheses are rejected, resulting in a total of 40% rejection rate of all associated null hypotheses.
- When considering tenure, seven out of ten hypotheses are rejected, leading to a total of 70% rejection rate of all associated null hypotheses.

Table 26 Consolidated hypotheses outcomes

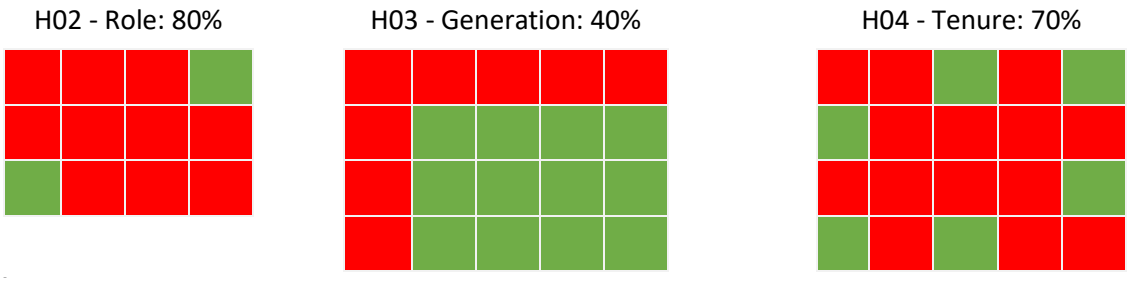


Figure 20 (*Role, Generation and Tenure hypothesis test*) below provides a graphical representation of null hypotheses that are not rejected in terms of role (H02), generation (H03) and tenure (H04).

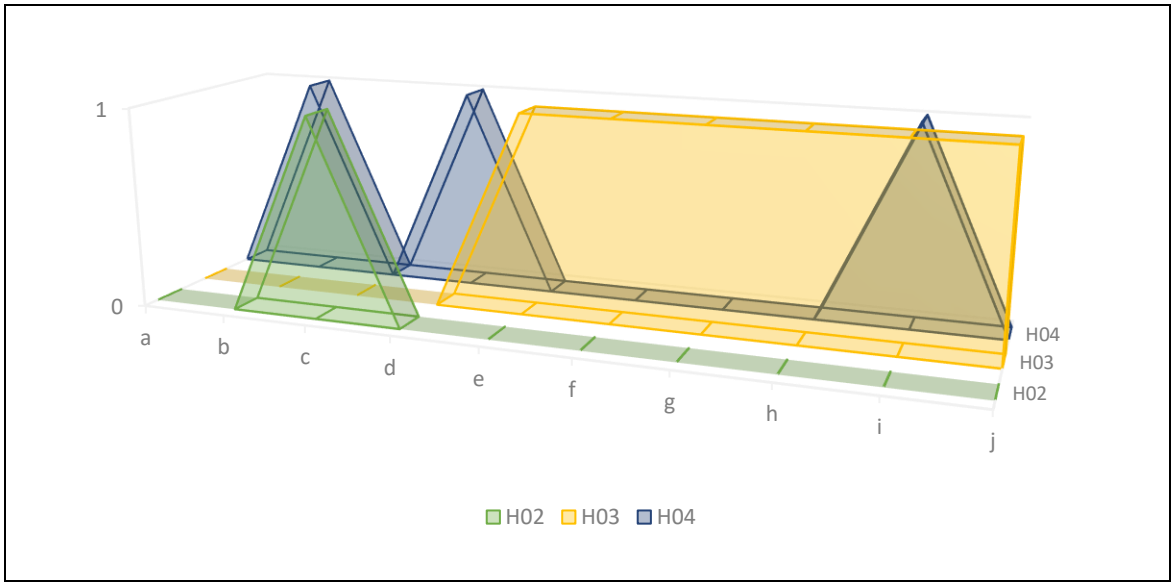


Figure 20: Role, Generation & Tenure hypothesis test

In addition, the analysis phase incorporates Pearson's correlational tests to verify the strength of relationships, should they exist. It is found that when considering all variables, no linear relationships are present. As such, the following three main conclusions are presented as a summary of the data analysis phase.

When considering perspectives about the software development project outcome provided by experts in different software development roles; and when these roles can be classified in terms of development, business analysis, project management or architecture, it is expected that a difference would exist between experts in different roles. It is found that when perspectives of most of these roles are compared to others, a significant statistical difference is recorded between each. However, in terms of the perspectives of developers against those of architects, the difference is not significant.

Similarly, when considering perspectives about the software development project outcome provided by software development experts with different tenure, it is expected that a difference would exist between experts with varying levels of experience. It is found that when the perspectives of most of these tenure groups are compared to others, a significant difference is recorded. However, in terms of the perspectives between learner level and junior to intermediate level, learner level and senior level as well as junior to intermediate level and senior level, the difference is not significant.

Lastly, when considering perspectives about the software development project outcome provided by software development experts of different generations, it is expected that a difference would exist between experts from various ages. It is found that when the perspectives of most of these generation groups are compared to others, no significant difference is recorded between each, but in terms of the perspectives of Generation Z against each of the other generation groups, the difference is significant.

6 Discussion

This section describes the evolution of the term software crisis and how it related to modern software development. Even though it may be considered interesting to all reader types, it is believed that pragmatic readers may find this section most valuable.





6.1 Evolution

Chapter 5 explains the data analysis at the highest level and as such a level of repetition of at the highest level within Annexure I is expected. This section has been reviewed and where the level of detail was inappropriate, was removed or moved into Annexure I. This approach is based on the lack of guarantee that the Annexure would be read in conjunction with the report and as such, a level of detail is required to explain the outcome and argument made in the paper

6.1.1 Software crisis

It appears that a divide in the international software engineering community exists, one sceptical of positive progress in software engineering; the other credulous about its successes. Arguments are made that suggest that the software engineering industry remains in crisis, and the software crisis of 1969 survived the test of time (Boehm, 2006; Crockford, 2007). In unpacking the history of the software engineering industry, it cannot be argued that the software crisis did not exist, nor can an argument be made against its perpetuity. In its beliefs and opinions about the existence of the software crisis, the software population remained divided over the years. One camp guided by a positivist approach, the other by interpretivism; sketching a bleak picture of impacts on humanity that could almost be considered equal to a looming software apocalypse. The software crisis rhetoric as termed by Ensmenger (Ensmenger, 2010), is not founded on theory and evidence has highlighted only one instance of “crisis” in the history of the industry. Why would software crisis, or the continued use of the term be as popular as the topic has been over the years?

This dissertation suggests that the evolution of the industry is somewhat of a 'maturing software crisis'. This phrase can be considered an oxymoron, but the research reveals that even though the software industry has had problems, they have mostly been referred to as crises, thus turning software crisis into a somewhat generic term. It is no longer a term that references a specific incident but instead became an abstract idea that represents problems in software. Its generic use can be considered equal to how Google became an abstract concept of searching the internet. In terms of patent and trademark law, 'genericise'[14] has been responsible for transforming names of corporations and products alike into everyday speech, such as Hoover describing the act of vacuuming a floor, Kreepy Krauly describing all types of pool cleaners and Windsurfer describing any board-sailing equipment. Sikwane (2013), in reporting on the risks Twitter raised against losing value in their trademark noted that "a trademark is used extensively, but instead of becoming associated with one particular company it becomes the generic term for the product" (Sikwane, 2013). The term software crisis became exactly that. It is no longer a reference to a single incident, such as the insufficient number of skilled software engineers of the late 1960's, but its continued use over the years has made it an abstract term used to describe challenges in the software industry. As such,

> *software crisis is understood as the reference to its original incarnation and 'software crisis' as a general term or abstract concept associated to software development failure.*

6.1.2 History and the need to succeed

The world-wide-web is littered with self-help literature, ranging from '100 simple secrets to happy people' to simple concepts such as 'success is a choice', suggesting that humans need to succeed, particularly in situations where failure thrives. The phenomenon is evident in reconstructing the history of software method and practice evolution in terms of Boehm's (2006) interpretation of Hegel's dialectic of increased human understanding

[14] The generic behaviour of something

(Boehm, 2006). In constructing this view, it is revealed that software methodology evolution saw several paradigm-shifts over the years as methods moved from thesis to antithesis, antithesis to synthesis and synthesis to antithesis, and so on. It is also evident that during this period, several concepts were developed that were considered revolutionary to the industry at that time. Most fascinating is that each paradigm shift appears to have been initiated by some crisis, for instance, formal structured design and the waterfall methodology coincide with the original software crisis, evolutionary models such as the spiral and rapid application development methods with the 'time to market crisis' and Agile methods with the 'dot-com crisis', to name a few. It is revealed that each paradigm shift is focused on a specific theme as the industry developed, such as engineering in the early 1950's, software craftsmanship in the early 1960's, formality in the early 1970's, productivity and then concurrent engineering in the mid to late 1980's, software criticality in the late 1990's to early 2000's, followed by value based engineering and globalization towards the 2010. Based on this trend, it can be reasoned that the evolution of software engineering would continue as long as the industry remains fixated on the 'software crisis'.

6.1.3 Problem significance

Professor Simon Conway Morris (2012), a leading researcher in the evolution of humanity suggested that evolution is inevitable (Morris, 2012). The author draws from the work done by Galileo and noted that history (*and in return the present*) may have been different if certain key events had been different (Ibid.). When considering the evolution of software engineering, this theory holds true.

It can be argued that processes and practices are created to solve common problems in a consistent manner. In addition, should a specific process or practice no longer be appropriate, or the problem is no longer considered common, the process or practice will be changed to align to the new condition that underpins the problem.

When considering the history of software engineering, three distinct concepts can be highlighted, software development failure, 'software crisis' and an ever-changing software

engineering landscape. It is important to note that these concepts have one thing in common, people.

It is not argued that software development initiatives have failed over the years. Similarly, it is noted that even though software development failure has adopted ‘software crisis’ as a common term to describe the state of software engineering, an argument can be made that many software development projects have succeeded. In the fog of confusion fuelled by an everlasting ‘software crisis’, the software engineering landscape evolved.

Even though the necessity of evolution can be reasoned, in terms of this research, it is suggested that a major contributor to the evolution of software engineering is how experts in this realm experience and understand failure. As such, it is argued that

> *expert opinion has a direct influence on the evolution of software engineering.*

This section describes cognitive theory as it relates to the expert judgement and the factors that influence software development outcomes. This section covers both theoretical concepts as well as insight into the relation of perceptions and reality and as such is considered valuable to both academic and pragmatic readers.			

6.2 Cognitive theory

Cognitive theory suggests that the interpretations of abstract concepts are based on perception. Sandoval (2016), in describing abstract and concrete perception theory suggested that the perception of the world was created by both literal and implied senses. The author noted that literal perception was based on realistic and tangible forms as they appeared, and implied perception, where reality was founded based on a proxy, or meaning of the environment, rather than its reality. The author suggested that “were we without abstract perception, we would be incapable of seeing beyond the immediately observable – unable to foretell coming events or imagine alternatives to situations” (Sandoval, 2016).

What role does implied perception play in the evolution of software methodology? How did the ‘software crisis’ contribute towards creating problems in the industry? Sandoval

noted that "were we without concrete perception, our minds would never form a tactile and correct concept of reality and would be utterly lost in an abstract delusion where imagination is taken as literal reality" (Sandoval, 2016). This dissertation suggests that without collecting and using empirical evidence, experts are forced to rely on their perceptions to explain outcomes of software development initiatives. In terms of Sandoval's (2016) explanation, it can be argued that an ill-conceived focus on the 'software crisis' has created a misguided view of reality, suggesting that reasoning in terms of software practice evolution may be flawed. Also, as experts contribute in several different roles, generations and tenure, this research argues that their social class biases their reasoning. Taherdoost and Keshavarzsaleh (2015) in reviewing software failure and successes noted that "important factors differ across projects; and that the approach fails to account for the dynamics of social, organizational [*sic*], and political life that surround any IS project should be considered significantly" (Taherdoost and Keshavarzsaleh, 2015a). Several authors noted the importance of social factors in software development, but no research can be found that tests the impact these have on the outcomes of software development initiatives, and in turn, the evolution of software methodologies. Thus, this study aims to understand whether experts in different social classes argue differently about the outcomes of software development initiatives.

6.2.1 Problem significance

It can be reasoned that the 'software crisis' is a contributor towards the evolution of the software engineering industry. In addition, this research argues that *expert opinion has a direct influence on the evolution of software engineering*. Sandoval's work (Sandoval, 2016) supports this reasoning and provides a theory that clarifies how experts 'predict' future situations. For example:

Cloud-based software development is a paradigm that enjoys the focus of the industry. Microservice computing is enabled by all Cloud service-providers and development tools and libraries now have some capabilities that will allow a developer to build and deploy microservices to any cloud-based environment. This realm is not without its challenges, as many more moving parts in a solution increases the complexity and reduces its overall

maintainability (*expert opinion*). Even though microservice computing remains a hot topic, the murmur of failure with microservices is becoming louder. Based on Sandoval's view (Sandoval, 2016), the combination of 'software crisis' (*abstract perception*) and microservice failure (*concrete perception*) would create a world-view that action should be taken, suggesting that the industry is gearing up for another paradigm shift. Thus,

> *it is predicted that based on the established trend and a relentless focus on microsystem cloud-computing, the industry would arrive at a synthesis of idealistic and pragmatic paradigms, focused on micro-measurement within the next three years.*

This section consolidates software crisis, cognitive theory and extends these concepts in terms of how they contribute towards the perceived chaos in the software engineering industry. The discussion is considered most valuable to pragmatic readers.			

6.3 Cognitive bias

Cognitive bias is defined as a systematic deviation from optimal judgment (Ralph, 2013). Dvorsky (2013) noted twelve cognitive biases responsible for preventing rational behaviour and explained negativity bias as a phenomenon where humans often pay more attention to bad news (Dvorsky, 2013). The author pointed out that this attraction to negativity was not due to a morbid disposition, but rather that negative news was perceived to be more significant or profound and as such, enjoyed more credibility. The author wrote that "heeding bad news may be more adaptive than ignoring good news" and "we run the risk of dwelling on negativity at the expense of genuinely good news" (Ibid). Rozin and Royzman (2001) suggested that a general bias existed that was based on inborn predispositions as well as experience to value negative entities higher. The authors wrote that "1 [*sic*] feature of negative events that make them dominant is that negative entities are more contagious than positive entities" (Rozin and Royzman, 2001). Ralph (2013) noted that "some biases interact and reinforce each other, forming complexes of biases, or biasplexes [*sic*]" which "in turn, lead to behavioral [*sic*] antipatterns, which may deleteriously affect software engineering projects" (Ralph, 2013). It can be argued that 'software crisis' remains relevant over the years due to negativity bias. A focus on

negativity has an ability to cultivate a drive to continuously ‘fix’ problems. Similarly, it can be argued that ‘software crisis’ as an abstract phenomenon creates negativity, but from a concrete perception would not have been capable of providing enough insight into what needed to be fixed. In terms of creating a reality that combined both abstract and concrete perceptions, the industry must ask the hard question of what the physical reasons for ‘software crisis’ are, but due to a lack of, or access to empirical evidence is forced to create empiricism by leveraging human perspective. Herein exists several inherent problems.

In 1977, Leo M. Cherne wrote that “the computer is incredibly fast, accurate, and stupid. Man is unbelievably slow, inaccurate, and brilliant. The marriage of the two is a force beyond calculation” (Garland, 1982). Individuals are complex beings, and even though their reasoning can be considered individually brilliant, it cannot be considered cumulatively accurate. It is argued that

> *individual brilliance is formed by combining own experiences and understanding and ultimately settling on a plausible individual reality. Group understanding is created by discussing individual perspectives and the addition of the understanding and expertise of others; own understanding may be augmented creating a revised individual brilliance. However, when personal understanding cannot be augmented, or when there is a fundamental difference in base opinion, individuals may tend to settle on a perspective for the greater good of the group, creating a false reality that over time becomes individual experience. As such, cumulative perspectives may be made more inaccurate when the complexity of the situation increases.*

Authors Rosqvist, Koskela and Harju (2003) suggested that experts had their own “mental model” of what was believed to be their realities and wrote that “*the evaluation of software quality may be based on expert group decision-making rather than empirical evidence alone”* (Rosqvist, Koskela and Harju, 2003). Given strict instructions, computer technologies can accurately measure and report variance. Automation can greatly increase the accuracy of measurement if it can be defined in a simplistic form, but it can be argued

that measuring software outcomes are complex, and in most cases, would be neglected due to a lack of understanding of what would be deemed valuable to measure.

When considering the software industry and the evolution of software engineering, especially methods and practices, the severity of the problem is made evident.

6.3.1 Problem significance

It is not argued that practices created over the years have not contributed towards solving the 'software crisis'. Nor is it claimed that a method considered an antithesis to another did not invigorate efforts to 'do better' in the industry. To the contrary, it is believed that each partial- or dissimilar approach contributed significantly to maturing the industry and the evolution is seen in terms of methodology would continue to do so into the future. The argument is that individual perspectives are based on the biases created by their social class and accuracy is reduced when a group perception is created. It is also argued that accuracy is reduced the more complex the situation. When considering the industry and the complexity of software engineering at the highest level, a common solution may only solve the problems of some of the population. Consider the following as an example:

> *The Agile movement is founded on the principles that consider pragmatism over formality and suggests that "we are uncovering better ways of developing software by doing it and helping others do it"* (Fowler and Highsmith, 2001). *In defining what quality is, the movement provided four values that suggested there was more value in focusing on the individual and the interactions they have, collaborating with customers, delivering working software as a proof of success, whilst being able to respond to change, than the processes, tools and execution plans used and followed by a team, the extent to which documentation was considered complete and finally the contracts that were negotiated with customers. Several frameworks have been developed since the movement was founded in 2001 embracing these values as core to their existence. However, in critically evaluating these values as well as the principles defined by the movement, it can be argued that the ability to adopt*

these, require a supposed maturity not just in terms of the individuals developing software, but also in terms of the environment these individuals execute within.

Agile frameworks presuppose that its adoptees have a predisposition to working and acting maturely and have an inherent expert capability. Additionally, Agile frameworks assume that from a management and control perspective, adoptees are enabled and entitled to make their own decisions and are trusted to not just make the right decision but to make the decisions right, suggesting an overarching maturity in terms of the organisation supporting Agile practitioners. Thus, Agile frameworks are considered biased towards a high level of maturity in terms of the individuals practising within this realm as well as the organisations supporting them in their efforts. Equally, Agile frameworks are biased in a belief that the most appropriately skilled individuals are those doing the work and not those managing the team. As such, Agile frameworks operate on the premise that a team is self-managing.

Now, consider a failing software development team working in an environment with an intrinsic culture of chaos management, defined, refined and formalised approaches to conceptualising and producing software with several formalised stage-gate approaches for controlling quality during execution. It can be argued that an environment such as the one explained has a maturity of its own in terms of the formality in which it operates. However, it is suggested that a software development team in such an environment would have a marginal chance of success should it adopt an Agile framework to solve their problems. A problem exists in that even though either approach could be argued as mature in its own right; the second is biased towards hands-on management control as it assumes that the management contains all the skill and are, as such the most appropriate when decision-making rights are concerned. Should there be an organisational drive aimed at adopting Agile as a way of working,

the culture may prove to be a factor for failure to adopt, or may slow down the adoption to a degree where it is difficult to demonstrate value.

This section articulates the argument the research is making by explaining the theoretical proposition developed as part of the research. This section is considered to add most value to pragmatic readers.





6.4 Chaos contributors

Is the software engineering industry in trouble? It can be argued that continuous change in an environment is a tell-tale sign of immaturity. Equally, it can be claimed that maturity could only be gained through experimentation. Is the evolution of software methods and practices based on experimentation, or is there relevant literal and implied senses that when combined with 'software crisis' fuel evolution to move towards a utopia filled with promises of success? Geethalakshmi and Shanmugam (2008) suggested that "failure and success provide different perspectives on improvement: failure tells what not to do in future, where as [*sic*] success shows what should be done again" (Geethalakshmi and Shanmugam, 2008).

Over time, some investment has been made into studies on software success. However, the meaning of success remains unclear, and its measurement appears to be divided across two primary perspectives. One group favours a pragmatic view, suggesting success is born out of a balance between time, cost and function; and the other, an idealistic view suggesting that the only measure of success is working software. In reviewing relevant literature on software failure, it is found that the knowledge base is littered with studies aimed at understanding software development failures. However, the number of studies focused on the management of software development initiatives outweighed those focused on any other area that underpins the actual act of developing software. In addition, most studies cannot factually prove that the reasons provided for failure are founded in measured project execution data but are rather collected from individuals involved in the initiative.

If understanding failure has an impact on attaining success, why would there be a lack of industry data that could be used to avoid failure? It can be argued that finding and implementing appropriate measures capable of collecting relevant information about

software development initiatives across the act of developing software has been difficult. It can also be argued that the failure in appropriately collecting relevant data on software project execution, teams have to rely on their experts to provide guidance, but as *success and failure mean different things to different people*, noise has diluted the accuracy of the outcomes. Thomas and Fernández (2008) noted that "the ascription of success and failure is a social accomplishment dependent on the perspective of the subject" (Thomas and Fernández, 2008). Finally, it can be argued that understanding failure is not in the best interest of those involved in the software development initiative. May (1998) suggested that "studies into software project development failures were neglected in an effort to reduce further losses, and in certain cases, where investigations were done, they resulted in hidden or manipulated data in order to protect careers and reputations of the parties involved." (May, 1998). Heeks and Bhatnagar (1999) stated that the prevalence of failure was rarely found, as failures are swept under the carpet, while successes were shouted from the rooftops (Heeks and Bhatnagar, 1999).

The study argues that factors responsible for influencing the outcomes of software development initiatives should be categorised from two perspectives. The team perspective is viewed as factors the team could control and as such are termed "internal variability". The overarching perspective is seen as factors the team had no control over. These include environmental, political and organisational factors and are termed political impacts. In dissecting the work by Fred Brooks (2002), a correlation is established between essential complexities and "internal variability" as well as "political impacts" and accidental complexities. Brooks (2002) noted that "complexity of software is an essential property, not an accidental one" and "when we examine the three steps in software technology that have been most fruitful in the past, we discover that each attacked a different major difficulty in building software, but they have been the accidental, not the essential, difficulties" (Brooks, 2002). Even though the Brooks (2002) essays are dated, they are widely regarded as relevant to modern software engineering as they provide an abstract reality of software practice.

6.4.1 Problem significance

This study argues that without formal data collection on the outcomes of software development initiatives, the industry can only rely on the perceptions of those who are involved in the initiative. In addition, as success and failure are experienced differently by all of the parties concerned (Thomas and Fernández, 2008), should the reasons for failure be used to evolve a development practice from its current form to one aimed at driving a different outcome, it may influence the perspective of a particular party if their views are not considered as input. For example:

> *Consider a software development team working in an Agile environment and following an Agile framework in its native form. The team's measurement of success is delivering business value in terms of a potentially shippable product increment. The team's performance has been hampered over several increments due to a lack of automating their continuous integration activities. Even though the team can complete the functional tasks they agreed on before starting each iteration, their lack of continuously integrating their software, results in limited testing being done on the product. In review, the team can present a functional product meaning success to stakeholders, but their lack of integration means failure to the team. A natural evolution of process would suggest the team negotiate time in the following iterations to solve their integration problems and even though they may deliver less functionality while this is being done, they would have a better chance of succeeding and potentially moving faster in the future, meaning success to the team. This approach, however, means that stakeholder expectations cannot be met by demonstrating the progress they had become accustomed to, meaning failure to stakeholders.*

This section describes the impact cognitive bias has on the software outcomes. The description explains several concepts and provides an example to augment the understanding of the impact of this phenomenon on group understanding. In terms of the theoretical basis of this argument, the section is considered valuable to both academic and pragmatic readers.			

6.5 Argument

This research does not question the value of methods, practices or processes followed by development teams to produce software. It does not question how they are conceived, nor does it suggest they are incapable of facilitating the production of quality software. The research suggests that in the absence of formal approaches aimed at collecting empirical evidence on the failure or success of software development projects, software development teams are forced to record different expert opinions and perspectives. It suggests that, as experts innovate in the realm of software practice, methodologies are born underpinning bias, rather than fact. As experts create perspectives based on their knowledge and experiences, factual cause and effect could be considered diluted with social prejudices.

This research argues that a plausible and indirect relationship exists between the social class of experts working in software development teams and the understanding of factors that affect the outcomes of a software development initiative. It argues that professionals capable of providing authoritative reasoning propose reasons associated with the result of software development initiatives being successful or unsuccessful and suggests that these reasons are drawn from their “particular” attitude towards the situation as well as their practical understanding and justified beliefs. The research suggests that an expert’s point of view is based on the construction of their reality and is rooted in their social class.

As such, it is proposed that experts in diverse social classes may provide different interpretations of factors that influence software development outcomes. This proposition serves as the basis and scope of research testing.

This section provides a description of what was found during the data analysis phase of the research. Before providing a conclusive result to the primary research question, each sub question is explained in terms of the analysis that was done. A such, this section is considered valuable to both academic and pragmatic readers.





6.6 Research findings

This study aims at understanding whether there is a difference in expert opinion on factors that influence software development outcomes. As no research is available to understand whether experts in different social classes reason differently about the outcomes of software projects, a study testing expert opinion among the various expert groups is considered valuable. The study attempts to determine the relationship between various groups present in a software development teams and their perspectives of factors in terms of "internal variability" and "political impacts" as "influences" of software development outcomes.

The independent variable, experts in different groups, refers to social class suggesting that several sub-questions must be answered in order to respond to the research question conclusively. As such, the independent variable is decomposed into three distinct areas that include, role, tenure and generation.

The research reveals that in terms of the factors that influence the outcomes of software development projects in South Africa, there is a statistically significant difference between the opinion of experts in various roles and having different levels of experience but found that there is however no statistically significant difference in opinion between experts in different generations.

6.6.1 Experts in different roles

The study reveals that in terms of a comparison between roles, a notable degree of difference can be found. In addition, it is found that apart from the relationship between developers and architects, all other relationship differences are statistically significant, thus suggesting that experts in different roles in a software development team have differing views of "internal variability" and "political impacts" as the factors that

influence bespoke software development outcomes in South Africa could be considered plausible.

As such, a scientific argument can be made that is both statistically significant and relevant suggesting that when factors that influence software development outcomes are considered at a group level, and the group consists of different roles; should the outcome not be based on measurement or empirical evidence, it can be viewed as biased and flawed.

6.6.2 Experts in different generations

The study reveals that in terms of a comparison between generations, a degree of difference can be found. However, statistical significance is found in only four out of the ten relationships, thus suggesting that there is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts from different generations based on the classification of generations used in this study.

As such, no argument can be made suggesting that when factors that influence software development outcomes are considered at a group level, and the group consisted of different generations; should the outcome not be based on measurement or empirical evidence, it can be viewed as biased and flawed as no statistically significant evidence can be produced that underpins this as a fact.

6.6.3 Experts with different tenure

The study reveals that in terms of a comparison between tenures, a notable degree of difference can be found. In addition, it is found that seven out of a total of ten relationships are statistically significant, thus suggesting that experts with different tenure in a software development team have differing views of “internal variability” and “political impacts” as the factors that influence bespoke software development outcomes in South Africa could be considered plausible.

As such, a scientific argument can be made that is both statistically significant and relevant suggesting that when factors that influence software development outcomes are discussed at a group level, and the group consists of individuals with different tenure; should the outcome not be based on measurement or empirical evidence, it can be viewed as biased and flawed.

6.6.4 Experts in different groups

When the results of all sub-questions are considered, it is found that in certain cases a scientific argument cannot be made. The study reveals that difference could be found between experts in different roles, tenure and generation, but in terms of significance, the research can only prove that role and tenure have statistically significant differences, meaning that neither H01 nor H1 can be rejected and a more granular alternative hypothesis would be more appropriate.

As such, a scientific argument can be made that is both statistically significant and relevant suggesting that when factors that influence software development outcomes are considered at a group level, and the group consists of individuals with different roles and tenure; should the outcome not be based on measurement or empirical evidence, it can be viewed as biased and flawed.

The following alternative hypothesis is accepted as plausible:

> *Ha1: Software development experts in different roles and tenure of a software development team have differing views of "internal variability" and "political impacts" as factors that influence bespoke software development outcomes in South Africa.*

This section highlights the deficiencies of the research and explains a marginal shortcoming in answering the primary research question. As such, the section is considered most valuable to academic readers.





6.7 Research deficiency

6.7.1 Social class

Moe (2017) noted that society could be classified into a hierarchy of social classes, based on several different dimension that included age, designation and level of experience (Moe, 2017). It is argued that each contributor categorised under social class has the ability to contribute toward creating individual perspective. It was however decided to only introduce age, role and level of experience as independent variables to test, due to the ease of their measurability. This narrow focus is considered appropriate as it assists in maintaining a scope broad enough on which to conduct a relevant study. However, this decision is based on the expectation that each selected independent variable would be able to prove statistical significance and as such can be considered relevant to represent social class in its entirety. The inability of proving statistical significance for all independent variable means that the outcome of this research cannot be generalised across social class.

The theory created as part of the research suggests that social class has an indirect relationship with the factors that influence software development outcomes. Even though role and tenure are technically considered part of social class, all that can be proven is that an indirect relationship existed between factors that influence software development outcomes and software development groups with different roles and tenure. As such, based on the evidence provided by this study, the theory cannot be decisively proven as the new evidence that came to light cannot be generalised across social class as an abstract concept.

6.7.2 Data collection focus

Rebuttal

It is suggested that a survey study is complex, especially if the participants are spread across a wide geographical area such as South Africa. The survey requires participants to provide insight into reasons for software failure and success, knowing that the research sample would not have a common development project to base their opinion on. This, by its very nature raises questions on the internal validity of the findings and the overall generalisation of the study. As such, it may be reasoned that the research outcome is flawed as different software development projects may have different and very clear reasons that underpin their outcomes. It could also be argued that participating experts may perceive the reasons for failure or success equally, regardless of their roles or tenure. As a result, several perspectives may be argued, such as:

- Experts with more experience often work on the more complex parts of a software development project than those with less experience. In addition, experts with more experience often have more specific information to work with than those with less experience. As such, a difference in perception is circumstantial as experts with less experience are not exposed to these complexities, do not have the same information and would have a dissimilar experience to skilled experts.
- Software development projects vary in construction time, have dissimilar domains and team constructs, and are externally influenced in dissimilar ways. Not all software development efforts are equal and as such cannot be compared.
- Software development teams use different technologies, have different execution methodologies and measure quality differently. In addition, it could be argued that the effectiveness of a methodology is related to the maturity of the software development team, the software development company and its management and clients. These factors drive different behaviour and serve different outcomes and as such cannot be compared.

- Different teams with varying levels of tenure and maturity understand and experience internal variability and political impacts differently. As such, mature teams are naturally equipped to deal with both internal variability and political impacts.

Surrebutter:

It is suggested that regardless of complexity or information quality, experts at different levels would naturally have dissimilar experiences. What may be a simple task for a senior resource may be a challenge to a junior resource, suggesting a difference in their perceptions. The study supports this argument highlighting a notable and statistical difference between experts with different tenure.

In addition, it is suggested that regardless of construction time, domain, team construct, external influences, technology and methodology selection; a comparison is possible. Software development projects have one of two outcomes, success or failure and these opposing results symbolizes the finality of the entire development effort. It can be reasoned that in order to provide a basis for comparison, commonality is required. In terms of this study, one of two approaches are considered appropriate as both provide a common ground for comparison. These include a common software development project and a common set of reason for failure or success.

Common development project

It stands to reason that a common development project executed by the entire research sample is the most appropriate way of comparing expert opinion in terms of the outcomes of the project and the reasons that underpin it. This approach leverages the commonality of one development project and compares the reason for failure or success, research respondents provide. Even though this approach may reduce questions raised on the generalisation and internal validity of the survey outcome, incorporating it into any research study may prove to be challenging.

Crowdsourcing provides an interesting avenue. The ability to leverage a population to fulfil the requirements of a software development project can overcome common problems

experienced in the industry, such as a lack of highly skilled resources at low cost. Crowdsourcing forms the basis of open-source software development where, apart from a small number of dedicated resources, a software development project is staffed from the population, generally at no cost. It allows developers with any skill level working from different locations to contribute towards a development project. The nature of these developments are, that most project information is publicly accessible, providing a rich resource base that could be incorporated into any research study. However, in addition to gaining access to local developers contributing towards open-source software development projects; it is argued that these projects do not have the generalisation properties required for this study.

In terms of their operation and value proposition, South African organisations understand the value software provides. Software systems are core to their success and protecting the intellectual property embedded in these systems are of utmost importance. As such, organisations may be reluctant to share information about the development of their systems and to protect intellectual property, would be sceptical of open-source operational systems.

A common development project as an approach would require a different research methodology, as the researcher would have no control over the reasons experts provide in terms of failure or success of the development project. In addition, a coding technique capable of consolidating the data into a workable set has the potential of introducing researcher bias into the study as an interpretation of the data is required before statistical analysis would be possible.

A fundamental flaw with using this approach is that the findings have the potential of being biased by the controlling body. A common software development project would be governed by a single sponsor, limiting the statistical relevance when political impacts are considered.

A common set of reasons for failure or success

When considering a common set of reasons for failure or success of software development projects, a different dimension is provided. This approach leverages the commonality of a defined set of reasons for failure or success and as such can extend data collection across internal variability and political impacts. In terms of this study, the defined set of factors is extracted from research outcomes in existing empirical studies.

As opposed to collecting reasons for failure and success from a single development project, then reducing the collected data into a workable dimension; this approach performs data-reduction up-front and as such, does not have to deal with interpreting new factors introduced by the research sample. In addition, the limitations in terms of statistical relevance introduced by focusing on a single development sponsor, is eliminated. It is argued that the reasons for failure and success of software development projects are well-established and utilising the established body of knowledge is common practice. Existing and well-known studies done by large international organisations such as the Standish Group (The Standish Group, 1994, 1999, 2001, 2011) and McKinsey (Bloch, Blumberg and Laartz, 2012; McDonald, 2012) continue their research into factors that influence the success or failure of software development projects by focusing on the outcomes and commonalities associated to them, rather than the commonality introduced by a single software development project. The difficulties and risks associated with focusing on a single development project are severe enough to eliminate it as an approach to data collection.

Redirect

A statistically significant difference between perceptions of experts in dissimilar roles as well as experts with different tenure cannot be refuted. Validity is secured by focusing on the foundation on which the research instrument was created. The aggregation of factors influencing the outcomes of software development project into internal variability and political factors are founded in established empirical research. These efforts aimed at

highlighting common causes for software success and failure and managed to establish a foundation future research could be built upon.

As such, it is argued that regardless of working on the same software development effort, when considering experts in different roles and tenure, perception differs when common causes for software development failure or success are considered.

This section concludes the discussion part of the research by highlighting some of the most important parts of the of the study. The content is considered valuable to all reader types.





6.8 Conclusions

In his work, Neuman (2013) wrote that causality could only be established if all three requirements of the temporal order, empirical association and the elimination of plausible alternatives were considered (Neuman, 2013). It is well understood that these requirements form the basis of defining causality in an academic research realm. From a non-academic perspective, it can be argued that when discussing software development outcomes, teams consider the order in which events occurred as part of their understanding mechanism. This research does not focus on evaluating the truth of such an argument and as such does not question its validity; nor does it investigate mechanisms software development teams employ to eliminate plausible alternatives. This research focuses on empirical association and finds little evidence of any approach aimed at providing verifiable data capable of demonstrating that the suggested factors for failure or success of software development projects are factually correct. This finding is however not unique as several researchers are suggesting a severe lack of available empirical evidence as well as a lack of commitment and maturity in recording it.

The continuous debate between several parties around the fundamentals of software development being an art or a science remains, but all appear to have an agreement in that the practice should be straightforward and predictable. This study suggests that the lack of formal approaches aimed at collecting empirical evidence on failure or success of software development projects creates a misguided view of the actual state of software engineering. It suggests that the ‘software crisis’ has remained over the years due to an inherent negativity bias of the industry. It is also noted that the ‘software crisis’ provides a breeding ground for method and practice evolution and suggests that methodologies are natively biased towards an audience capable of aligning with the concrete perceptions they are created under.

This research scientifically proves that there is a measurable and statistically significant difference in how experts in different groups, specifically roles and tenure, reason about

the factors that influence software development outcomes. Based on the evidence, it is suggested that expert perspectives are based on their biases, and as experts innovate in the realm of software methods and practices, it can be argued that these biases make their way into methodology evolution. No research could be found that demonstrated that experts reason differently in different groups, suggesting that this contribution is both valuable and unique. However, this finding manages to open a Pandora's box and raised several controversial questions.

Did the software engineering industry become a slave to methodologies? Methodologies are believed to suffer from the same fate as fashion. Their popularity is thought to reduce once a new one arrives. It is noted that software development teams often adopt new methods to remain 'cool' and in some cases, it can be argued that adoption is driven by an attempt to remain mainstream. However, it is believed that failing teams would have considered new method adoption in a final attempt to 'fix' their broken way of working. It is suggested that the adoption of methodologies is largely driven by individuals and in terms of this research, it is demonstrated that opinions might differ when groups are considered. As such, the adoption of the fad may or may not align with the overall perspective of the whole team.

Would software teams and organisations alike know and understand what the impact is on its native culture when adopting a new methodology or approach? It is noted that methods are biased towards an audience capable of aligning with the concrete perceptions they are created under. It is observed that cultural difference might exist between the team, its governing organisation, the customers they assist and the fundamentals of the methodology. Should the adopted methodology not align with the inherent culture of the team, organisation and customer; the environment may experience an evolution in terms of morphing the new method to align with what was previously used. This phenomenon is evident in the local industry, where Agile frameworks are adopted to replace Waterfall and due to the organisational culture, the Agile framework morphed into something resembling Waterfall, commonly referred to as WetAgile. This approach over time results in the practices of Agile frameworks being neglected and ultimately dropped, yet the

terminology and original reason for adopting them remain. When deciding to adopt a different methodology, organisations need to understand the biases the methodology underpins and measure its fit-for-purpose properties against the organisation's culture.

How would organisation know that adopting a different methodology would add value to their execution and ultimately the quality of delivery? Methodologies are described as repeatable and proven practices that could be employed to solve common problems in software engineering. It is argued that methodologies become valuable when there is a weak understanding of how to address a software development issue and as such suggest that there are varying levels of essential team skills. If a team can be considered expert, has experience and a track record of developing and delivering quality software, would such a team need to change the way they build software if the environment remained the same? Equally, would a young and healthy 22-year-old male go for a coronary artery bypass if there is no signs of heart disease? The Agile Manifesto's opening line suggests that "we are uncovering better ways of developing software by doing it and helping others do it" (Fowler and Highsmith, 2001). It is contended that this, by its very nature is flawed in that it assumed that everyone has the same problems. This research argues that without empirical data collection, organisations continue to be visionless as to the value different methodologies provide. This argument is founded in that organisations are already working blind as to the value its current way of working has added, as no approach to formal measurement and data collection is employed. Also, organisations fixate on the perspectives of the experts they employ, and as suggested, expert judgement is founded in individual bias. From a group perspective, it is found that dissimilar beliefs exist between different roles and tenure of the development environment, meaning group perspectives cannot be trusted. It is argued that organisations have to approach the evaluation of value in terms of methodology adoption from an empirical perspective that includes formal measurement and data collection.

Why would empirical data collected be considered valuable if teams are enabled to re-invent themselves to align to quality? Author, life-long student and creator of the personal and team software processes, Watts Humphrey (2017) noted that "if you don't know

where you are, a map won't help" (Software Engineering Institute, 2017). Watts' analogy on assessment was considered a foundational problem in software manufacturing. It is argued that when everything goes per plan and teams deliver quality repeatedly; there is no need to measure anything. However, when the proverbial wheel comes off the wagon, suddenly measurement is deemed critical. It can be argued that measurement and the collection of empirical data must enjoy equal attention, regardless of the state of development, as this approach can increase team maturity and given enough time, become the DNA of the team and the organisation supporting it.

Why is this research important?

> *In a perfect world, there is the concept of a Phoenix team; a mythical software development team that is both successful and non-empirical as well as capable of building maturity through rationalism. This team bases their approach to evolution on innovative thinking, rational and deductive reasoning and often re-create themselves out of the ashes of their learnings. The Phoenix team is effective, trusted, enabled and by its very nature, extremely capable. Methodology, processes and practices do not feature for them. They consume it when it makes sense, alter it when they need to and eliminate it when it is no longer valuable.*

The software industry, however, exists in a real world; one where software engineering has been plagued by the 'software crisis' for more than half a century. One where developing software is a complex task, filled with a multitude of differing approaches, all aimed at making the experience of crafting software easier. Measurement and empirical data collection are critical to this environment as without which, chaos management becomes the order of the day. This research argues that should data not be collected to assist in understanding why some software development initiatives succeed while others fail, all that is left is expert judgement. The study reveals that reliance on data collected from expert judgement cannot be trusted as experts in different roles and tenure reason differently when software failure and success if examined. Should this be the only hope in

understanding factors that influence software development outcomes, it must be noted that the conclusion can be based on the biases of those providing the insight.

7 Future work

This section provides an academic discussion on the value of replicating the research study by providing several ideas to augment the value this study already established. As such, this section is considered most valuable to academic readers.



7.1 Replicating research

Core to quantitative research is the burden of proof. By performing this research, it quickly became evident that no words would be able to describe the validity of understanding gained by this study. Scientific protocols assist in removing the threat of data being coincidental, but as described in several chapters of this research, individual perspective typically contains some bias. Although the researcher undertook to evaluate every aspect of the research to test for personal bias injection, readers of the study may create their own opinions while reading this work.

Also, the research topic has the potential to span a knowledge base wider than what was incorporated. The research scope is narrowly defined and is strictly monitored to ensure that the study can be completed in due course while adding a fresh perspective to the existing bodies of knowledge. Thus, it is believed that this research can potentially ignore the larger context of the problem space.

Research by its very nature is a repetitive exercise; as new theories come to light while others are being tested. Thus, the ability to replicate this study would add additional benefit to the topic, especially when conducted in a different setting. Each area of the research, such as its design, methodology, instrumentation and analysis protocols are documented as to allow the replication of this study in the future.

In addition to replication studies, the research presented in this dissertation is narrow enough to be extended. During the research, several areas of interest are uncovered, some with the potential to augment the understanding created by this research, others to widen the topic to include the larger context.

The following provides a high-level explanation of potential future work:

- *What is the relationship between experts from different social classes and their perspectives of the factors that influence software development outcomes?*

  This research aims to understand whether software development experts in different groups reason differently about the factors that influence software development outcomes. In defining the research question, a theory is created that suggests an indirect relationship between the perceptions of experts on software outcome reasons and social class. The research assumes that the incorporation of role, tenure and generation is sufficient for generalisation across social class. During data analysis and after closer evaluation of this assumption, it was deemed flawed as social class is considered a clear and wide area of research. Thus, this research can be expanded to include social class in its native form to test the proposed theory by restating the research question to include social class, rather than different groups.

- *What is the relationship between experts in different groups and their perspectives on the factors that influence software development outcomes in relation to software development methodologies employed during development?*

  This research touches on software development methodologies, practices and principles, especially in explaining the role of 'software crisis' in the evolution of the industry. The study does however not delve into details about the relationship of expert opinion on software failure factors in relation to the methodologies employed during execution. The research can be extended to incorporate an additional independent variable of methodology to increase the understanding of how experts using different methodologies reason about the factors that influence software development outcomes.

- *What are the contributions of the testers, users, stakeholders or organisations as first-class citizens in relation to the perspectives of experts on the factors that influence software development outcomes?*

This research expands its scope to include roles that were historically considered software development roles. These include developers, business analysts, project managers and architects. As the industry has matured, other roles have become core to the development of software and in some cases, are considered part of the software development team. The research can be expanded to incorporate those roles that included testers, users, stakeholders and even the organisation controlling the software team as experts in a software development team. In doing so, the research would increase its generalisation from conventional teams to include new-age development teams.

- *What constitutes being an expert in a software development team?*
  The research refers to experts of the software development team and defined the concept to some degree but does not provide details as to exactly what constituted being an expert. Research can be done into how experts are created in the software industry, what role they play in a software development organisation and the impact they have on the evolution of software engineering. Also, the outcomes can form part of a replication of this study to refine its findings.

- *Increase the overall generalisation qualities of the research by removing the bounded context of South Africa.*
  The research is bounded to the local software industry and as such collected evidence from South Africa alone. Due to a lack of relevant local research, the study had to include the bodies of knowledge that cross this boundary, and as such, an international perspective is provided in terms of the background of the study. Due to the bounded context of South Africa, the research can only be generalised across South Africa and not internationally. This research can be replicated to include a word perspective to allow the outcome to be generalised across the entire international software industry.

This section provides and academic discussion on the generalisation and transferability properties of the research. As such, it is considered most valuable to academic readers.





7.2 Research generalisation

Research can only be deemed successful if it can be generalised across a whole population. In considering generalisation, every attempt is made to ensure the research can be regarded as reliable and valid. However, the bounded context of the research limits the collection of data to the local South African software industry. While bounded, the focus remains on the ability to promote generalisation and transferability properties.

7.2.1 Generalization

The generalisation properties of any research study depend on the population sample used to collect the research data. It is argued that without a clear design and significant intervention, research outcomes cannot be considered representative of the greater population.

In terms of the local software industry, no indication can be found that describes the makeup of the entire population. In addition, South Africa is a country with nine provinces, all potentially contributing towards the local software engineering effort. With a limited view, the ability to generalise the research outcomes may have been at risk.

As such, during design it was decided to stretch the sample population to a number that could not be argued against, suggesting a total population size of 500,000 professional software employees. The decision resulted in a research sample size of 384 respondents[15].

Sadly, as no estimations can be made that suggested the population size of each province, data collection does not incorporate a specific view of each province. Even though data is collected from the entire country and as such, can be considered generalised across the whole of South Africa, insight into specific areas cannot be given.

[15] See Annexure A (*Sample size calculation reasoning*) for more details on how the sample was calculation

7.2.2 Transferability

When considering the validity of the research outcomes, transferability properties have to be designed and built into the study. Transferability is understood as a research quality attribute suggesting an ability to project the outcomes across a greater population that is made up of all the constituent aspects of the population.

Even though the research outcomes are considered sufficiently transferable in terms of their bounded context, the research scope limits transferability to software development teams previously considered conventional and utilising the professional services of developers, business analysts, project managers and architects.

This section provides a short but conclusive summary of the dissertation. As such, it is considered valuable to all reader types.





8 Summary

In a perfect world, a software development team would understand the requirements, have the necessary skills to complete its tasks and recognise the pitfalls they could encounter along that way. They base their approach to evolution on innovative thinking, rational and deductive reasoning and often re-create themselves out of the ashes of their learnings. They are efficient, trusted, enabled and by their very nature, extremely capable. Methodology, processes and practices do not feature for them. They consume them when it makes sense, alter them when they need to and eliminate them when they are no longer valuable. Unfortunately, in the real world, formal approaches aimed at empirically collecting evidence on the factors that influence software development outcomes have been neglected, resulting in recording different expert perspectives. As experts innovate in the realm of software practice, methodologies are believed to be born underpinning bias, rather than fact.

The ‘software crisis’ endured the test of time but is not considered a primary contributor towards the evolution of software practice. This research is significant in that no research has been done into the phenomenon that drives software methodology evolution and the fundamental principles that underpin its motivations. This research provides concrete evidence on the topic and advances the existing bodies of knowledge by offering a relevant view of why practices evolve and why software development teams should be selective when adopting a software development methodology.

The research investigates the differences in recognising factors that influence bespoke software development outcomes in South Africa between software development experts in different groups of a software development team and found statistically significant differences in perspectives of experts when different roles and tenure are considered.

The research concludes that ‘software crisis’ became an abstract concept over the years and when combined with real perspectives of failure, a breeding ground for software method evolution was created. In addition, the research suggests that expert judgement contains several cognitive biases and as software development experts innovate in the realm of software methodology, methods may be regarded as biased towards audiences responsible for its evolution.

Finally, the research argues that due to the inherent biases of software methodologies, experts and organisations alike should consider employing measurement aimed at collecting empirical evidence on the state of their current processes before blindingly adopting a different approach. In addition, experts should consider empirical data collection as a primary mechanism for understanding the factors that influence software development outcomes.

9 References

Abeyasekera, S. (2005) *Quantitative analysis approaches to Qualitative data: Why, When and How*. Edited by J. D. Holland and J. Campbell. Warwickshire: ITDG Publishing.

Ahmadzai, N. (2016) 'Why Do Projects Crash & Burn in Fragile Countries?', *Integrative Business and Economics Research*, 5(1), pp. 315–328.

Al-Ahmad, W. *et al.* (2009) 'A taxonomy of an IT project failure: Root Causes', *International Management Review*, 5(1). Available at: http://www.usimr.org/IMR-1-2009/v5n109-art8.pdf.

Ambler, S. W. (2010) 'Scaling agile: an executive guide', *Agility@ Scale Whitepaper*. Somers: IBM Corporation Software Group. Available at: ftp://170.225.15.26/software/emea/de/rational/ekit/Scaling_Agile.pdf.

Ambler, S. W. (2013) 'Going Beyond Scrum: Disciplined Agile Delivery'. Boston: Disciplined Agile Consortium.

Atlee, J. (2002) 'Software Engineering Profession and Discipline Director of Software Engineering University of Waterloo'. Ontario: University of Waterloo.

Baddeley, M. C., Curtis, A. and Wood, R. (2004) 'An introduction to prior information derived from probabilistic judgements: elicitation of knowledge, cognitive bias and herding', *Geological Society, London, Special Publications*, 239(1), pp. 15–27. doi: 10.1144/GSL.SP.2004.239.01.02.

Baggen, R. *et al.* (2011) 'Standardized code quality benchmarking for improving software maintainability', *Software Quality Journal*, 20(2), pp. 287–307. doi: 10.1007/s11219-011-9144-9.

Baratta, A. (2006) 'The Triple Constraint, A Triple Illusion', in *PMI Global Congress Proceedings*. Seattle, pp. 1–6.

Beauchamp, T. L. and Childress, J. F. (2001) *Principles of Biomedical Ethics*. 5th edn. Edited by T. L. Beauchamp. Washington, D.C.: Oxford University Press. doi: 10.1177/004057368003600423.

Benington, H. (1983) 'Production of large computer programs', *Annals of the History of Computing*, 5(4), pp. 350–361. doi: 10.1109/MAHC.1983.10102.

Berander, P. *et al.* (2005) *Software quality attributes and trade-offs, Blekinge Institute of Technology*. Available at: http://www.uio.no/studier/emner/matnat/ifi/INF5180/v10/undervisningsmateriale/reading-materials/p10/Software_quality_attributes.pdf (Accessed: 3 August 2013).

Bijker, W. E., Hughes, T. P. and Pinch, T. J. (1987) *The Social Construction of Technological Systems*. Edited by Bijker Wiebe E, Hughes Thomas P, and Pinch Trevor. Massachusetts: MIT Press. doi: 10.1177/030631289019001010.

Bloch, M., Blumberg, S. and Laartz, J. (2012) *Delivering large-scale IT projects on time, on budget, and on value, Digital McKinsey*. Available at:

https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value.

Boehm, B. (1972) 'Software and Its Impact: A Quantitative Assessment'. Available at: https://www.rand.org/content/dam/rand/pubs/papers/2008/P4947.pdf.

Boehm, B. (2006) 'A view of 20th and 21st century software engineering', *Proceeding of the 28th international conference on Software engineering - ICSE '06*. New York: ACM Press, p. 12. doi: 10.1145/1134285.1134288.

Boghossian, Z. J. (2002) *An investigation into the critical success factors of software development process, time, and quality*. Pepperdine University Malibu, CA.

Borden, L. M. (2016) *Understanding t-test*. Arizona. Available at: https://reachmilitaryfamilies.umn.edu/sites/default/files/rdoc/Understanding t Test_0.pdf.

Brooks, F. P. (1986) 'No silver bullet-essence and accidents of software engineering', in Kugler, H. j (ed.) *Proceedings of the IFIP Tenth World Computing Conference*. Amsterdam, pp. 1069–1076. Available at: http://www.cgl.ucsf.edu/Outreach/pc204/NoSilverBullet (Accessed: 20 November 2013).

Brooks, F. P. (2002) *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition*. 20th Anniv. Edited by H. J. Kugler. Boston: Addison Wesley Longman, Inc.

Bruner, J., Vygotsky, L. and Feuerstein, R. (2008) *Constructivist Theories, Constructivist Theories*. Available at: http://mennta.hi.is/starfsfolk/solrunb/construc.htm (Accessed: 13 May 2017).

Buckler, C. (2010) *10 Reasons Why Software Project Estimates Fail, Sitepoint*. Available at: http://www.sitepoint.com/10-reasons-why-software-project-estimates-fail/ (Accessed: 20 June 2017).

Buettner, M. *et al.* (2006) *Software Engineering is indeed in a crisis*. Seattle. Available at: http://www.cs.washington.edu/education/courses/503/08wi/crisis-pro.pdf (Accessed: 13 November 2013).

Burgman, M. *et al.* (2006) 'ACERA Project 0611 Eliciting Expert Judgments', *Risk Analysis*, pp. 1–71.

Campbell, D. and Campbell, S. (2008) 'Introduction to Regression and Data Analysis'. New Haven: Yale University.

Charette, R. N. (2005) *Why Software Fails, IEEE Spectrum*. Available at: http://spectrum.ieee.org/computing/software/why-software-fails (Accessed: 23 May 2017).

Charvat, J. (2003) *Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects*. Edited by M. Holt and T. Hummel. Hoboken, New Jersey: John Wiley & Sons.

Colburn, A. *et al.* (2008) 'There is no software engineering crisis', pp. 1–5. Available at: http://www.cs.washington.edu/education/courses/503/08wi/crisis-con.pdf (Accessed: 13 November 2013).

Colman, A. M. and Pulford, B. D. (2008) *A Crash Course in SPSS for Windows*. 4th edn.

Edited by John Wiley and Sons Ltd. West Sussex: Wiley & Blackwell.

Coughlan, M., Cronin, P. and Ryan, F. (2007) 'Step-by-step guide to critiquing research. Part 1: quantitative research.', *The British journal of nursing*, 16(11), pp. 658–663. doi: 10.12968/bjon.2007.16.11.23681.

Creswell, J. W. (2014) *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 4th edn. Edited by H. A. Night V, Koscielak K, Bauhaus B, Markanich M. Lincoln: Sage Publications, Inc.

Crockford, D. (2007) *Software Crisis, Software Quality and Douglas Crockford, Yahoo 2007 Frontend Engineering Summit*. Edited by J. Salado. United States: Yahoo.

Crotty, M. (1998) *The foundations of social research: meaning and perspective in the research process*. 1st edn. Edited by D. Bourget and D. Chalmers. London: SAGE Publications Ltd. doi: 10.1111/j.1469-0691.2011.03558.x/pdf.

Dalal, S. and Chhillar, R. S. (2012) 'Case Studies of Most Common and Severe Types of Software System Failure', *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8), pp. 341–347. Available at: http://www.ijarcsse.com/docs/papers/8_August2012/Volume_2_issue_8/V2I800209.pdf (Accessed: 20 January 2014).

Dalcher, D. (2014) 'Success in Software Projects: Looking Beyond the Failure Factors', *Software Project Management in a Changing World*, pp. 247–273. doi: 10.1007/978-3-642-55035-5.

DeCoster, J. (2006) 'Testing group differences using t-tests, ANOVA, and nonparametric measures'. Charlottesville: Statistics Explained. Available at: http://www.stat-help.com/ANOVA 2006-01-11.pdf.

Dorsey, P. (2005) *Top 10 reasons why systems projects fail*. Available at: https://sites.hks.harvard.edu/m-rcbg/ethiopia/Publications/Top 10 Reasons Why Systems Projects Fail.pdf (Accessed: 20 January 2014).

Dvorsky, G. (2013) *The 12 cognitive biases that prevent you from being rational, Institute for Ethics and Emerging Technologies (IEET)*. Available at: https://ieet.org/index.php/IEET2/more/dvorsky20130110?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+EthicalTechnology+Ethical+Technology (Accessed: 9 August 2017).

Dwolatzky, B. (2010) *Some further thoughts on the professionalisation of SA's ICT sector, The Software Engineer*. Available at: http://www.softwareengineer.org.za/professionalisation-of-sas-ict-sector/296/ (Accessed: 2 April 2014).

Ebbesen, J. B. and Hope, A. (2013) 'Re-imagining the Iron Triangle: Embedding Sustainability into Project Constraints', *PM World Journal*, 2(Iii), pp. 1–13. doi: 10.1146/annurev.nutr.26.061505.111236.

Eliot, B. T. S. (1921) *The Sacred Wood: Essays on Poetry and Criticism*. Edited by A. A. Knopf. New York: Alfred A. Knopf.

Elliott, R. and Timulak, L. (2005) *Descriptive and interpretive approaches to qualitative*

research, A Handbook of Research Methods for Clinical and Health Psychology. Edited by J. Miles and P. Gilbert. York: Oxford University Press. doi: 10.1093/med:psych/9780198527565.001.0001.

El Emam, K. and Koru, A. G. (2008) 'A Replicated Survey of IT Software Project Failures', *IEEE Software*, 25(5), pp. 84–90. doi: 10.1109/MS.2008.107.

Ensmenger, N. (2010) *History of Computing : Computer Boys Take Over : Computers, Programmers, and the Politics of Technical Expertise*. 1st edn. Edited by W. Aspray. London: The MIT Press.

Ensmenger, N. (2017) *The computer boys take over, The compueter boys take over*. Available at: http://thecomputerboys.com/?tag=crisis (Accessed: 1 July 2017).

Fitzgerald, B. (2012) 'Software Crisis 2.0', *Computer*, 45(4), pp. 89–91. doi: 10.1109/MC.2012.147.

Fowler, M. and Highsmith, J. (2001) 'The agile manifesto', *Software Development*, (August). Available at: http://www.pmp-projects.org/Agile-Manifesto.pdf (Accessed: 20 November 2013).

Frederiksz, H., Hedeman, B. and Heemst, G. V. van (2010) *Project Management Based on PRINCE2 2009 Edition*. 2009th edn. Edited by S. Newton. Van Haren Publishing.

Gallant, R. (2016) *5 things recipients find annoying about surveys (and how to fix them!), Giftbit*. Available at: http://blog.giftbit.com/article/5-things-recipients-find-annoying (Accessed: 20 May 2017).

Garland, R. (1982) *Microcomputers and Children in the Primary School*. Edited by R. Garland. London: Falmer Press.

Gartner (2008) *Gartner Says Worldwide IT Spending On Pace to Surpass $3.4 Trillion in 2008, Gartner Press Release*. Available at: http://www.gartner.com/newsroom/id/742913 (Accessed: 12 May 2017).

Garvin, D. A. (1984) 'What does "Product Quality" really mean?', *MIT Sloan Management Review*, 26(1). Available at: http://www.oqrm.org/English/What_does_product_quality_really_means.pdf.

Geethalakshmi, S. N. and Shanmugam, A. (2008) 'Success and Failure of Software Development: Practitioners' Perspective', in *International MultiConference of Engineers and Computer Scientists*. Hong Kong, pp. 19–21.

Goodfriend, W. (2017) *Sociology's Four Theoretical Perspectives: Structural-Functional, Social Conflict, Feminism & Symbolic Interactionism, study.com*. Available at: http://study.com/academy/lesson/sociologys-four-theoretical-perspectives-structural-functional-social-conflict-feminism-symbolic-interactionism.html (Accessed: 1 May 2017).

Götschl, J. (1995) *Revolutionary Changes in Understanding Man and Society*. Edited by J. Götschl. Austria: Springer Science & Business Media Dordrecht.

Green, S. (2009) 'Agile Development Meets Cloud Computing for Extraordinary Results at Salesforce.com'. San Francisco: Salesforce.com.

Greenwood, D., Khajeh-Hosseini, A. and Sommerville, I. (2010) *Lessons from the Failure and Subsequent Success of a Complex Healthcare Sector IT Project*. St Andrews. Available at: http://arxiv.org/abs/1003.3880.

Grey, J. (2011) *The development of a hybrid agile project management methodology*. North-West University.

Guest, K. (2014) *R1bn IFMS project scrapped, ITWeb*. Available at: http://www.itweb.co.za/index.php?option=com_content&view=article&id=134290:R1bn-IFMS-project-scrapped&catid=250 (Accessed: 23 November 2017).

Gulla, J. (2011) 'Seven Reasons Why Information Technology Projects Fail', in *Share in Orlando*. Orlando. Available at: https://share.confex.com/share/117/webprogram/Handout/Session9341/seven Reasons Why Information Technology Projects Fail.pdf (Accessed: 15 January 2014).

Haigh, T. (2010) '"Crisis, What Crisis?" Reconsidering the Software Crisis of the 1960s and the Origins of Software Engineering', in *2nd Inventing Europe/Tensions Of Europe Conference*. Sofia, pp. 1–36. Available at: http://link.springer.com/article/10.1007/s005860050018 (Accessed: 13 November 2013).

Hallows, J. (1998) *Information Systems Project Management: How to Deliver Function and Value in Information Technology Projects*. Edited by J. Hallows. New York: AMACOM Division of the American Management Association.

Harrington, A. (ed.) (2005) *Modern Social Theory: An Introduction*. New York: Oxford University Press.

Hayes, I. S. (2002) *Just Enough Wireless Computing*. Edited by W. Mara. New Jersey: Prentice Hall Professional Technical Reference.

Heeks, R. and Bhatnagar, S. (1999) *Understanding Success and Failure in Information Age Reform, Reinventing Government in the Information Age*. Edited by R. Heeks. London: Routledge.

Henderson, P. (2006) *Why large it projects fail*. Southhampton. Available at: http://pmh-systems.co.uk/phAcademic/papers/LargeIT.pdf (Accessed: 20 January 2014).

Holgeid, K. (2013) 'A Reflection on Why Large Public Projects Fail', in Römmele, A. and Schober, H. (eds) *The Governance of Large-Scale Projects - Linking Citizens and the State*. Nomos, pp. 219–243.

Houghton Mifflin Company (2000) *The American Heritage Dictionary of the English Language*. 4th edn. Houghton Mifflin Company. Available at: http://www.thefreedictionary.com/influence (Accessed: 23 April 2017).

Humphrey, W. S. (1996) *The Personal Software Process*. Essex: Addison-Wesley Professional.

Humphrey, W. S. (1999) *The Team Software Process*. Essex: Addison-Wesley Professional.

Hurry, J. (2014) 'Research Methods: Postgraduate Study in Educational and Social Research by Distance Learning Approaches to Educational Research Lecture Pack'.

London: University of London. doi: 10.1177/1468794107071408.

Jacobson, I. *et al.* (2012) *The essence of software engineering*. 1st edn. Addison-Wesley. doi: 10.1145/2380656.2380670.

Jacobson, I., Spence, I. and Ng, P. (2013) 'Agile and SEMAT - Perfect Partners', *Communications of the ACM*, November, pp. 53–59. doi: 10.1145/2524713.2524723.

Jones, C. (2006) 'Social and technical reasons for software project failures', *CrossTalk*, 19(6), pp. 4–9. Available at: http://nas.uhcl.edu/Boetticher/SWEN5430/WhySoftwareProjectsFail_CapersJones.pdf (Accessed: 15 January 2014).

Kappelman, L. a., McKeeman, R. and Zhang, L. (2006) 'Early Warning Signs of it Project Failure: The Dominant Dozen', *Information Systems Management*, 23(4), pp. 31–36. doi: 10.1201/1078.10580530/46352.23.4.20060901/95110.4.

Karch, E. (2011) *The Software Crisis: A Brief Look at How Rework Shaped the Evolution of Software Methodolgies*, *Microsoft Developer*. MSDN Blogs. Available at: https://blogs.msdn.microsoft.com/karchworld_identity/2011/04/04/the-software-crisis-a-brief-look-at-how-rework-shaped-the-evolution-of-software-methodolgies/ (Accessed: 23 May 2017).

Kaur, R. and Sengupta, J. (2011) 'Software Process Models and Analysis on Failure of Software Development Projects', *International Journal of Scientific & Engineering Research*, 2(2), pp. 1–4.

Keele, R. (2010) 'Quantitative versus qualitative research, or both?', in Sibley, A. and Donnelly, P. (eds) *Nursing Research and Evidence-Based Practice*. Sudbury: Jones & Bartlett Learning.

Kessler, G. P. (2001) 'Why technical projects fail: Avoiding disaster', *Inside Project Management*, 1(2), pp. 1–6. Available at: http://people.virginia.edu/~gpk2n/tpm018.pdf (Accessed: 25 January 2014).

Knobel, R. (2009) '13 software error, leading to catastrophes', *Tages Anzeiger*. Available at: http://www.tagesanzeiger.ch/digital/computer/13-softwarefehler-die-zu-katastrophen-fuehrten/story/21703807.

Kothari, C. (2004) *Research methodology: methods and techniques*. 2nd edn. Edited by N. A. International. Jaipur: New Age International Publishers.

Krigsman, M. (2012a) *Who's accountable for IT failure? (part one)*, *ZDNet*. Available at: http://www.zdnet.com/blog/projectfailures/whos-accountable-for-it-failure-part-one/15451 (Accessed: 23 September 2016).

Krigsman, M. (2012b) *Who's accountable for IT failure? (part two)*, *ZDNet*. Available at: http://www.zdnet.com/blog/projectfailures/whos-accountable-for-it-failure-part-two/15486 (Accessed: 23 September 2016).

Lapham, M. A. and Woody, Cc. C. (2006) *Sustaining Software-Intensive Systems*. Pittsburgh. Available at: https://studylib.net/doc/8139629/sustaining-software-intensive-systems.

Lee, S. *et al.* (2002) 'Perception gaps between IS academics and IS practitioners: an exploratory study', *ScienceDirect: Information & Management*, 40(1), pp. 51–61. doi: 10.1016/S0378-7206(01)00132-X.

Leedy, P. D. and Ormrod, J. E. (2009) *Practical Research: Planning and Design*. 9th edn. Edited by C. Robb. Merrill Publishing Company Inc.

de Leeuw, E. D., Hox, J. J. and Dillman, D. A. (eds) (2008) *International Handbook of Survey Methodology*. 1st edn. East Sussex: Psychology Press. doi: 10.4324/9780203843123.

Lientz, B. and Rea, K. (2003) *International Project Management*. 1st edn. San Diego, California: Routledge.

Lock, D. (2007) *Project Management*. 9th edn. Aldershot: Gower.

Lockyer, S. (2004) *Coding Qualitative Data, In The Sage Encyclopedia of Social Science Research Methods*. Edited by M. S. Lewis-Beck, A. Bryman, and T. F. Liao. Thousand Oaks: SAGE.

Macdonald, S. and Headlam, N. (2008) *Research methods handbook : introductory guide to research methods for social research*. Manchester: Centre for Local Economic Strategies.

Malan, R. and Bredemeyer, D. (2001) 'Defining non-functional requirements'. Bloomington, p. 8. Available at: http://www.bredemeyer.com/pdf_files/NonFunctReq.PDF (Accessed: 10 August 2013).

May, E. and Zimmer, B. (1996) 'The evolutionary development model for software', *Hewlett Packard Journal*, Article 4(August), pp. 1–8. Available at: http://www.hpl.hp.com/hpjournal/96aug/aug96a4.pdf (Accessed: 20 November 2013).

May, L. J. (1998) 'Major causes of software project failures', *CrossTalk: The Journal of Defense Software Engineering*, (July). Available at: http://static1.1.sqspcdn.com/static/f/702523/9173178/1288308435973/199807-May.pdf?token=4xs0aYCXYkZL1YX14TIvRpBeWss%3D (Accessed: 15 January 2014).

McCrindle, M. (2010) *The ABC of XYZ: Understanding the Global Generations*. Sydney: University ofNew South Wales Press Ltd.

McDonald, M. P. (2012) *McKinsey Report Highlights Failure of Large Projects: why it is better to be small, particularly in IT, Gartner Blogs*. Available at: http://blogs.gartner.com/mark_mcdonald/2012/10/29/mckinsey-report-highlights-failure-of-large-projects-why-it-is-better-to-be-small-particularly-in-it/.

Mcilroy, M. D. (1969) *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee*. Garmisch. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Software+Engineering:+Report+on+a+Conference+sponsored+by+the+NATO+Science+Committee#1 (Accessed: 11 November 2013).

Melton, T. (2007) *Project management toolkit: the basics for project success*. 2nd edn. Jordan Hill: Butterworth-Heinemann.

Moe, R. (2017) 'Academic Research Foundations: Quantitative'. Lynda.com. Available at: https://www.lynda.com/Education-Elearning-tutorials/Academic-Research-Foundations-Quantitative/553499-2.html?srchtrk=index%3A4%0Alinktypeid%3A2%0Aq%3Aresearch%0Apage%3A1%0As%3Arelevance%0Asa%3Atrue%0Aproducttypeid%3A2.

Morris, S. C. (2012) 'Why the evolution of humans is inevitable'. Cambridge: SCI.

Nawi, H. S. A., Rahma, A. A. and Ibrahim, O. (2014) 'Government ICT Project Failure Factors: Project Stakeholders' Views', *Journal of Information Systems Research and Innovation*, 2, p. 76.

Nelson, R. R. (2005) 'Project retrospectives: Evaluating project success, failure, and everything in between', *MIS Quarterly Executive*, 4(3), pp. 361–372. Available at: http://www2.commerce.virginia.edu/cmit/Research/MISQE 9-05.pdf.

Neuman, W. L. (2013) *Social Research Methods: Qualitative and Quantitative Approaches*. 7th edn. Harlow: Pearson Education Limited. doi: 10.2307/3211488.

Neville, C. (2007) *Introduction to Research and Research Methods*. Bradford: University of Bradford, School of Management.

Oracle Corporation (2011) 'Why Projects Fail : Avoiding the Classic Pitfalls'. Oracle Corporation.

Ouchi, F. (2004) 'A Literature Review on the Use of Expert Opinion in Probabilistic Risk Analysis', *Policy Research Working Paper*, 3201(February), p. 17. doi: 10.1596/1813-9450-3201.

Oxford University Press (2017a) *Definition of Quality*, *Oxford Learner's Dctionaries*. Available at: http://oald8.oxfordlearnersdictionaries.com/dictionary/quality_1 (Accessed: 11 February 2017).

Oxford University Press (2017b) *Oxford Dictionary*, *Oxford Dictionary - Online*. Available at: https://en.oxforddictionaries.com/ (Accessed: 12 February 2017).

Peter, I. (2004) *The Browser wars*, *The Internet History Project*. Available at: http://www.nethistory.info/History of the Internet/browserwars.html (Accessed: 23 February 2017).

Pitt-Catsouphes, M., Matz-Costa, C. and Besen, E. (2009) 'Age & Generations : Understanding Experiences at the Workplace', *Research Highlight*, 6(3), pp. 1–43.

Poynter, R. (2016) *How long, unwanted surveys hurt the market research industry*, *VisionCritical*. Available at: https://www.visioncritical.com/unwanted-surveys/ (Accessed: 20 May 2017).

Quelhas, A. *et al.* (2011) *Biases in questionnaire construction : how much do they influence the answers given ?* Porto. Available at: http://medicina.med.up.pt/im/trabalhos_10_11/Sites/Turma21/Protocolo Final.pdf.

Rai, K., Madan, L. and Anand, K. (2014) 'Software crisis', *International Journal of Innovative Research in Technology*, 1(11), pp. 95–102.

Rajkumar, G. and Alagarsamy, K. (2013) 'The most common factors for the failure of

Software Development Projects', *The International Journal of Computer Science & Applications*, 1(11), pp. 74–77.

Ralph, P. (2013) 'Possible Core Theories for Software Engineering', in *2013 2nd SEMAT Workshop on a General Theory of Software Engineering*. Stockholm: IEEE, pp. 35–38. doi: 10.1109/GTSE.2013.6613868.

Reichheld, F. (2006) *The Ultimate Question: Driving Good Profits and True Growth*. Boston: Harvard Business School Press.

Rosqvist, T., Koskela, M. and Harju, H. (2003) 'Software quality evaluation based on expert judgement', *Software Quality Journal*, 11(1), pp. 39–55. Available at: isi:000182660400005.

Rozin, P. and Royzman, E. B. (2001) 'Negativity Bias, Negativity Dominance, and Contagion', *Personality and Social Psychology Review*, 5(4), pp. 296–320. doi: 10.1207/S15327957PSPR0504.

Sage (2004) *Coding*. 1st edn, *The Sage Encyclopedia of Social Science Research Methods*. 1st edn. Edited by M. S. Lewis,-Beck, A. Bryman, and T. F. Liao. Thousand Oaks: SAGE. Available at: http://sk.sagepub.com/reference/socialscience/n128.xml?term=coding.

Saldaña, J. (2010) *The Coding Manual for Qualitative Researchers*. Los Angeles: SAGE Publications Ltd.

Sale, J. E. M., Lohfeld, L. H. and Brazil, K. (2002) 'Revisiting the quantitative-qualitative debate: Implications for mixed-methods research', *Quality and Quantity*, 36(1), pp. 43–53. doi: 10.1023/A:1014301607592.

Sandoval (2016) *Abstract & Concrete Perception (Theory), Cognitive Type*. Available at: http://cognitivetype.com/2016/01/25/abstract-concrete-perception-theory/ (Accessed: 26 July 2017).

Santayana, G. (2011) *The Life of Reason: Introduction and Reason in Common Sense*. Edited by M. S. Wokeck and M. A. Coleman. Cambridge: MIT Press.

Schmidt, R. *et al.* (2001) 'Identifying software project risks: an international Delphi study', *Journal of management information systems*, 17(4), pp. 5–35.

Serrador, P. and Pinto, J. K. (2015) 'Does Agile work? - A quantitative analysis of agile project success', *International Journal of Project Management*. Elsevier Ltd. APM and IPMA., 33(5), pp. 1040–1051. doi: 10.1016/j.ijproman.2015.01.006.

Shriver, R. (2012) *Agile Cloud Development: The Future of Software, The Virtualization Practice: Virtualization & Cloud Computing News, Resources, and Analysis*. Available at: http://www.virtualizationpractice.com/agile-cloud-development-the-future-of-software-16226/ (Accessed: 14 February 2017).

Sikwane, R. (2013) *Tweet, twitter and generic words, IP Ensight*. Available at: https://www.ensafrica.com/news/Tweet-twitter-and-generic-words?Id=1225&STitle=IP ENSight (Accessed: 25 July 2017).

Skjong, R. and Wentworth, B. (2000) 'Expert Judgement and Risk Perception', *Det Norske Veritas*, pp. 1–8. Available at: http://research.dnv.com/skj/Papers/SkjWen.pdf.

Smith, J. (2002) 'A South African perspective on Information Technology software project failure', in *African Rhythm Project Management Conference*. Johannesburg, South Africa.

Software Engineering Institute (2017) *Watts Humphrey: An Outrageous Commitment, A Lifelong Mission, Software Engineering Institute Website*. Available at: http://www.sei.cmu.edu/watts/ (Accessed: 23 January 2017).

Sommerville, I. (2011) *Software Engineering*. 9th edn. Edited by M. Horton et al. Boston: Addison-Wesley.

Summers, P. (2015) '98 . 8 %! Is project failure acceptable and inevitable?', in *IRNOP*. London: Barlett School.

Taherdoost, H. and Keshavarzsaleh, A. (2015a) 'A Theoretical Review on IT Project Success/Failure Factors and Evaluating the Associated Risks', in *14th International Conference on Telecommunications and Informatics*. Sliema, Malta, pp. 80–88. doi: 10.13140/RG.2.1.3214.9206.

Taherdoost, H. and Keshavarzsaleh, A. (2015b) 'How to Lead to Sustainable and Successful IT Project Management ? Propose 5Ps Guideline', *International Journal of Advanced Computer Science and Information Technology*, 4(1), pp. 14–37.

Tedre, M. (2015) *The Science of Computing: Shaping a Discipline*. Edited by C. & Hall. Baco Raton, Florida: CRC Press.

The Standish Group (1994) *The Chaos Report (1994)*. Boston.

The Standish Group (1995) *Chaos (1995)*. Boston.

The Standish Group (1999) *CHAOS : A Recipe for Success*.

The Standish Group (2001) *Extreme chaos (2001)*. Boston.

The Standish Group (2011) *Chaos manifesto: The lwas of Chaos and the Chaos 100 Best PM Practices*.

The Standish Group (2012) *The CHAOS Manifesto 2012: The Year of the Executive Sponsor*. Boston.

Thejendra, B. . (2014) *Disaster Recovery and Business Continuity: A Quick Guide for Organisations and Business Managers*. 3rd edn. Cambridgeshire: IT Governance Limited.

Thomas, G. and Fernández, W. (2008) 'Success in IT projects: A matter of definition?', *International Journal of Project Management*, 26(7), pp. 733–742. doi: 10.1016/j.ijproman.2008.06.003.

Verner, J., Sampson, J. and Cerpa, N. (2008) 'What factors lead to software project failure ?', in *2008 Second International Conference on Research Challenges in Information Science*. Marrakech: IEEE. doi: 10.1109/RCIS.2008.4632095.

Visitacion, M., Grant, T. and Brown, R. (2012) *Govern Your Lean And Agile Adoption, Forrester*. Available at: http://www.forrester.com/Govern+Your+Lean+And+Agile+Adoption/fulltext/-/E-RES74721 (Accessed: 5 February 2017).

Vosloo, J. J. (2014) *A sport management programme for educator training in accordance with the diverse needs of South African schools*. North-West University.

Weaver, P. (2007) 'The Origins of Modern Project Management', in *Fourth Annual PMI College of Scheduling Conference*. Vancouver: Mosaic Project Services Pty Ltd.

Welman, K. *et al.* (2005) *Research Methodology*. 3rd edn. Cape Town: Oxford University Press.

Westland, J. (2007) *The Project Management Life Cycle: A Complete Step-by-step Methodology for Initiating, Planning, Executing & Closing a Project Successfully*. Philadelphia: Kogan Page Publishers.

Wilson, I. M. and Stern, R. D. (2001) *Approaches to the Analysis of Survey Data, Biometrics Advisory and Support Service*. Reading.

Wirsing, M. (2004) *Report on the EU/NSF Strategic Workshop on Engineering Software-Intensive Systems*. Edinburgh.

Yilmaz, M., O'Connor, R. V. and Clarke, P. (2012) 'A Systematic Approach to the Comparison of Roles in the Software Development Processes', in Mas, A. et al. (eds) *Software Process Improvement and Capability Determination*. Palma: Springer, pp. 198–209. doi: 10.1007/978-3-642-30439-2.

Young, R. R., Brady, S. M. and Nagle, D. C. (2009) *How to Save a Failing Project: Chaos to Control*. Vienna: Berrett-Koehler Publishers.

Young, T. L. (2007) *The Handbook of Project Management: A Practical Guide to Effective Policies, techniques and processes*. 2nd edn. Philadelphia: Kogan Page Publishers.

Yudkowsky, E. (2008) 'Cognitive Biases Potentially Affecting Judgment of Global Risks', in Bostrom, N. and Ćirković, M. M. (eds) *Global catastrophic risks*. New York: Oxford University Press, pp. 91–119.

Annexure A: Sample size calculation reasoning

The calculation defined in Table 28 (*Sample size algorithm*) below, allows a researcher to calculate the required sample size, given several attributes. These include the margin of error (e), confidence interval (z), estimated population size (N) and response distribution (p).

The population size is referred to as the total number of relevant individuals that could be used in the study. In terms of this research, the population size for the overall software engineering industry in South Africa is unknown. There are however several suggestions that have been made in the past, all suggesting different numbers. These include a total of 30000, 50000 for the Gauteng area and around a quarter of that in the Western Cape, 6000 for Kwazulu Natal, and so it continues. Until now, no real answer could be found as most of the attempts aimed at getting a glimpse of the real number failed to collect real evidence. As such, a researcher in the field of software engineering in South Africa is forced to assume a total population size and once done, struggles to collect the number specified as a sample. There are several suggestions that in calculating the sample size, a population larger than 20,000 does not yield a lot of difference, and researchers settle on using this number when an assumption about the population can not be made.

The research this annexure forms part of is no different. No real assumption can be made about the total number of individuals in the local South African software industry, as none of them can be validated in any way. As such, the researcher set out to understand the differences in numbers between different variables used in calculating a sample size and opted to settle on a number that could be considered appropriate for a populations size, whilst pushing the boundary to define the largest sample size that could be studied against.

For this exercise, the highest estimated population size was capped at 500,000. This population was reduced by 25% at each level to create a list of population sizes to be

calculated. The result was a total of 52 different population sizes ranging from 500,000 to 1 visible as the first column in Table 29 (*Sample calculation*) below.

In addition to the estimated population size, the margin of error can severely impact the sample size. The lower the margin of error, the higher the sample size would be. Margin of error refers to the margin of error that could be tolerated by a study and still be representative. As such, the range of 1% to 10% was included. Table 29 (*Sample calculation*) below provided a column for each margin of error, ultimately creating a cross-tabulation of estimated population against margin of error.

Confidence level refers to the amount of uncertainty the study can tolerate. This measure represents the certainty that the sample accurately reflects the population within the margin of error. It was argued that in terms of the uncertainty present in estimating the total number of individuals in the general South African software engineering population, nothing less than a 95% confidence level should be entertained. A lower confidence level would reduce the sample size further and as such reduce the overall generalization of the sample.

The uncertainty associated to the population and the answers expected from a sample made calculating the response distribution difficult. It was however noted that setting the response distribution to 50% would provide the largest sample size and as such could be considered the most conservative assumption.

In calculating the required sample size, the following questions were created to provide guidance:

> *Q1: How large a sample should be selected to provide 95% (**z** = 1.96) confidence interval with a margin of error (**e**) of 4%,5% or 7%, assuming a total population (**N**) of 500,000 with a normal distribution (**p**) of 50%?*

*Q2: How large a sample should be selected to provide 95% (**z** = 1.96) confidence interval with a margin of error (**e**) of 4%,5% or 7%, assuming a total population (**N**) of 250,000 with a normal distribution (**p**) of 50%?*

*Q3: How large a sample should be selected to provide 95% (**z** = 1.96) confidence interval with a margin of error (**e**) of 4%,5% or 7%, assuming a total population (**N**) of 100,000 with a normal distribution (**p**) of 50%?*

*Q4: How large a sample should be selected to provide 95% (**z** = 1.96) confidence interval with a margin of error (**e**) of 4%,5% or 7%, assuming a total population (**N**) of 20,000 with a normal distribution (**p**) of 50%?*

Table 27 Sample size calculation attributes

Attribute	**Symbol**	**%**	**Q1**	**Q2**	**Q3**	**Q4**
Margin of error (*e = 0.04)*	*e*	4%	0.04	0.04	0.04	0.04
Margin of error (*e = 0.05)*	*e*	5%	0.05	0.05	0.05	0.05
Margin of error (*e = 0.07)*	*e*	7%	0.07	0.07	0.07	0.07
Confidence interval	*z*	95%	1.96	1.96	1.96	1.96
Planning value (*Estimated)*	*N*	100%	500,000	250,000	100,000	20,000
Response distribution	*p*	50%	0.5	0.5	0.5	0.5

The standard sample size algorithm was used to determine the required sample size:

Table 28 Sample size algorithm

$$\textbf{Sample size} = \frac{\frac{z^2 \times p(1-p)}{e^2}}{1 + \left(\frac{z^2 \times p(1-p)}{e^2 N}\right)}$$

The following table provides a cross-tabulation of estimated population against margin of error. The estimated population if decreased by 24% for each instance, but the numbers, 250 000, 200 000, 150 000, 100 000, 25 000, 20 000 and 10 000 were injected for illustration.

Table 29 Sample calculation

Estimated population	Margin of error									
	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
n: 500000	9423	2390	1065	600	384	267	196	150	119	96
n: 375000	9364	2386	1064	599	384	267	196	150	119	96
n: 281250	9287	2381	1063	599	384	267	196	150	119	96
n: 250000	9249	2378	1063	599	384	266	196	150	119	96
n: 210938	9186	2374	1062	599	383	266	196	150	119	96
n: 200000	9164	2373	1061	598	383	266	196	150	118	96
n: 158204	9054	2365	1060	598	383	266	196	150	118	96
n: 150000	9026	2363	1060	598	383	266	196	150	118	96
n: 118653	8885	2353	1058	597	383	266	196	150	118	96
n: 100000	8762	2345	1056	597	383	266	196	150	118	96
n: 88990	8668	2338	1054	596	383	266	196	150	118	96
n: 66742	8396	2318	1050	595	382	266	195	150	118	96
n: 50057	8058	2291	1045	593	381	265	195	150	118	96
n: 37543	7648	2257	1038	591	380	265	195	149	118	96
n: 28157	7161	2212	1028	588	379	264	195	149	118	96
n: 25000	6939	2191	1023	586	378	264	194	149	118	96
n: 21118	6602	2156	1016	584	377	263	194	149	118	96
n: 20000	6488	2144	1013	583	377	263	194	149	118	96
n: 15839	5979	2085	1000	578	375	262	194	149	118	95
n: 11879	5310	1997	979	571	372	261	193	148	117	95
n: 10000	4899	1936	964	566	370	260	192	148	117	95
n: 8909	4622	1891	953	562	368	259	192	148	117	95
n: 6682	3940	1766	920	551	363	257	190	147	117	95
n: 5012	3293	1623	880	536	357	253	189	146	116	94
n: 3759	2701	1465	831	518	349	249	186	144	115	94

n: 2819	2179	1297	774	495	338	244	183	142	114	93
n: 2115	1733	1124	709	468	325	237	179	140	112	92
n: 1586	1361	955	638	435	309	228	174	137	110	91
n: 1190	1058	795	562	399	290	218	168	133	108	89
n: 892	816	650	486	359	269	205	161	128	105	87
n: 669	625	523	411	316	244	191	152	123	101	84
n: 502	477	415	341	273	218	174	141	116	96	81
n: 377	362	325	278	231	190	156	129	107	90	77
n: 283	274	253	223	192	163	137	116	98	83	72
n: 212	207	195	177	156	136	118	102	88	76	66
n: 159	156	149	138	126	112	100	88	77	68	60
n: 120	118	113	107	99	91	82	74	66	59	53
n: 90	88	86	82	78	72	67	61	56	51	46
n: 67	67	65	63	60	57	54	50	46	43	39
n: 51	50	49	48	46	44	42	40	38	35	33
n: 38	38	37	36	35	34	33	32	30	29	27
n: 29	28	28	28	27	26	26	25	24	23	22
n: 22	21	21	21	20	20	20	19	19	18	17
n: 16	16	16	16	15	15	15	15	14	14	14
n: 12	12	12	12	12	12	11	11	11	11	11
n: 9	9	9	9	9	9	9	9	8	8	8
n: 7	7	7	7	7	7	7	6	6	6	6
n: 6	5	5	5	5	5	5	5	5	5	5
n: 4	4	4	4	4	4	4	4	4	4	4
n: 3	3	3	3	3	3	3	3	3	3	3
n: 1	1	1	1	1	1	1	1	1	1	1

Conclusion

Figure 21 (*Variable margin of error (e) versus estimated population (n)*) below demonstrates the effect of changing the margin of error per estimated population. The diagram suggests that there is little impact on the number of individuals required for a sample, regardless of the estimated population, but only when the margin of error is higher than about 4.5%. For example, an estimated population of 500 000 or 20 000 would yield a total sample size of 96 individuals when a margin of error of 10% was used. Similarly, when using 5%, an estimated population of 500 000 and 100 000 would yield between 383 and 384 individuals. However, any margin of error below 4.5% would yield major differences in the size of the sample, for example, an estimated population of 20 000 would yield a sample size of 1013 and an estimated population of 500 000 would yield a sample size of 1065, both at 3% margin of error. Similarly, when considering 1% as the margin of error, an estimated population size of 20 000 would yield 6488 as a sample size and a population of 500 000 would yield 9423 as a sample size. As such, it was found that for a population of 90 000 to 500 000, the sample size remains relatively static for each margin of error of 5% and above, but exponentially increases for each margin of error of 4% and below the larger the population grows. This suggested that settling on 20 000 as an estimated population size is only relevant should the researcher introduce a margin of error of 5% and above.

Table 30 (*Sample size question results*) below provides the outcomes of the questions posed in calculating the desired sample size. The sizes calculated below are illustrated in Figure 22 (*Sample Size calculation distribution*) below. This illustration demonstrates the s-curve associated to the sample size calculation and shows that variances in size could still be found between an estimated population of 20 000 and 66 000 when a margin of error of 4% was selected. The illustration however reveals that when selecting a population of 66 000 and above, little variance in the sample size would be experienced, regardless of the margin of error chosen. As such, the research planned to incorporate the

standard 5% margin of error, but to include an estimated population size of 250 000 to ensure that the research data could be generalised.

Table 30 Sample size question results

Estimated population	Margin of error		
	4%	**5%**	**7%**
n: 500000	600	384	196
n: 250000	599	384	196
n: 100000	597	383	196
n: 20000	583	377	194

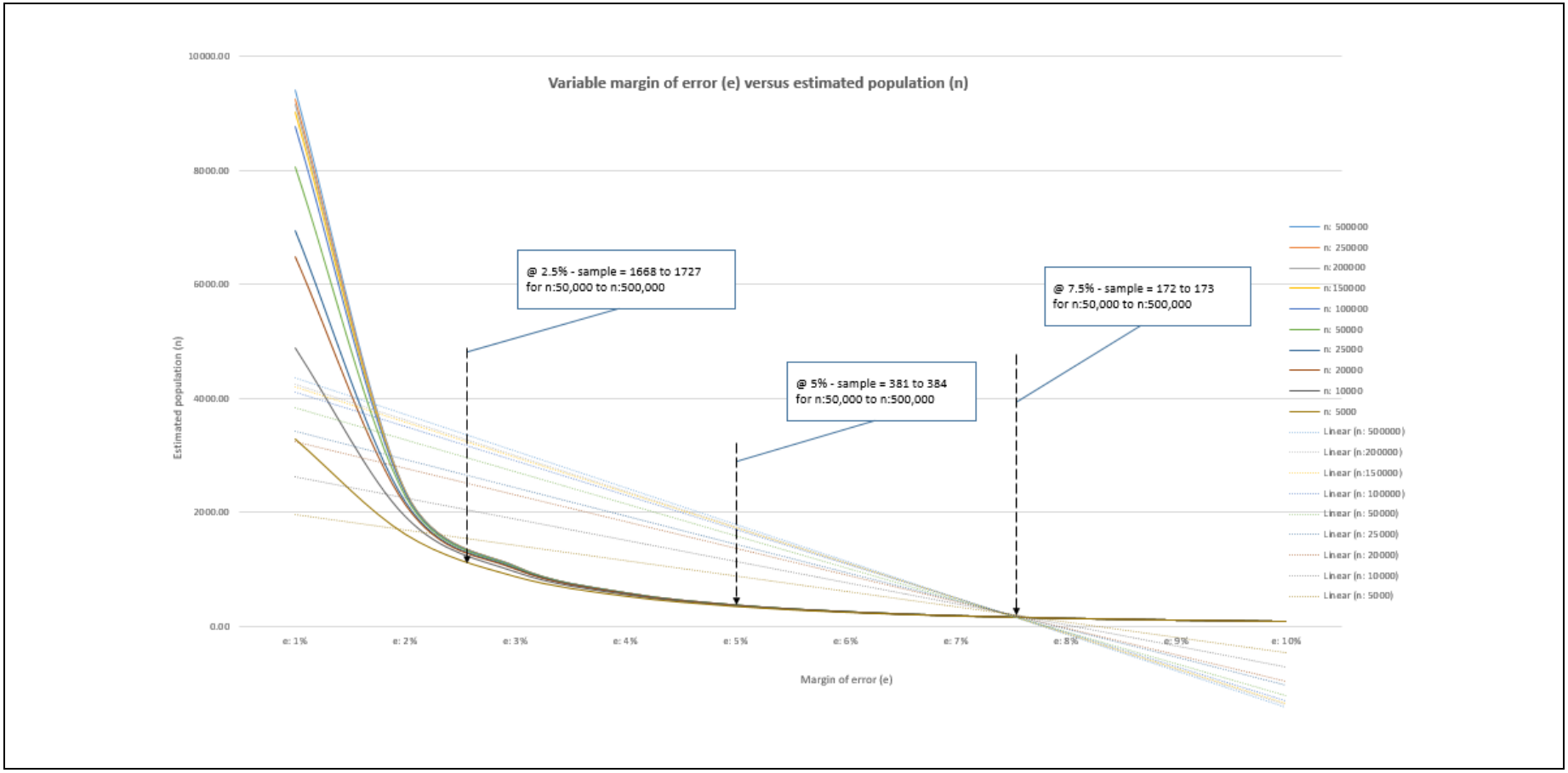


Figure 21 Variable margin of error (e) versus estimated population (n)

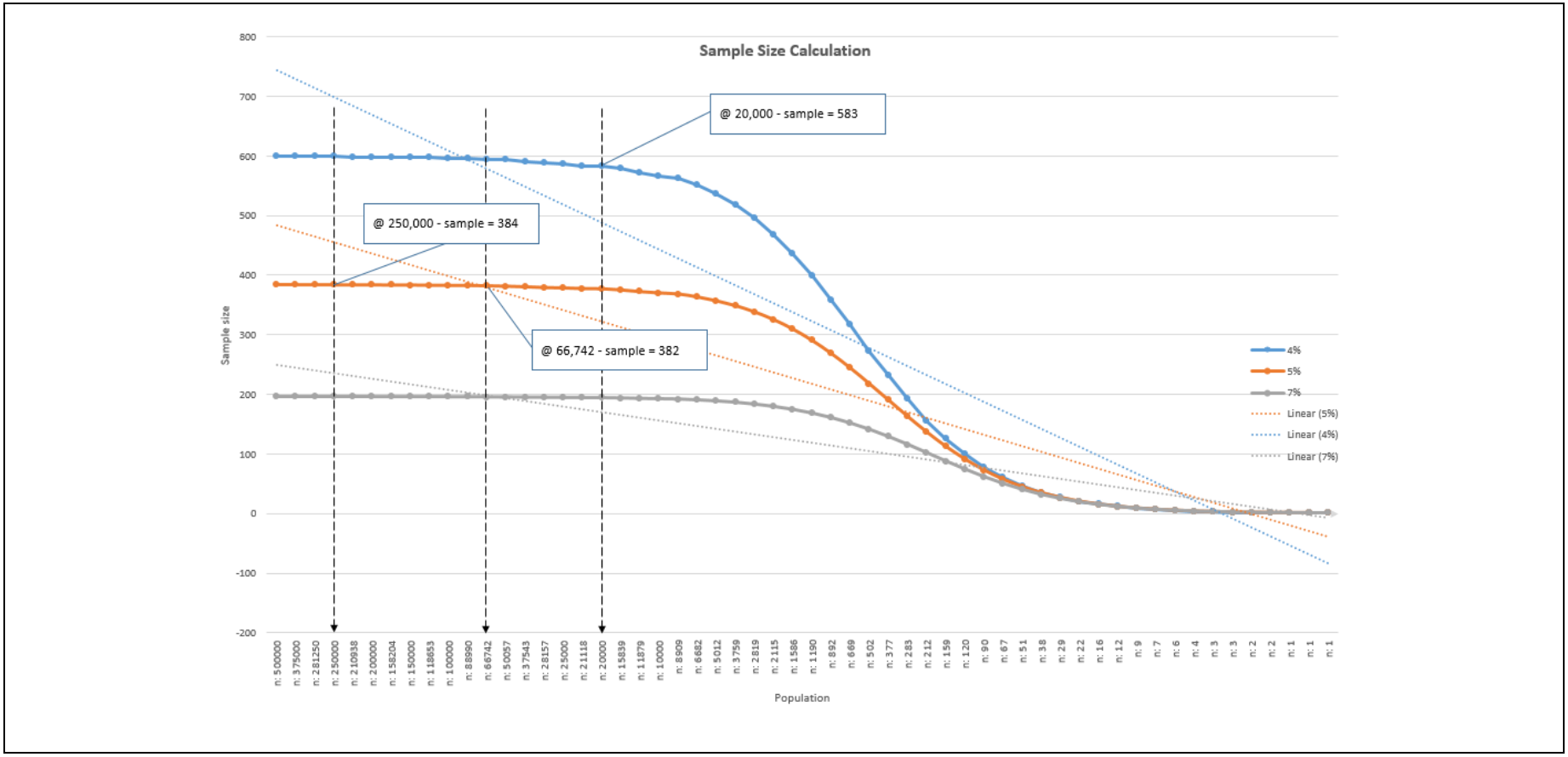


Figure 22 Sample Size calculation distribution

Annexure B: Literature review summary on reasons for software project failure

Research Categorisation	Political Impacts		Internal Variability			
McKinsey – Oxford Categorisation	Inadequate focus		Problems with system content		Skills problems	Problems in execution
Al-Ahmad et al. Categorisation	Top management factors	Organizational factors	Project management factors	Complexity or size factors	Technology factors	Process factors
Commercial pressures or cost/value benefit	(Charette, 2005; Henderson, 2006)					
Projects are approved based on intuitive, personal or political reasons, instead of conducting the necessary risk assessment		(Lock, 2007)				
External factors, such as vendor and regulatory issues	(Thejendra, 2014)					
Senior management negatively impacted project		(Verner, Sampson and Cerpa, 2008)				
Lack of top management support or commitment to the project	(Kappelman, McKeeman and Zhang, 2006)					
No project communications plan or resources devoted to managing and communicating project expectations		(Kappelman, McKeeman and Zhang, 2006)				
Project team members are overscheduled		(Kappelman, McKeeman and Zhang, 2006)				
Approved project budget less than budget estimated by the project team	(Kappelman, McKeeman and Zhang, 2006)					
Unstable organization environment (such as changes in senior management or restructuring)		(Kappelman, McKeeman and Zhang, 2006)				
PM not given full authority to manage project		(Verner, Sampson and Cerpa, 2008)				
Lack of or inadequate change management	(Young, 2007)					
Ineffective change control because of constantly changing business and user requirements			(Charvat, 2003; Frederiksz, Hedeman and Heemst, 2010)			
No buy-in or support from the project stakeholders		(Westland, 2007)				
No executive ownership	(Frederiksz, Hedeman and Heemst, 2010)					
Little support from senior management		(Hayes, 2002; Frederiksz, Hedeman and Heemst, 2010)				
Misalignment within the internal organisation						(Grey, 2011)
Stakeholder concerns and interests are neglected			(Lock, 2007)			
Sponsor commitment did not last through project		(Verner, Sampson and Cerpa, 2008)				
Project stakeholders have not been interviewed for project requirements					(Kappelman, McKeeman and Zhang, 2006)	
Key project stakeholders do not participate in major review meetings					(Kappelman, McKeeman and Zhang, 2006)	
Subject matter experts are overscheduled: retain all prior duties yet expected to provide substantial participation to the project		(Kappelman, McKeeman and Zhang, 2006)				
Key stakeholders do not review and sign off deliverables on a timely basis		(Kappelman, McKeeman and Zhang, 2006)				
Project stakeholder decision delays have caused due dates to be missed		(Kappelman, McKeeman and Zhang, 2006)				
No written commitment for the project outside of the project team		(Kappelman, McKeeman and Zhang, 2006)				
Key stakeholders have not signed the project charter	(Kappelman, McKeeman and Zhang, 2006)					
Users or technical support team feel threatened by a project to replace their legacy system		(Kappelman, McKeeman and Zhang, 2006)				
Team commitment due to conflicting interests or cultures		(Al-Ahmad *et al.*, 2009; Krigsman, 2012a)				
Organisational politics and culture clashes		(Thejendra, 2014)				

Personal interests are not taken into consideration		(Lientz and Rea, 2003)				
Developers not involved in estimates Project underestimated						(Verner, Sampson and Cerpa, 2008)
Aggressive schedule affected team motivation					(Verner, Sampson and Cerpa, 2008)	
Lack of top management support		(Kappelman, McKeeman and Zhang, 2006)				
Project team members have weak commitment to the project scope and schedule		(Kappelman, McKeeman and Zhang, 2006)				
Project resources have been assigned to a higher priority project		(Kappelman, McKeeman and Zhang, 2006)				
Team members have undefined roles and responsibilities		(Kappelman, McKeeman and Zhang, 2006)				
Key team member turnover after project kick-off		(Kappelman, McKeeman and Zhang, 2006)				
Poor user input					(May, 1998)	
Failure to gain user commitment or inadequate user involvement					(Schmidt *et al.*, 2001; Al-Ahmad *et al.*, 2009; Kaur and Sengupta, 2011; Rajkumar and Alagarsamy, 2013)	
No user involvement and commitment from project initiation					(Young, 2007; Frederiksz, Hedeman and Heemst, 2010)	
Low level of customers/user involvement		(Verner, Sampson and Cerpa, 2008)				
Involved customers/users did not stay through project		(Verner, Sampson and Cerpa, 2008)				
Customers/users not involved in schedule estimates		(Verner, Sampson and Cerpa, 2008)				
High customer/user turnover		(Verner, Sampson and Cerpa, 2008)				
Customers/users did not make enough time for requirements gathering		(Verner, Sampson and Cerpa, 2008)				
Users are not willing to cooperate		(Kappelman, McKeeman and Zhang, 2006)				
Lack of executive management commitment with no-one to champion the project.	(Schmidt *et al.*, 2001; Al-Ahmad *et al.*, 2009; Gulla, 2011; Krigsman, 2012a)					
Lack of business focus		(Henderson, 2006; Bloch, Blumberg and Laartz, 2012)				
Failure to align with constituents and stakeholders and a late realization of warning signals with a refusal to recognise/admit the project is in trouble. Responsive project planning					(May, 1998; Charette, 2005; Jones, 2006; Gulla, 2011; Oracle Corporation, 2011; Bloch, Blumberg and Laartz, 2012)	
Insufficient critical project communication					(May, 1998; Charette, 2005; Gulla, 2011; Oracle Corporation, 2011; Dalal and Chhillar, 2012)	
Incorrect and Optimistic Status Reporting with a lack of success and failure criteria					(Charette, 2005; Jones, 2006; Krigsman, 2012a)	
Communication breakdown among project stakeholders		(Kappelman, McKeeman and Zhang, 2006)				
Team misalignment due to improper communication resulting in a failure to act as a team					(Dorsey, 2005; Henderson, 2006; Kaur and Sengupta, 2011; Bloch, Blumberg and Laartz, 2012; Rajkumar and Alagarsamy, 2013)	
Business decisions made by senior management are commands to be executed, which results in problems and issues being addressed by team members because there is no open communication channel to senior management	(Thejendra, 2014)					
Conflict between user departments		(Schmidt *et al.*, 2001)				
Failure to manage end-user expectations					(Schmidt *et al.*, 2001)	
Poor communication					(Charvat, 2003; Westland, 2007)	
Difficulty in determining the input and output of the system				(Kappelman, McKeeman and Zhang, 2006)		
Large number of interfaces to other system required				(Kappelman, McKeeman and Zhang, 2006)		
Project involves implementing a custom or beta version of hardware or software				(Kappelman, McKeeman and		

				Zhang, 2006)		
Cultural conflict among organizations involved		(Kappelman, McKeeman and Zhang, 2006)				
No stakeholder investment or alignment or stakeholder conflict wrt. unrealistic and mismatched expectations		(May, 1998; Charette, 2005; Henderson, 2006; Bloch, Blumberg and Laartz, 2012; Krigsman, 2012b)				
Significant goal, scope, or schedule requirements change immediately after project kick-off					(Kappelman, McKeeman and Zhang, 2006)	
Inadequate Quality Control						(Jones, 2006; Kaur and Sengupta, 2011)
Inaccurate - unrealistically low bids and budget requests and an underestimation of time requirements					(May, 1998; Charette, 2005; Jones, 2006; Dalal and Chhillar, 2012; Rajkumar and Alagarsamy, 2013)	
Enthusiastic/optimistic estimations were taken literally and became execution time.					(Buckler, 2010)	(Buckler, 2010)
Testing time was sacrificed to avoid late delivery.					(Buckler, 2010)	
Cost and project schedule estimates are too optimistic and they are not revised as the project progresses and new information becomes available					(Charvat, 2003; Lock, 2007)	
Poor control during project delivery					(Melton, 2007)	
Poor quality control					(Young, 2007)	
No project status progress process					(Kappelman, McKeeman and Zhang, 2006)	
No project management methodology						(Kappelman, McKeeman and Zhang, 2006)
Major new risks are identified after the project kick-off					(Kappelman, McKeeman and Zhang, 2006)	
IT operations infrastructure and network infrastructure problems have major impact on project team productivity		(Kappelman, McKeeman and Zhang, 2006)				
Project manager did not control the project		(Verner, Sampson and Cerpa, 2008)				
Inappropriate development methodology						(Verner, Sampson and Cerpa, 2008)
Customer/user expectations not managed					(Verner, Sampson and Cerpa, 2008)	
Customers/users did not have realistic expectations					(Verner, Sampson and Cerpa, 2008)	
PM abandoned plan under pressure					(Verner, Sampson and Cerpa, 2008)	
No clear or unified vision and objectives		(Hayes, 2002:196);				
No clear and robust business case or project definition					(Melton, 2007; Frederiksz, Hedeman and Heemst, 2010)	
No real demand for the products that will be delivered by the project	(Young, 2007)					
Team conflicting interests		(Al-Ahmad *et al.*, 2009)				
Problems due to large numbers of customer/uses involved						(Verner, Sampson and Cerpa, 2008)
No change control process			(Kappelman, McKeeman and Zhang, 2006)			
Budget, schedule, scope, and quality all mandated from outside the project team	(Kappelman, McKeeman and Zhang, 2006)					
Change control not monitored effectively						(Verner, Sampson and Cerpa, 2008)
Poor management of scope			(Oracle Corporation, 2011)			
Inadequately dissected and poorly scoped projects			(Buckler, 2010)			
Project scope changes and scope creep			(Hallows, 1998; Charvat, 2003; Young, 2007)			
Unrealistic client and business expectations			(Hayes, 2002)			
Stakeholders have different and unrealistic expectations		(Young, 2007)				
Lack of change, risk, financial and performance management and measurement						(Charette, 2005; Gulla, 2011; Oracle Corporation, 2011)
Project team member(s) have low morale		(Kappelman, McKeeman and Zhang, 2006)				
Unrealistic or inaccurate scheduling using task time as schedule time			(Jones, 2006; Kaur and Sengupta, 2011; Bloch, Blumberg and Laartz, 2012; Rajkumar and Alagarsamy, 2013)			

Inadequate budgets and unrealistic timelines			(Thejendra, 2014)			
No defined checkpoints and structure			(Frederiksz, Hedeman and Heemst, 2010)			
No effective project plan			(Verner, Sampson and Cerpa, 2008)			
Project not replanned after requirements changes			(Verner, Sampson and Cerpa, 2008)			
No planning and estimation documentation			(Kappelman, McKeeman and Zhang, 2006)			
Incomplete project risk assessment is conducted			(Hitech Dimensions, 2002:7);			
No due diligence on vendor(s) and team members	(Kappelman, McKeeman and Zhang, 2006)					
No robust project delivery plan			(Melton, 2007:9);			
Risks not identified at start of the project			(Verner, Sampson, & Cerpa, 2008)			
Risks not incorporated into project plan			(Verner, Sampson, & Cerpa, 2008)			
Risks not reassessed, and controlled			(Verner, Sampson, & Cerpa, 2008)			
Insufficient project sizing			(Grey, 2011)			
Ineffective planning and task assignment and direction			(May, 1998; Dorsey, 2005; Henderson, 2006; Gulla, 2011; Rajkumar and Alagarsamy, 2013)			
No business case for the project	(Kappelman, McKeeman and Zhang, 2006)					
Inadequate impact analysis is done to determine the impact the project will have on the organisation and business processes	(Grey, 2011)					
Avoidance of, lack of or suitability of effective project management methodology						(Schmidt *et al.*, 2001; Dorsey, 2005; Al-Ahmad *et al.*, 2009; Gulla, 2011; Oracle Corporation, 2011)
No quality and acceptance criteria						(Frederiksz, Hedeman and Heemst, 2010)
Lack of delivering benefits and value to the organisation						(Melton, 2007)
No robust and understandable project scope			(Lock, 2007; Melton, 2007)			
Poor project scoping			(Grey, 2011)			
Poor requirements definition and gathering					(Westland, 2007; Young, Brady and Nagle, 2009; Frederiksz, Hedeman and Heemst, 2010; Grey, 2011)	
Deliverables to be produced are poorly defined					(Frederiksz, Hedeman and Heemst, 2010)	
No clear understanding of the user and technical requirements and what must be done					(Lock, 2007; Westland, 2007)	
Lack of Resources		(Rajkumar and Alagarsamy, 2013)				
More developers increase execution		(Buckler, 2010)				
Inappropriate team organisation		(Kaur and Sengupta, 2011)				
Inappropriate team structure or an improper definition of roles and responsibilities. Insufficient or inappropriate team staffing		(Schmidt *et al.*, 2001; Al-Ahmad *et al.*, 2009; Krigsman, 2012a)				
Poorly defined roles, authority and responsibilities		(Frederiksz, Hedeman and Heemst, 2010)				
Resource shortages (money, time, expertise)		(Hayes, 2002)				
Poor capacity planning			(Thejendra, 2014)			
Project not adequate staffed		(Verner, Sampson and Cerpa, 2008)				
PM changed during project		(Verner, Sampson and Cerpa, 2008)				
Staff added late to meet aggressive schedule		(Verner, Sampson and Cerpa, 2008)				
Staff not appreciated for working long hours		(Verner, Sampson and Cerpa, 2008)				
Staff not rewarded for working long hours		(Verner, Sampson and Cerpa, 2008)				
Schedule affected team members' lives		(Verner, Sampson and Cerpa, 2008)				
No reviews at end of each phase						(Verner, Sampson and Cerpa, 2008)
Poor requirement set due to badly defined or misunderstanding the user requirements or failed to appropriately extract requirements					(Schmidt *et al.*, 2001; Charette, 2005; Kaur and Sengupta, 2011; Rajkumar and Alagarsamy, 2013)	
Lack of management of changing user requirements and scope during execution			(Schmidt *et al.*, 2001; Henderson,			

			2006; Jones, 2006; Buckler, 2010; Bloch, Blumberg and Laartz, 2012)			
Objective and goals are unarticulated, vague or blurred and often change			(May, 1998; Schmidt *et al.*, 2001; Charette, 2005; Al-Ahmad *et al.*, 2009; Kaur and Sengupta, 2011; Bloch, Blumberg and Laartz, 2012; Dalal and Chhillar, 2012; Rajkumar and Alagarsamy, 2013)			
Unclear / misunderstood scope and objectives			(Schmidt *et al.*, 2001; Al-Ahmad *et al.*, 2009)			
Large and multifaceted with lots of integration points				(Schmidt *et al.*, 2001; Al-Ahmad *et al.*, 2009)		
Inability to handle the project's technical complexity				(Charette, 2005; Bloch, Blumberg and Laartz, 2012)		
Inadequate estimation techniques or estimated by non-developers or overly optimistic developers			(Dorsey, 2005; Buckler, 2010; Rajkumar and Alagarsamy, 2013)			
No clear methodology is defined and followed						(Westland, 2007)
In practice project management theory is not executed correctly						(Charvat, 2003)
Project managers have poor training			(Kappelman, McKeeman and Zhang, 2006)			
Project manager(s) have never managed a project of this scale before			(Kappelman, McKeeman and Zhang, 2006)			
Deliverable due dates missed during the first 10 percent of the project schedule			(Kappelman, McKeeman and Zhang, 2006)			
No contingency budget for known risks and rate of changes		(Kappelman, McKeeman and Zhang, 2006)				
Earned value systems not in place or used to control program			(Kappelman, McKeeman and Zhang, 2006)			
Poor or inadequate leadership		(Young, Brady and Nagle, 2009)				
Weak managers		(Thejendra, 2014)				
Insufficient organisational structure		(Thejendra, 2014)				
Insufficient utilisation of planning tools			(Grey, 2011)			
Insufficient attention is given to the management of cash flow and fund provision			(Lock, 2007)			
Project problems are identified to late			(Young, Brady and Nagle, 2009)			
Delivery decision not made with appropriate requirements info	(Verner, Sampson and Cerpa, 2008)					
Delivery date affected development process	(Verner, Sampson and Cerpa, 2008)					
Weak project manager			(Kappelman, McKeeman and Zhang, 2006)			
Project manager(s) cannot effectively lead the team and communicate with clients			(Kappelman, McKeeman and Zhang, 2006)			
Schedule deadline not reconciled to the project schedule			(Kappelman, McKeeman and Zhang, 2006)			
Early project delays are ignored — no revision to the overall project schedule			(Kappelman, McKeeman and Zhang, 2006)			
No project charter document at early stage of project			(Kappelman, McKeeman and Zhang, 2006)			
No risk analysis documentation and process			(Kappelman, McKeeman and Zhang, 2006)			
No performance and reliability requirements metrics tracking process			(Kappelman, McKeeman and Zhang, 2006)			
No or poor planning			(Hallows, 1998; Hayes, 2002)			
Plans are not used to guide the project or are not executed correctly			(Charvat, 2003)			
Inappropriate PMM is followed			(Charvat, 2003)			
Intended project management strategy is inappropriate						(Lock, 2007)
No documented milestone deliverables and due dates			(Kappelman, McKeeman and Zhang, 2006)			
Undefined project success criteria			(Kappelman, McKeeman and Zhang,			

			2006)			
Project managers are not given training for developing their project management skills which results in the same mistakes being repeated		(Charvat, 2003)				
PM had no input into estimates						(Verner, Sampson and Cerpa, 2008)
No method for requirements gathering						(Verner, Sampson and Cerpa, 2008)
Project did not have good requirements at any stage					(Verner, Sampson and Cerpa, 2008)	
Scope of project not well defined Scope changes during project					(Verner, Sampson and Cerpa, 2008)	
No single requirements repository for requirements Size of project affected requirements elicitation					(Verner, Sampson and Cerpa, 2008)	
Functional, performance, and reliability requirements and scope are not documented					(Kappelman, McKeeman and Zhang, 2006)	
Project sponsors lack experience		(Kaur and Sengupta, 2011)				
Poor technical designs and practices					(Thejendra, 2014)	
Too great a reliance on technology					(Hallows, 1998; Lientz and Rea, 2003)	
Lack of effective project management skills			(Schmidt *et al.*, 2001)			
Introduction of new or immature technology, skills do not match the job, lack of design, technical experience					(May, 1998; Schmidt *et al.*, 2001; Charette, 2005; Dorsey, 2005; Al-Ahmad *et al.*, 2009; Gulla, 2011; Bloch, Blumberg and Laartz, 2012; Dalal and Chhillar, 2012)	
Sloppy development practises						(May, 1998; Schmidt *et al.*, 2001; Charette, 2005; Dorsey, 2005; Al-Ahmad *et al.*, 2009; Gulla, 2011; Bloch, Blumberg and Laartz, 2012; Dalal and Chhillar, 2012)
Lack of technology experience					(Hayes, 2002)	
Insufficient evaluation of technology					(Grey, 2011)	
Team did not have adequate experience with tools and techniques					(Verner, Sampson and Cerpa, 2008)	
Project team members do not have required knowledge/skills		(Kappelman, McKeeman and Zhang, 2006)				
No team member experience with the chosen technology					(Kappelman, McKeeman and Zhang, 2006)	
Failure to gather requirement via joint application design					(Kappelman, McKeeman and Zhang, 2006)	
No documented analysis of business strategy alignment					(Kappelman, McKeeman and Zhang, 2006)	
Users cannot get involved because of lack of understanding of new system capabilities				(Kappelman, McKeeman and Zhang, 2006)		
Return on investment assessment is not performed	(Grey, 2011)					
Incorrect steps to reproduce and improper fault assignment or a general poor quality of testing					(Dorsey, 2005; Kaur and Sengupta, 2011; Dalal and Chhillar, 2012; Rajkumar and Alagarsamy, 2013)	
Poor testing of product developed					(Charvat, 2003)	
Inappropriate tooling					(Dorsey, 2005; Rajkumar and Alagarsamy, 2013)	
Incomplete Requirements					(The Standish Group, 1995; Smith, 2002)	
Lack of User Involvement			(The Standish Group, 1995; Smith, 2002)			
Lack of Resources		(The Standish Group, 1995; Smith, 2002)				
Unrealistic Expectations			(The Standish Group, 1995; Smith, 2002)			
Lack of Executive Support		(The Standish Group, 1995; Smith, 2002)				
Changing Requirements & specifications			(The Standish Group, 1995; Smith,			

			2002)			
Lack of Planning			(The Standish Group, 1995; Smith, 2002)			
Didn't Need It Any Longer		(The Standish Group, 1995; Smith, 2002)				
Lack of IT Management	(The Standish Group, 1995; Smith, 2002)					
Technology Illiteracy					(The Standish Group, 1995; Smith, 2002)	
Poor estimating or scheduling; this involves scoping the project and estimating the time and effort required			(Nelson, 2005)			
Ineffective stakeholder management; this involves identifying stakeholders and their influence and interest in the project and then managing their expectations accordingly			(Nelson, 2005)			
Insufficient risk management; this involves identifying and mitigating anything which may impact upon the project			(Nelson, 2005)			
Insufficient planning; this happens when project managers jump straight into delivery			(Nelson, 2005)			
Short changed quality assurance; this comes about when projects come under pressure and as a result testing and training are often cut			(Nelson, 2005)			

Annexure C: Participants information sheet

An investigation into the catalysts that fuel success or failure of software development initiatives in South Africa

My name is Tony vd Linden and I am conducting research at WITS University for a Master of Science in Engineering. The purpose of my research is to better understand the factors for failure and success of software development projects in South Africa. The questionnaire data will be used to define a classification framework that software development teams can use to be successful more often.

This study requires information from people that earn a living creating software. If you are between the ages of 18 and 65 and work in a software development team as a developer, analysts, project manager or software architect, your input will be valuable. In addition to your age and title, you will be asked to answer thirty multiple choice questions based on your experience developing software systems. This should not take more than fifteen minutes of your time.

Every precaution has been made to ensure anonymity and confidentiality. As an online participant in this research, there is always the risk of intrusion by outside agents such as hacking and therefore the possibility of being identified. As the questionnaire is online, your IP address will be saved to make sure that only responses from South Africa are counted. The risk to you is that an IP address can be used to get the location of your internet service provider, but cannot be used to get any personal information. No other digital information will be collected or saved. You are furthermore not required to provide any identifiable information. Your response is collected using a secure and encrypted web protocol, to ensure that it is not tampered with and cannot be viewed by any third party. The data is pooled and kept securely online.

The individual data will be summarised to draw an industry related opinion and reported as part of my dissertation, after which it will be deleted. My research

dissertation will be made available on the internet by the university and the summarised data may be used to submit an article to an academic or professional journal. Besides these, your contribution will not be used for anything else, nor will it be provided for additional or related research.

Taking part in this research is voluntary and you are in no way obligated to complete the questionnaire. Your refusal to participate will in no way result in any penalty or loss of benefit whatsoever to which you are otherwise entitled. Should there be a need, or you feel uncomfortable answering any question, you may skip these questions or withdraw from the study completely, again, without any prejudice, penalty or loss of benefit to you.

Limited to no risks are expected when participating in this study, but should you experience any distress following your participation, I encourage you to contact me directly using the contact information below to arrange appropriate counselling. Similarly, besides your interest in participating in this study, there are no direct benefits to you, but should you wish to see the outcome of the summarised questionnaire results, feel free to contact me or my research supervisor. Relevant contact information is available at the end of this questionnaire.

I thank you for taking the time to read this information sheet and welcome you to contact me should you have any questions related to this study.

Tony

Please note: By completing the questionnaire, your agreement to participate in the research is assumed.

Contact information

To ensure your safety and peace of mind, the study has been reviewed by the Human Research Ethics Committee (Non-medical) (HREC) at WITS University. Their contact information has been added below. Should you wish to lay a

complaint or raise concerns about any aspect of this study and do not want to contact me directly, you can contact my research supervisor.

Researcher	***Research supervisor***	***HREC WITS***
Anthony vd Linden	*Prof. Barry Dwolatzky*	*Lucille Mooragan*
082 619 0037	*078 288 1785*	*011 717 1408*
Tony.vd.linden@live.co.za	*Barry.Dwolatzky@wits.ac.za*	*Lucille.Mooragan@wits.ac.za*

Annexure D: Question design

Participants are asked to complete a questionnaire focusing on reasons for software development successes and failures collected during the factory aggregation exercise. The questionnaire is divided into four sections. The first functioning as a participant information sheet, explaining the purpose of the survey and the second collecting demographic data. The third focuses on the experience of the participant on a software development project deemed to be successful and the fourth focusing on the experience of the participant on a software development project deemed to be a failure.

The design of sections three and four of the questionnaire are based on the failure areas described in Annexure B (*Literature review summary on reasons for software project failure*).

This approach allows for a reasonable comparison between data collected during the literature review and data gathered during the questionnaire stage. Questionnaire questions are created as follows:

1. Communication & Internal Organizational

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Organizational factors	Inadequate focus	Political impacts

- Did the participant know who the responsible person(s) were, and what they were responsible for during the development project? Was there a good balance of capabilities in the team and did the participant know whom to ask for help when required?

- Did the team work together well and were everyone's views and opinions respected? Were decisions made as a team and were all parties that were involved, able to support the decisions that were made?

- Did the development team work together on the project from start to finish? Were members allocated full-time to the team and did the company consistently support the team?

2. Complexity

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Complexity or size factors	Problems with system content	Internal variability

- Was the aim of the system under development easy to understand from start to finish?
- Was the system under development modular and straightforward to implement? Was it easy to see how each portion of the system worked together?
- Was the system under development simple and fun to build? Did the team struggle to understand what was required?

3. Estimation

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Process factors	Problems in execution	Internal variability

- Did the teams' daily non-development activities make it easy to complete the work on time? Was the participant happy with the way the development team worked and was the value to be found in their way of working, clear?

4. Management

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Top management factors	Inadequate focus	Political impacts

- Was the management of the participant's company interested in what they were busy with and were they available and willing to assist the development team with any issues they experienced? Did the management of the participant's company often meet with the development team to see what they could do to assist the team in completing the project?

5. Stakeholders & User

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Project management factors	Problems with system content	Internal variability

- Was the client considered a part of the executing team? Was the client always available and were they able to give guidance, or clear up requirements when required?

6. Planning, Requirements & Goals

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Project management factors	Problems with system content	Internal variability

- Did the development team have a good understanding of what was required and did they have the knowledge required to achieve their goals. Did the development team's output resemble the requirements?

7. Schedule, Scope & Quality

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Process factors	Problems in execution	Internal variability

- Did the development team struggle to complete their work, regardless of the non-development activities they had to complete, including everything promised, such as design, development, testing and integration? Were all non-development activities completed regardless of time pressures?

- Did the development team sacrifice on any deliverables, such as design, development, testing and integration in order to perform the non-development activities associated with the way the team worked? Were these performed within the allowed time?

- Did the development team find it easy to work in the manner that they did? Did it feel natural and was it easy to see the value the team derived from working the way they did?

8. Team, Tooling & Methodology

Al-Ahmad Category	McKinsey-Oxford Category	Research category
Technology factors	Skills problems	Internal variability

- Did the participant's team have the required technical and non-technical capabilities to deliver on the requirements? If the development team found they were stuck, did they have the time to learn new skills through training initiatives provided by the company?

- Did the development team implement different checks and balances to ensure the quality of the deliverable? Was this approach welcomed by the company and were the team able to perform these activities without management interference. Was the participant proud of what was delivered by the development team?

Annexure E: Questionnaire

Introduction

This questionnaire forms part of research being done on why some software development project fail while others succeed. You will be asked to answer thirty questions. The questionnaire has two sections, the first focusing on software development projects you would consider successful (section 1) and the second software development projects you would consider challenged or failed (section 2). Success means that the project was completed in time, on budged and had all the functionality required from the start. Failure means that the project was not delivered, were abandoned before it was complete or were completed, but never used. Challenged means that a project was delivered, but not within the required time, was over budget or did not have all the planned functionality when it was delivered.

Please select an answer that best describes your view of the situation in both sections.

Demographic information

Please select your age group

18 - 22	23 - 30	31 - 40	41 - 50	51 - 65

Please select the option that describes your role or position most accurately

Developer	Business Analyst	Project Manager	Architect	Other

Other (please specify)

About how many years have you been in the software industry?

Less than 1 year	At least 1 year but less than 2 years	At least 2 years but less than 7 years	At least 7 years but less than 12 years	12 years or more

Section 1: Successful software projects

In this section, you are asked to focus on what you would consider a software development projects that went WELL. In answering the following questions, focus on the successful project and evaluate each question according to its success criteria, then rate it using the agreement scale of 1 to 5 where 1 is the lowest and 5 the highest.

1. The management of our company was always interested in what we were busy with and was available and willing to assist us with issues we experienced. (and/or) They often met with the team to see what they could do to help us complete the project.

1	2	3	4	5

2. The team worked well together and everyone's views and opinions were respected and discussed. (and/or) Decisions were made as a team and everyone could support it.

1	2	3	4	5

3. The team worked together on the project from start to finish. (and/or) All members were allocated full time to the team and our company supported us

1	2	3	4	5

4. We knew exactly who was responsible for what. (and/or) There was a good capability balance in the team and you always knew who to ask for help within the team

1	2	3	4	5

5. The client was considered part of the team, was always available and could give guidance or clear up requirements when we needed it

1	2	3	4	5

6. We had a good understanding of what was required and the knowledge to achieve our goals. Our output closely resembled our requirements

1	2	3	4	5

7. It was easy to understand what the aim of the system * was from end to end

1	2	3	4	5

8. The system was very modular and simple to implement. (and/or) It was easy to see how each portion of the system fitted together

1	2	3	4	5

9. It was a simple and fun system to build. (and/or) We never struggled with understanding what was required

1	2	3	4	5

10. My team had the required technical and non-technical capabilities to delivery on our requirements (and/or) If we were stuck, we had time to learn new skills through training that our company provided

1	2	3	4	5

11. The team implemented several different checks and balances to make sure that what we delivered was always correct and working. (and/or) This was welcomed by our company and we could do this without any interference. (and/or) Once we were done, we were proud with what we delivered

1	2	3	4	5

12. The teams daily non-development activities made it easy for us to complete the work on time. (and/or) We were happy with the way we worked as the value was clear

1	2	3	4	5

13. We never struggled to complete our work, regardless of the non-development activities we had to do. This included everything we promised, such as design, development, testing and integration. (and/or) We always completed all non-development activities as well, and never had to skip any of them due to time pressures

1	2	3	4	5

14. We found it easy to work the way we did. (and/or) It felt natural and it was easy to see the value we were getting from working the way we did

1	2	3	4	5

15. We did not have to sacrifice any of the things we promised, such as design, development, testing and integration due to performing the non-development activities associated to how we worked. (and/or) They were planned for and did not take any longer than what they were supposed to

1	2	3	4	5

16. Which of the following best describes the software development methodology followed during the development of the project you used to answer the questions above?

Formal style methodologies	Agile style methodologies	Own/bespoke methodologies	No Methodologies

Section 2: Failed/Challenged software projects

In this section, you are asked to focus on what you would consider a software development projects that did NOT go well, i.e. the project was challenged and/or failed. In answering the following questions, focus on the failed/challenged project and evaluate each question according to its failure/challenge criteria, then rate it using the agreement scale of 1 to 5 where 1 is the lowest and 5 the highest.

1. My team lacked the required technical and non-technical capabilities to delivery on our requirements. (and/or) If we were stuck, we were unable to learn new skills through training as there was no time

1	2	3	4	5

2. It was difficult to understand what the aim of the system * was from end to end

1	2	3	4	5

3. The team struggled to work together (and/or) everyone's views and opinions were not respected and discussed. (and/or) Decisions were not made as a team (and/or) decisions that were made enjoyed little buy in from team members

1	2	3	4	5

4. We had to sacrifice some of the things we promised, such as design, development, testing and integration due to performing the non-development activities associated to how we worked. (and/or) They were not planned for and took longer than what they were supposed to

1	2	3	4	5

5. The teams daily non-development activities made it difficult for us to complete the work on time. (and/or) We were unhappy with the way we worked as the value was not clear

1	2	3	4	5

6. The client was not considered part of the team, was rarely available (and/or) was not able to give guidance or clear up requirements when we needed it

1	2	3	4	5

7. It was a difficult system to build. (and/or) We struggled with understanding what was required

1	2	3	4	5

8. The system was not very modular and simple to implement. (and/or) It was difficult to see how each portion of the system fitted together

1	2	3	4	5

9. The team did not work together on the project from start to finish. (and/or) All members were not allocated full time to the team (and/or) our company did not support us

1	2	3	4	5

10. The management of our company was not interested in what we were busy with (and/or) was not available and willing to assist us with issues we experienced. (and/or) They did not meet with the team to see what they could do to help us complete the project

1	2	3	4	5

11. The team did not implement several different checks and balances to make sure that what we delivered was always correct and working. (and/or) Once we were done, we were not proud with what we delivered

1	2	3	4	5

12. We did not have a good understanding of what was required (and/or) did not have the knowledge to achieve our goals. (and/or) Our output did not resemble our requirements

1	2	3	4	5

13. We struggled to complete our work due to the non-development activities we had to do. (and/or) We never or seldom completed non-development activities, and had to skip some of them due to time pressures

1	2	3	4	5

14. We found it difficult to work the way we did. (and/or) It did not feel natural (and/or) it was difficult to see the value we were getting from working the way we did

1	2	3	4	5

15. We did not know exactly who was responsible for what. (and/or) There was a poor capability balance in the team (and/or) you never knew who to ask for help within the team

1	2	3	4	5

16. Which of the following best describes the software development methodology followed during the development of the project you used to answer the questions above?

Formal style methodologies	Agile style methodologies	Own/bespoke methodologies	No Methodologies

Thank you for your time. You may leave any comments you have, but ensure you don't reveal any identifiable information about yourself.

Annexure F: Ethical Considerations

It could be argued that effective research is founded on morality. The Oxford dictionary defines ethical as "relating to moral principles or the branch of knowledge dealing with these" and "morally good or correct" (Oxford University Press, 2017b). Beauchamp and Childress suggested that moral principles could be divided into one of four categories. These included autonomy, nonmaleficence, beneficence and justice (Beauchamp and Childress, 2001). These classes are based on the expectations an individual has when deciding whether to take part in research, or the responsibility the researcher has in conducting the research. The former includes autonomy, referring to the right of an individual to decide to participate in research without the fear of coercion while knowing what the study is about and justice; referring to all participants being treated equally. The researcher's responsibility includes non-maleficence, referring to the intention of not physically and psychologically harming the participant and beneficence, highlighting the benefits the research should provide to society. In addition, Beauchamp and Childress provided four moral rules that are related to autonomy. These included veracity, fidelity, confidentiality and privacy (Ibid.).

When considering this research, ethicality is considered upmost. It is noted that the use of the internet provides specific challenges related to confidentiality and anonymity. Hox described that "in order to obtain respondents' trust and cooperation, they should be assured that their data will be kept secure (i.e., data are secured during transport and when they are stored on the server), confidential (a response may be linked to a person's name, but only by the survey researcher for the purpose of the survey management and not by others), or sometimes even anonymous (a response cannot be linked to a person's name)" (de Leeuw, Hox and Dillman, 2008). It is argued that the breakdown in trust has resulted in a lack of research cooperation over the years, especially when considering the assumed unethical way market research had been abusing the population. Thus, the research attempted to incorporate best-practise ethical approaches in conducting

data collection, and as such, an application was made to the Human Research Ethics Committee (Non-Medical) at the University of the Witwatersrand. The application received approval, and a clearance certificate was issued (*Attached as Annexure G Ethics Clearance Certificate*).

Moe suggested that anonymous study was a practice that made it impossible to trace the respondents answer back to the respondent and confidentiality extends to removing any identifiable information about a respondent should there be a need to collect personal information (Moe, 2017). The use of an online tool provides some capabilities in relation to both anonymity and confidentiality, but Hox suggested that full anonymity "cannot always be assured for administrative reasons" (de Leeuw, Hox and Dillman, 2008). The author further stated that "ethical guidelines and principles dealing with web surveys pose large emphasis on problems of data security, confidentiality, and anonymity" (Ibid.).

The online surveying tool allows anonymity in terms of the way it is presented to respondents. Respondents are informed that no identifiable information is collected, and are asked, where applicable, not to provide any comments that could expose information about themselves. There was no need for a respondent to provide any credentials in order to access the questionnaire, thus allowing respondents to simply visit the website, interact with the questionnaire and complete it. However, there was a need to collect the IP address of the respondent to ensure that responses were gathered from the South African population only and as such, anonymity could be compromised. Respondents were informed about the potential risks should the IP information be jeopardised, but were assured that this information would only expose the location of their internet service provider and nothing else. In explaining the associated risks, respondents were informed of the availability of counselling, had they experienced any distress following their participation.

In considering confidentiality, the exporting of research data from the surveying tool to a locally administered database included a process that removed the IP

address information from the datasets. This approach was in order to limit the risk of leaking respondents IP addresses further.

Data security was considered in two forms, the first related to the transport of data from the respondent to the surveying tool and the second the encryption of data in the locally managed database. The online surveying tool solved the first challenge in terms of the essential security measures the tool provided. Data transportation from the respondent (client) to the surveying tool (server) was done over a secure version of HyperText Transfer Protocol, meaning that all communication between the client and the server were encrypted. An attack on the transportation layer may have resulted in the attacker (man-in-the-middle) gaining access to the data begin transported, but its encrypted nature precluded the attacker from viewing the data.

Respondents were advised of their right to withdraw from the process at any given point. The aim of the research was explained, and respondents were informed that taking part in the research was voluntary and they were in no way obligated to complete the questionnaire. In addition, it was stated that any refusal to participate would in no way result in any penalty or loss of benefit, to which they were otherwise entitled.

This information was consolidated and provided to respondents on receiving the survey request. The information sheet can be viewed in Annexure C (*Participants information sheet*).

The following ethical considerations were observed as part of this study:

- An application was made to conduct the survey research, upon which approval was granted by the Human Research Ethics Committee (Non-Medical) at the University of the Witwatersrand (*Attached as Annexure G Ethics Clearance Certificate*).
- No other parties, such as businesses or organisations were involved, so no permissions were required.

- A written statement provided to each participant informed them of their rights regarding participation, the aim of the research, the risks associated with partaking in the research and the availability of counselling, should there be a need. (*Annexure C Participants information sheet*)
- Respondents provided their consent of participation by taking part in the research. Participants were informed of their implied agreement by participation.
- Every precaution was taken to ensure that respondents remained anonymous.
- No participant was coerced into completing the questionnaire, and no rewards were offered.
- In no way were any participants or respondents subjected to any harm or risk associated to completing the questionnaire.

Finally, all data collected and findings made were approached from a position of ethicality, honesty and integrity. Any report provided by this study was created with the aim of maintaining the researcher's integrity whilst adding value to the software industry and the bodies of knowledge that they underpin.

Annexure G: Ethics Clearance Certificate

Research Office

HUMAN RESEARCH ETHICS COMMITTEE (NON-MEDICAL)
R14/49 van der Linden

CLEARANCE CERTIFICATE	**PROTOCOL NUMBER: H17/03/21**
PROJECT TITLE	An investigation into the catalysts taht fuel success or failure of software development initiatives in South Africa
INVESTIGATOR(S)	Mr A van der Linden
SCHOOL/DEPARTMENT	Electrical and Information Engineering/
DATE CONSIDERED	17 March 2017
DECISION OF THE COMMITTEE	Approved
EXPIRY DATE	06 April 2020

DATE 07 April 2017 **CHAIRPERSON** J. Knight
(Professor J Knight)

cc: Supervisor : Professor B Dwolatsky

DECLARATION OF INVESTIGATOR(S)

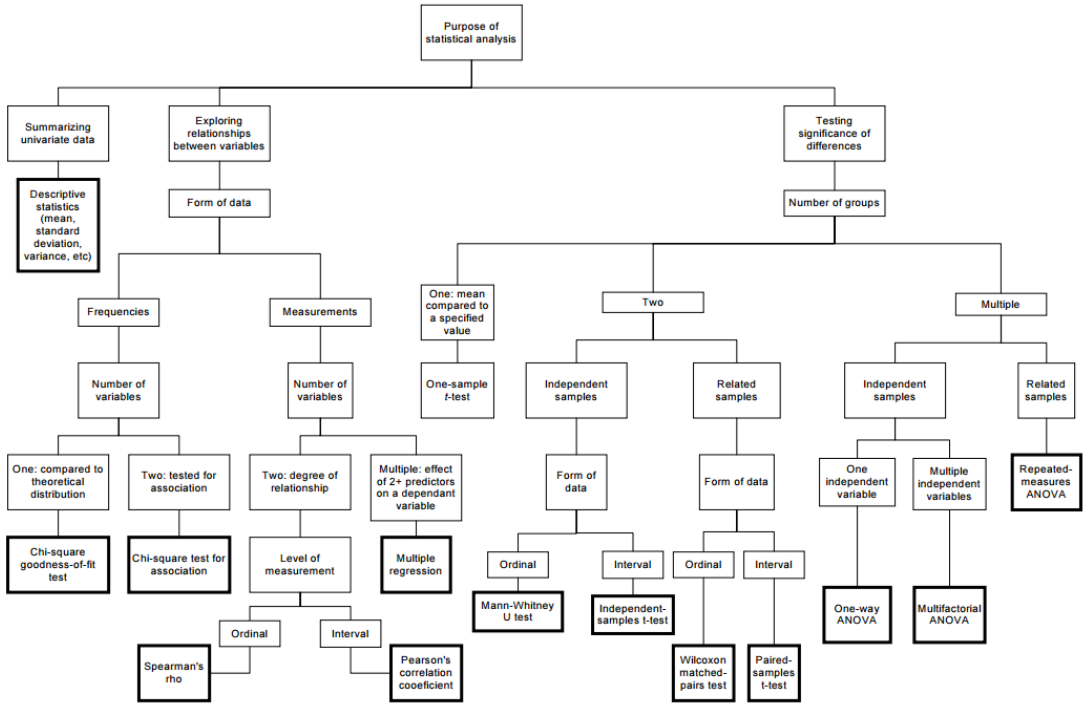
To be completed in duplicate and **ONE COPY** returned to the Secretary at Room 10004, 10th Floor, Senate House, University. Unreported changes to the application may invalidate the clearance given by the HREC (Non-Medical)

I/We fully understand the conditions under which I am/we are authorized to carry out the abovementioned research and I/we guarantee to ensure compliance with these conditions. Should any departure to be contemplated from the research procedure as approved I/we undertake to resubmit the protocol to the Committee. **I agree to completion of a yearly progress report.**

_____ _____/_____/_____
Signature **Date**

PLEASE QUOTE THE PROTOCOL NUMBER ON ALL ENQUIRIES

Annexure H: Choosing an appropriate statistical procedure



Source 1 : (Colman and Pulford, 2008)

Annexure I: Research Data Analysis

1 Sample

This section provides additional insight into the research sample with specific focus on the cross-tabulation of roles, generations and tenure.

1.1 Strata cross-tabulation

The role versus generation cross-tabulation was provided to demonstrate the sample makeup of generations as they apply to role. In illustrating the sample makeup in terms of a cross-tabulation, a different perspective was created that augmented the understanding of the primary contributors towards the study in terms of generation and role. Table 31 (*Role versus Generation*) illustrates that in terms of the role generation perspective, the sample majority consisted of developers and business analysts with a generation classification of early to mid-generation Y, followed by project managers and architects with a generation classification of mid to late generation X. Similarly, the role versus tenure cross-tabulation was provided to demonstrate the sample makeup of tenure as they apply to role. For this dimension, it was found that developers and business analysts in the tenure classification of junior to intermediate level provided the majority, followed by project managers and architects in the tenure classification of senior level. Table 32 (*Role versus Tenure*) provides an illustration of the sample makeup when role and tenure is considered.

In addition to the role versus generation and role versus tenure views, two additional views were added to describe the overall research sample. These views incorporated the role and generation versus tenure (*Table 33 Role and Generation versus Tenure cross-tabulation*) and role and tenure versus generation (*Table 34 Role and Tenure versus Generation cross-tabulation*) perspectives.

Table 31 Role versus generation

	Role	Developer	Business Analyst	Project Manager	Architect
Generations	Generation Z (18 - 22)	5	6	0	0
	Early-Mid Generation Y (23 - 30)	141	33	1	3
	Late Generation Y - Early Generation X (31 - 40)	75	27	14	3
	Mid - Late Generation X (41 - 50)	14	11	21	12
	Late Generation X - Early Boomers (51 - 65)	8	4	4	2

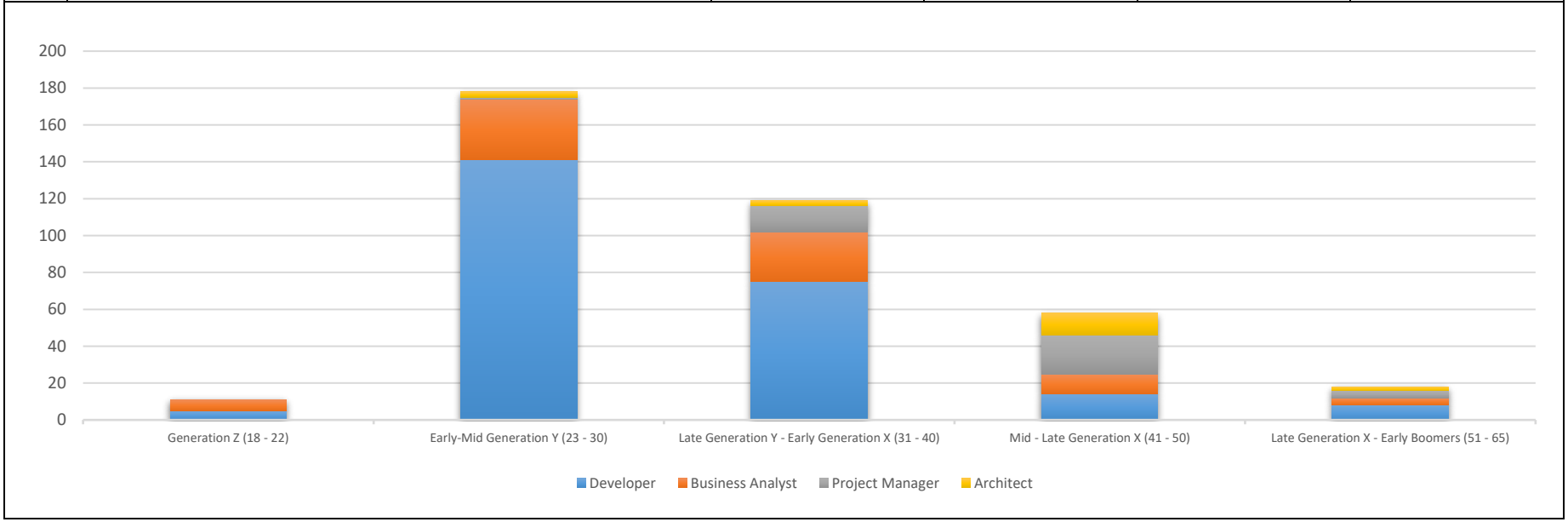


Table 32 Role versus Tenure

	Role	Developer	Business Analyst	Project Manager	Architect
Tenure	Learner level (Less than 1 year)	15	6	0	0
	Junior level (At least 1 year but less than 2 years)	25	6	0	0
	Junior to intermediate level (At least 2 years but less than 7 years)	115	31	3	3
	Intermediate to Senior level (At least 7 years but less than 12 years)	48	23	11	1
	Senior level (12 years or more)	40	15	26	16

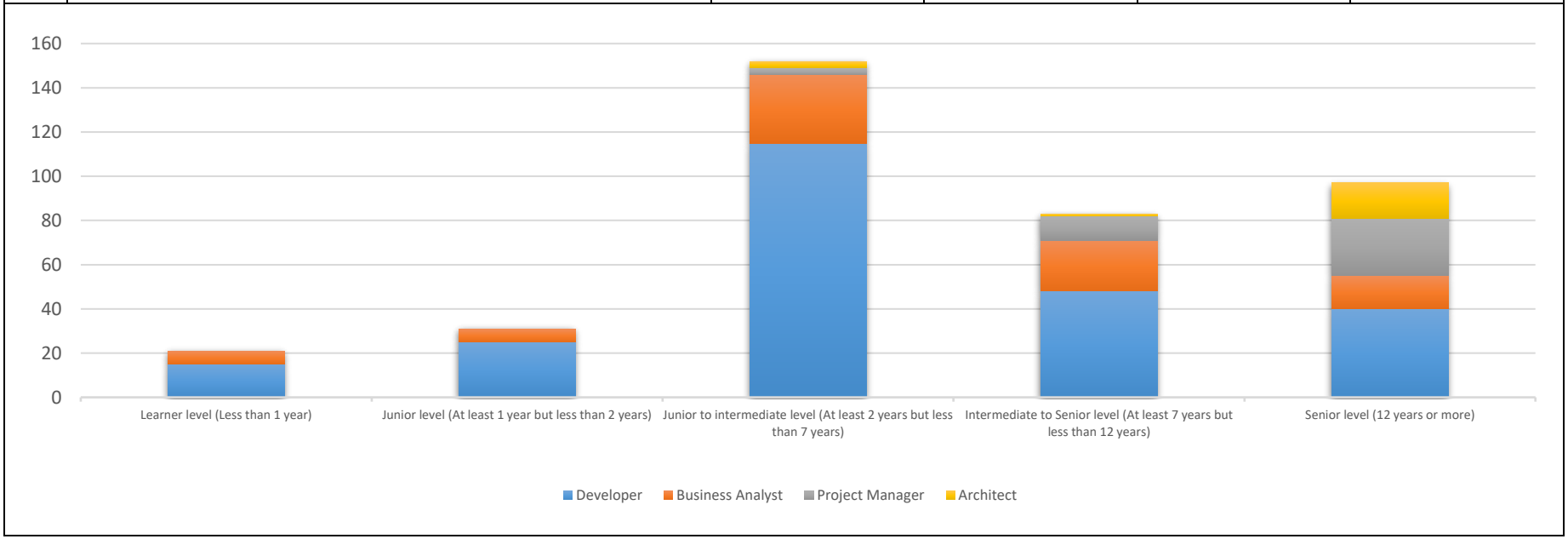


Table 33 Role and Generation versus Tenure cross-tabulation

	Developer					Business Analyst					Project Manager					Architect					Sum of category	Percentage of total
	Gen Z	Early-Mid Gen Y	Late Gen Y - Early Gen X	Mid - Late Gen X	Late Gen X - Early Boomers	Gen Z	Early-Mid Gen Y	Late Gen Y - Early Gen X	Mid - Late Gen X	Late Gen X - Early Boomers	Gen Z	Early-Mid Gen Y	Late Gen Y - Early Gen X	Mid - Late Gen X	Late Gen X - Early Boomers	Gen Z	Early-Mid Gen Y	Late Gen Y - Early Gen X	Mid - Late Gen X	Late Gen X - Early Boomers		
Learner level	4	11	0	0	0	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	21	5%
Junior level	1	23	1	0	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	31	8%
Junior to intermediate level	0	97	17	1	0	0	25	4	2	0	0	0	2	1	0	0	2	1	0	0	152	40%
Intermediate to Senior level	0	10	38	0	0	0	2	20	1	0	0	1	6	4	0	0	0	0	1	0	83	22%
Senior level	0	0	19	13	8	0	0	3	8	4	0	0	6	16	4	0	1	2	11	2	97	25%
Sum of category	5	141	75	14	8	6	33	27	11	4	0	1	14	21	4	0	3	3	12	2	384	
Percentage of total	1%	37%	20%	4%	2%	2%	9%	7%	3%	1%	0%	0%	4%	5%	1%	0%	1%	1%	3%	1%		

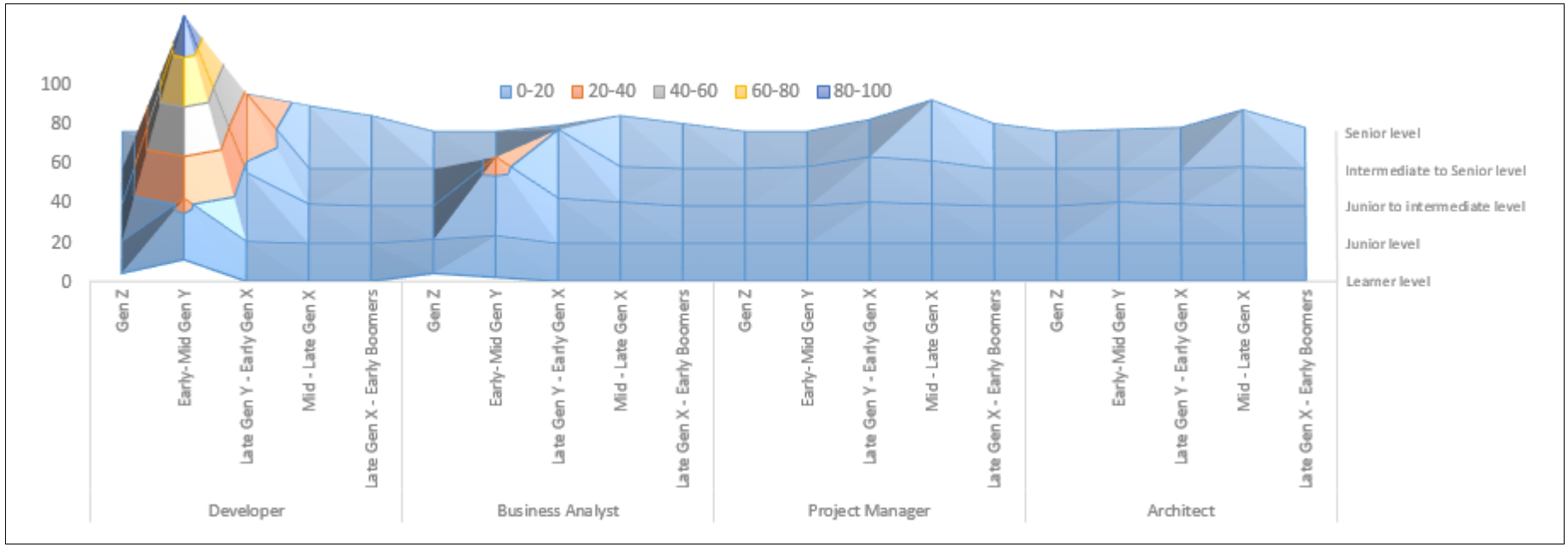


Figure 23: Role and Generation versus Tenure sample distribution

In viewing the data in graphical form (*Figure 23 Role and Generation versus Tenure sample distribution*), it was found that the research sample had a strong association to developers in the early to mid-generation Y and junior to intermediate tenure categories. For business analysts, a similar association was found when generation was considered, but in terms of tenure, both junior to intermediate and intermediate

to senior levels contributed the most. Both project managers and architects were found to be in a mid to late generation X and at a senior tenure level.

Table 34 Role and Tenure versus Generation cross-tabulation

	Developer					Business Analyst					Project Manager					Architect					Sum of category	Percentage of total
	Learner level	Junior level	Junior to intermediate level	Intermediate to Senior level	Senior level	Learner level	Junior level	Junior to intermediate level	Intermediate to Senior level	Senior level	Learner level	Junior level	Junior to intermediate level	Intermediate to Senior level	Senior level	Learner level	Junior level	Junior to intermediate level	Intermediate to Senior level	Senior level		
Gen Z	4	1	0	0	0	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	11	3%
Early-Mid Gen Y	11	23	97	10	0	2	4	25	2	0	0	0	0	1	0	0	0	2	0	1	178	46%
Late Gen Y - Early Gen X	0	1	17	38	19	0	0	4	20	3	0	0	2	6	6	0	0	1	0	2	119	31%
Mid - Late Gen X	0	0	1	0	13	0	0	2	1	8	0	0	1	4	16	0	0	0	1	11	58	15%
Late Gen X - Early Boomers	0	0	0	0	8	0	0	0	0	4	0	0	0	0	4	0	0	0	0	2	18	5%
Sum of category	15	25	115	48	40	6	6	31	23	15	0	0	3	11	26	0	0	3	1	16	384	
Percentage of total	4%	7%	30%	13%	10%	2%	2%	8%	6%	4%	0%	0%	1%	3%	7%	0%	0%	1%	0%	4%		

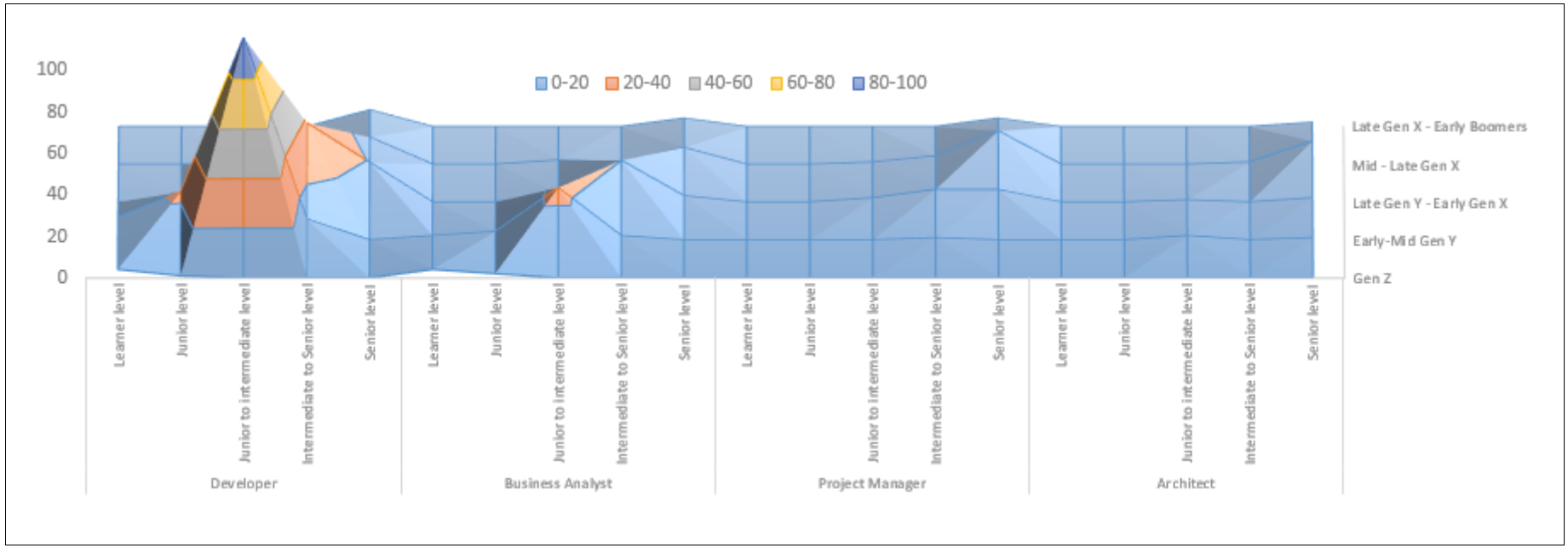


Figure 24: Role and Tenure versus Generation sample distribution

In viewing the data in graphical form (*Figure 24 Role and Tenure versus Generation sample distribution*), it was found that the research sample had a strong association to developers in the junior to intermediate and intermediate to senior level. For business analysts, a similar association was found when tenure was considered, but in terms of generation, both early to mid-generation Y and late generation Y to early generation X contributed the most. Both project managers and architects were found to be in a senior level and in mid to late generation X

1.2 Analysis boundary

The research instrument collected an enormous amount of data. Its design incorporated the three independent variables of role, tenure and generation as well as the dependent variable of software outcome. However, the instrument made available two parts capable of collecting software outcomes in terms of failure as well as success. This resulted in two datasets capable of testing the research hypotheses in multiple ways, for instance, perspectives of different roles (or tenure or generation) for outcomes that resulted in a failed software project, perspectives of different roles (or tenure or generation) for outcomes that resulted in successful software project, a combined perspective of roles (or tenure or generation) and software outcomes regardless of them being a failure or success and so on.

Two distinct categorisation were also incorporated into the research, capable of defining a consolidated view of factors responsible for the outcome. These included “internal variability” and “political impacts”. As such, a different dimension was created capable of adding an additional perspective of the data in terms of its independent and dependent variables. These included a perspective of different roles (or tenure or generation) for outcomes that resulted in a failed software development project, in terms of political impacts, a perspective of different roles (or tenure or generation) for outcomes that resulted in a failed software development project, in terms of “internal variability”, and so on.

In addition, both “internal variability” and “political impacts” were created as an aggregation of two different categorisation frameworks referenced to in this report as the McKinsey-Oxford categorisation and the Al-Ahmad categorisation. As such, additional perspectives were possible like those highlighted above. Tables 35 (*MScEng*), 36 (*McKinsey-Oxford*) and 37 (*Al-Ahmad*) below provides a view of all perspectives available in the research data.

Table 35 MScEng

MScEng Categorisation							
View number	Internal variability	Political impacts	Role	Tenure	Generation	Success	Failure
1	X		x			x	
2	x			x		x	
3	x				x	x	
4	x		x				x
5	x			x			x
6	x				x		x
7	x		x	x	x	x	
8	x		x	x	x		x
9	x		x	x	x	x	x
10		x	x			x	
11		x		x		x	
12		x			x	x	
13		x	x				x
14		x		x			x
15		x			x		x
16		x	x	x	x	x	
17		x	x	x	x		x
18		x		x	x	x	x
19	x	x	x			x	
20	x	x		x		x	
21	x	x			x	x	
22	x	x	x				x
23	x	x		x			x
24	x	x			x		x
25	x	x	x			x	x
26	x	x		x		x	x
27	x	x			x	x	x

Table 36 McKinsey-Oxford

McKinsey – Oxford categorization									
View number	Inadequate focus	Problems with system content	Skills problems	Problems in execution	Role	Tenure	Generation	Success	Failure
28	x				x			x	
29	x					x		x	
30	x						x	x	
31	x				x				x
32	x					x			x
33	x						x		x
34	x				x	x	x	x	
35	x				x	x	x		x
36	x				x	x	x	x	x
37		x			x			x	
38		x				x		x	
39		x					x	x	
40		x			x				x
41		x				x			x
42		x					x		x
43		x			x	x	x	x	
44		x			x	x	x		x
45		x			x	x	x	x	x
46			x		x			x	
47			x			x		x	
48			x				x	x	
49			x		x				x
50			x			x			x
51			x				x		x
52			x		x	x	x	x	
53			x		x	x	x		x
54			x		x	x	x	x	x
55				x	x			x	
56				x		x		x	
57				x			x	x	
58				x	x				x

Table 37 Al-Ahmad

	Al-Ahmad et al. categorization										
View number	Top management factors	Organizational factors	Project management factors	Complexity or size factors	Technology factors	Process factors	Role	Tenure	Generation	Success	Failure
73	x						x			x	
74	x							x		x	
75	x								x	x	
76	x						x				x
77	x							x			x
78	x								x		x
79	x						x	x	x	x	
80	x						x	x	x		x
81	x						x	x	x	x	x
82		x					x			x	
83		x						x		x	
84		x							x	x	
85		x					x				x
86		x						x			x
87		x							x		x
88		x					x	x	x	x	
89		x					x	x	x		x
90		x					x	x	x	x	x
91			x				x			x	
92			x					x		x	
93			x						x	x	
94			x				x				x
95			x					x			x
96			x						x		x
97			x				x	x	x	x	
98			x				x	x	x		x
99			x				x	x	x	x	X

View number	Inadequate focus	Problems with system content	Skills problems	Problems in execution	Role	Tenure	Generation	Success	Failure
59				x		x			x
60				x			x		x
61				x	x	x	x	x	
62				x	x	x	x		x
63				x	x	x	x	x	x
64	x	x	x	x	x			x	
65	x	x	x	x		x		x	
66	x	x	x	x			x	x	
67	x	x	x	x	x				x
68	x	x	x	x		x			x
69	x	x	x	x			x		x
70	x	x	x	x	x			x	x
71	x	x	x	x		x		x	x
72	x	x	x	x			x	x	x

View number	Top management factors	Organizational factors	Project management factors	Complexity or size factors	Technology factors	Process factors	Role	Tenure	Generation	Success	Failure
100				x			x			X	
101				x				x		x	
102				x					x	x	
103				x			x				x
104				x				x			x
105				x					x		x
106				x			x	x	x	x	
107				x			x	x	x		x
108				x			x	x	x	x	x
109					x		x			x	
110					x			x		x	
111					x				x	x	
112					x		x				x
113					x			x			x
114					x				x		x
115					x		x	x	x	x	
116					x		x	x	x		x
117					x		x	x	x	x	x
118						x	x			x	
119						x		x		x	
120						x			x	x	
121						x	x				x
122						x		x			x
123						x			x		x
124						x	x	x	x	x	
125						x	x	x	x		x
126						x	x	x	x	x	x
127	x	x	x	x	x	x	x			x	
128	x	x	x	x	x	x		x		x	
129	x	x	x	x	x	x			x	x	
130	x	x	x	x	x	x	x				x
131	x	x	x	x	x	x		x			x
132	x	x	x	x	x	x			x		x
133	x	x	x	x	x	x	x			x	x
134	x	x	x	x	x	x		x		x	x
135	x	x	x	x	x	x			x	x	x

When considering the tables above, it should be noted that dimensions 25,70 and 133 are the same and all relate to all categorisations across all three categorisation frameworks, but all using the independent variable, role. Similarly, 26, 71 and 134 relate to tenure and 27, 72 and 135 relate to generation.

The purpose of this research was to undertake a study to understand if perceptions of factors that influence software outcomes were similar between experts practicing in different roles, generations and tenure. Even though all views specified above assisted in providing insight into the phenomenon highlighted by this research, its narrow focus contained the boundary of explanation to view 26, 27 and 28 only. As each other view was considered a partial decomposition of these views, the inclusion on the selected set of views was deemed appropriate in explaining the research result.

As such, the boundary of the data analysis was confined to software outcomes, regardless of it being a success or failure with a combined view of the categorisation ("internal variability" plus "political impacts") responsible for its outcome as the dependent variable, against the independent variables of role, tenure and generation.

The following section explains each independent variable in relation to software outcomes.

2 Variable analysis

The research aimed to understand if experts from dissimilar social classes reasoned differently when software development outcomes were considered. In defining the research theory, it was noted that the incorporation of the three independent variables of role, generation and tenure could provide sufficient insight in terms social class. As such, three sub-questions were introduced to assist in investigating expert reasoning in terms of the independent variables.

2.1 RQ2: roles

It was suggested that experts in different roles reason differently when factors that influence software development outcomes were considered. To describe this phenomenon, the following research sub-question (RQ2) was posed:

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts in different roles?*

To allow testing of this question, the following hypotheses were posed:

- H02: There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts in different roles.
- H2: Experts in different roles in a software development team have differing views of "internal variability" and "political impacts" as the factors that influence bespoke software development outcomes in South Africa.

2.1.1 View number 25

Software failure and success including "internal variability" and "political impacts" against expert role

This view consolidated all research data into a single dataset that represents a combination of failed and successful software development projects as well as "internal variability" and "political impacts" as factor categorisations. This combination represents an expert's perspective of software outcomes, regardless of the factors that influence software development. The independent variable of role was used as a lens to illustrate the relationship of experts in different roles as a primary viewpoint.

When considering the means of each role, it could be argued that a difference in perspectives existed between each role. Table 38 (*Role – Mean, median, mode*

and standard deviation) below shows that the means for each role differ from between 1.64 and 6.89 percent and Figure 25 (*Role mean comparison*) at this level paints a clear picture of this difference.

Table 38 Role - Mean, median, mode and standard deviation

Role	Mean	Median	Mode	Standard Deviation
Architect	6.88807	7.14773	8.93182	1.60758
Developer	6.77609	6.97727	7.47727	1.56008
Project Manager	6.62386	6.71591	7.75	1.30731
Business Analyst	6.42915	6.21591	5.93182	1.23708

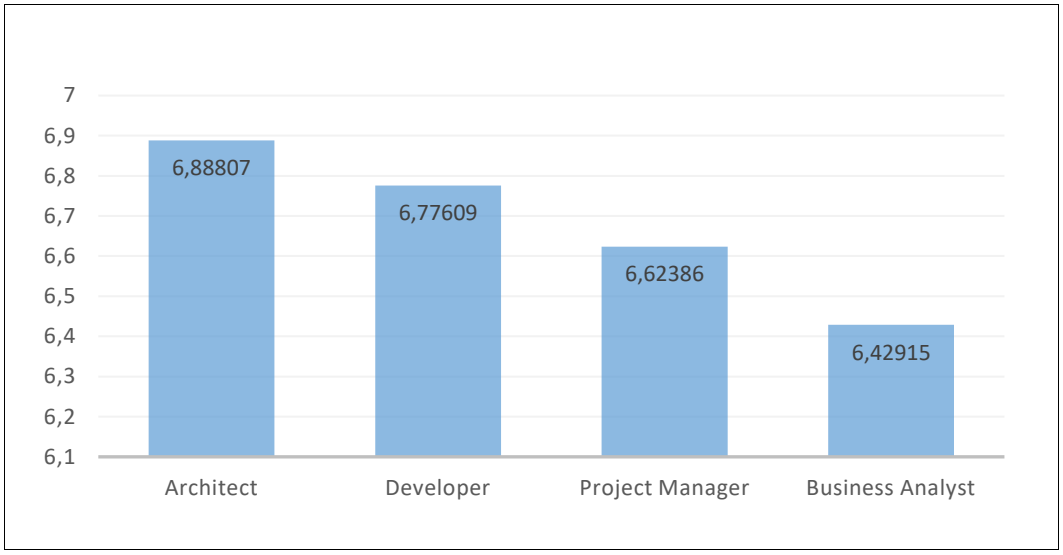


Figure 25: Role mean comparison

2.1.2 Role claims

In terms of what was presented in Figure 25 (*Role mean comparison*), the following claims were made:

- When considering architects and developers, there appears to be a difference of 1.64% between perspective
- When considering architects and project managers, there appears to be a difference of 3.91% between perspective

- When considering architects and business analysts, there appears to be a difference of 6.89% between perspective
- When considering developers and project managers, there appears to a difference of 2.27% between perspective
- When considering developers and business analysts, there appears to a difference of 5.25% between perspective
- When considering project managers and business analysts, there appears to a difference of 2.98% between perspective

2.1.3 Hypothesis testing

Hypothesis testing was done to describe the validity of claims made about the research data. The previous section described differences between each role in the way questions were answered by the research sample. From a pragmatic perspective, these claims shed some light in terms of what the data is saying, but without statistical examination, little relevance could be proven. As such hypothesis testing was included to assist in describing the differences suggested above. Hypothesis testing was done using the standard T-Test as well as the Student's probability test.

To cater for each relationship between the roles defined above, the following null hypotheses were posed to globally represent the null hypothesis, H02.

- $H02_a$: $\mu_{\text{developer}} - \mu_{\text{business analyst}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of developers and business analysts.
- $H02_b$: $\mu_{\text{developer}} - \mu_{\text{project manager}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of developers and project managers.
- $H02_c$: $\mu_{\text{developer}} - \mu_{\text{architect}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of developers and architects.
- $H02_d$: $\mu_{\text{business analyst}} - \mu_{\text{project manager}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of business analysts and project managers.

- $H02_e$: $\mu_{\text{business analyst}} - \mu_{\text{architect}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of business analysts and architects.

- $H02_f$: $\mu_{\text{project manager}} - \mu_{\text{architect}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of project managers and architects.

2.1.3.1 *P-Value & T-Test*

The P-Value test is commonly used for hypothesis testing as it describes the validity of claims made about data. For this research, the values calculated from the P-Value calculation were interpreted with the following in mind.

If a null hypothesis suggested that there was no difference between role and a P-Value was calculated, then the null-hypothesis would be tested against the following:

- **P-Values < 0.01**: *Suggests that strong evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- P-Values < 0.05: *Suggests that sufficient evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- P-Values > 0.06: *Suggests that little evidence can be found that there is a difference between means. The null hypothesis cannot be rejected.*

To calculate the P-Value, the t-Value was required. The standard t-value equation was used to calculate all t-Values for each role combination.

The following variables were defined for use in the equation:

x	Represents an item in the dataset

$n_?$	Number of responses
$\boldsymbol{df}_?$	Number of responses minus 1
$\mu_?$	The mean for the dataset
$\boldsymbol{SS}_?$	The sum of squares where the square is equal to the mean minus the value of x
s	The value of the sum of squares divided by the number of responses minus 1

The following provides a review of how the calculation was used to calculate the t-Values for each role combination. For brevity, this was documented only once.

Table 39 t-Value calculation

Developer	Business analyst
$n_1 = 486$ $\boldsymbol{df_1 = n_1 - 1} = 486 - 1 = 485$ $\mu_1 = 6.7760938$ $\boldsymbol{SS_1} = \sum_{i=0}^{n}(\boldsymbol{\mu_1 - x_1}) = 1182.85$ $\boldsymbol{s_1^2} = \frac{\boldsymbol{ss_1}}{\boldsymbol{(n_1-1)}} = 1182.85 / (486\text{-}1) = 2.44$	$N_2 = 486$ $\boldsymbol{df_2 = n_2 - 1} = 486 - 1 = 485$ $\mu_2 = 6.4291522$ $\boldsymbol{SS_2} = \sum_{i=0}^{n}(\boldsymbol{\mu_2 - x_2}) = 247.92$ $\boldsymbol{s_1^2} = \frac{\boldsymbol{ss_2}}{\boldsymbol{(n_2-1)}} = 247.92 / (486\text{-}1) = 0.51$
$\boldsymbol{s_p^2 = \left(\left(\frac{df_1}{df_1 + df_2}\right) * s_1^2\right) + \left(\left(\frac{df_2}{df_2 + df_1}\right) * s_2^2\right)}$ $\boldsymbol{s_p^2 = \left(\left(\frac{485}{485 + 485}\right) * 2.44\right) + \left(\left(\frac{485}{485 + 485}\right) * 0.51\right)}$ $\boldsymbol{s_p^2 = (0.5 * 2.44) + (0.5 * 0.51)}$ $\boldsymbol{s_p^2 = 1.475091}$	
$\boldsymbol{s_{\mu_1}^2 = \frac{s_p^2}{n_1}}$ $\boldsymbol{s_{\mu_1}^2 = \frac{1.475}{486}}$ $\boldsymbol{s_{\mu_1}^2 = 0.003035}$	

$$s^2_{\bar{\mu}_2} = \frac{s_p^2}{n_2}$$

$$s^2_{\bar{\mu}_2} = \frac{1.475}{486}$$

$$s^2_{\bar{\mu}_2} = 0.003035$$

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s^2_{\bar{\mu}_1} + s^2_{\bar{\mu}_2}}}$$

$$t = \frac{6.7760938 - 6.4291522}{\sqrt{0.003035 + 0.003035}}$$

$$t = \frac{0.3469416}{0.0779102047231298}$$

$$t = 4.4531$$

Table 40 (*Role t-Value*) provides a cross-tabulation of all t-Values calculated for this dimension.

Table 40 Role t-Values

t-Value	A	B	C	D
A	-	-4.4531	-2.0342	1.5162
B	4.4531	-	-4.8208	11.8882
C	2.0342	-4.8208	-	8.2782
D	-1.5162	-11.8882	-8.2782	-

Table legend: *A = Developer, B = Business Analyst, C = Project Manager, D = Architect*

Table 41 (*Role P-Value*) provides a cross-tabulation of all P-Values calculated for each role combination.

Table 41 Role P-Value

P-Value	A	B	C	D
A	-	0.0000	0.0211	0.0649
B	0.0000	-	0.0000	0.0000
C	0.0211	0.0000	-	0.0000
D	0.0649	0.0000	0.0000	-

Table legend: *A = Developer, B = Business Analyst, C = Project Manager, D = Architect*

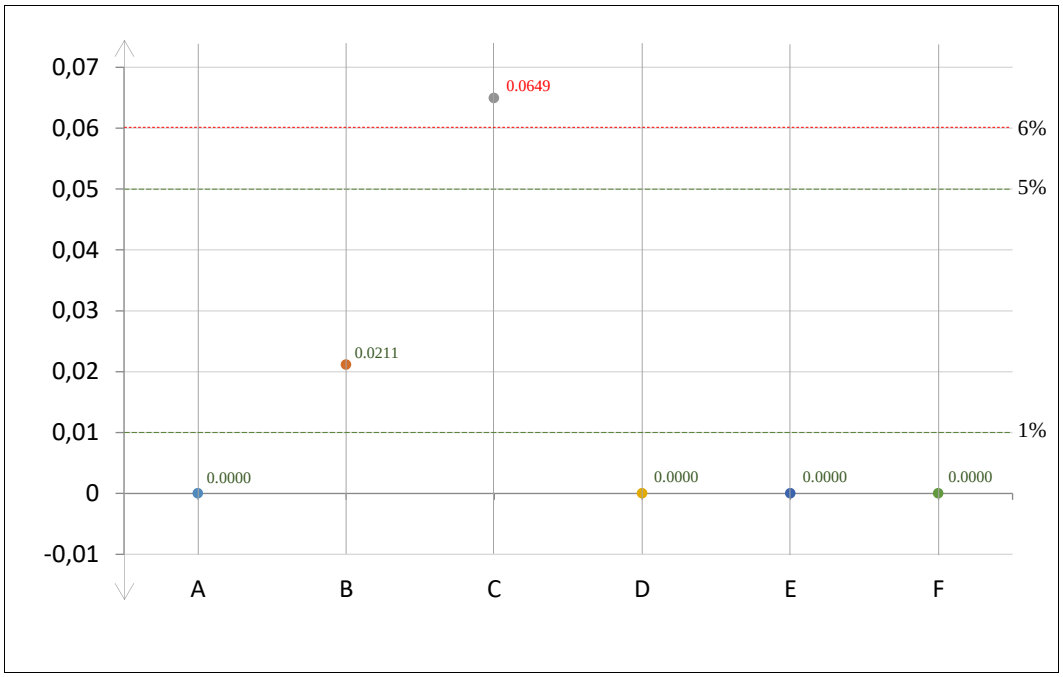


Figure 26: Role P-Value view

Graph legend: *A: Developer-Business Analyst, B: Developer-Project Manager, C: Developer-Architect, D: Business Analyst-Project Manager, E: Business Analyst-Architect, F: Project Manager-Architect*

2.1.3.1.1 ***Outcomes:***

In considering each role combination and the null hypotheses presented above, the following was found using the standard T-Test:

- $H02_a$: $\mu_{developer} - \mu_{business\ analyst} = 0.0000$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H02_b$: $\mu_{developer} - \mu_{project\ manager} = 0.0211$
  - With a P-Value of 2.11%, a difference can be found at 5%. As such, this hypothesis was rejected.
- $H02_c$: $\mu_{developer} - \mu_{architect} = 0.0649$
  - With a P-Value of 6.49%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.
- $H02_d$: $\mu_{business\ analyst} - \mu_{project\ manager} = 0.0000$

- With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.

- $H02_e$: $\mu_{\text{business analyst}} - \mu_{\text{architect}} = 0.0000$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H02_f$: $\mu_{\text{project manager}} - \mu_{\text{architect}} = 0.0000$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.

2.1.3.2 *Student's Probability*

Like the P-Value test, the Student's probability test was used for hypothesis testing. The probability test calculated a cumulative two-tailed density using ACM algorithm 209. The interpretations of the probability outcomes were considered in a similar way than that of the T-Test. These included the following:

If a null hypothesis suggested that there was no difference between role and a probability was calculated, then the null-hypothesis would be tested against the following:

- **probability < 0.01**: *Suggests that strong evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- probability < 0.05: *Suggests that sufficient evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- probability > 0.06: *Suggests that some evidence can be found that there is a difference between means. The null hypothesis cannot be rejected.*

The following degrees of freedom were calculated for use in the Student's probability test:

Table 42 Degree of freedom

df	A	B	C	D
A	-	679.753	595.644	569.128
B	679.753	-	895.188	829.548
C	595.644	895.188	-	951.636
D	569.128	829.548	951.636	-

Table legend: *A = Developer, B = Business Analyst, C = Project Manager, D = Architect*

Table 43 (*Probability test*) provides a cross-tabulation of all probability values calculated for each role combination.

Table 43 Probability test

Probability	A	B	C	D
A	-	0.0000098935	0.0423356949	0.1301195121
B	0.0000098935	-	0.0000016858	0.0000000000
C	0.0423356949	0.0000016858	-	0.0000000000
D	0.1301195121	0.0000000000	0.0000000000	-

Table legend: *A = Developer, B = Business Analyst, C = Project Manager, D = Architect*

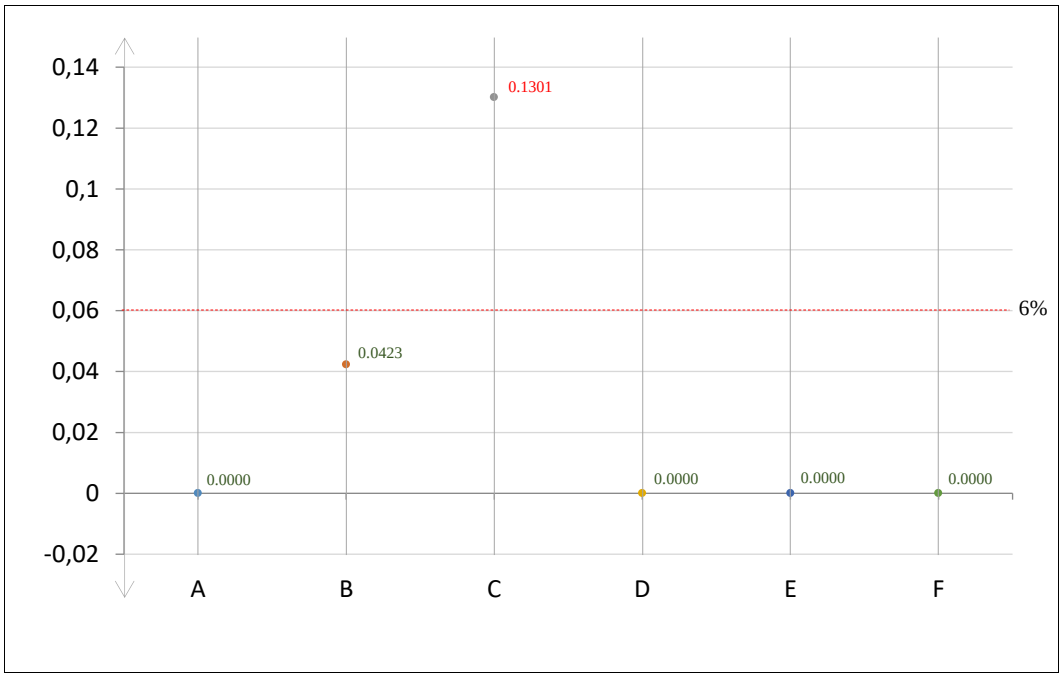


Figure 27: Role Probability test view

Graph legend: *A: Developer-Business Analyst, B: Developer-Project Manager, C: Developer-Architect, D: Business Analyst-Project Manager, E: Business Analyst-Architect, F: Project Manager-Architect*

2.1.3.2.1 *Outcomes:*

In considering each role combination and the null hypotheses presented above, the following was found using the Student's probability test:

- $H02_a$: $\mu_{developer} - \mu_{business\ analyst} = 0.0000$
    - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H02_b$: $\mu_{developer} - \mu_{project\ manager} = 0.0423$
    - With a P-Value of 4.23%, a difference can be found at 5%. As such, this hypothesis was rejected.
- $H02_c$: $\mu_{developer} - \mu_{architect} = 0.1301$
    - With a P-Value of 13.01%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.
- $H02_d$: $\mu_{business\ analyst} - \mu_{project\ manager} = 0.0000$
    - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H02_e$: $\mu_{business\ analyst} - \mu_{architect} = 0.0000$
    - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H02_f$: $\mu_{project\ manager} - \mu_{architect} = 0.0000$
    - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.

2.1.3.3 *Pearson correlation coefficient*

The Pearson correlation method is a parametric correlation test that depends on a distribution of data. The method was used to measure the linear dependence between each role relationship or the strength and direction of a linear relationship between two variables. The Pearson correlation formula produces a R-Value that is between + 1 and –1, where numbers greater than 0 are considered

positive relational and numbers smaller than 0 negative relational. The closer the number is to 0, the greater the variation is between data points. More variation means weaker relationships between data sets.

The following basic formula was used to determine the correlation between different roles:

$$r = \frac{\sum(x - \mu_x)(y - \mu_y)}{\sqrt{\sum(x - \mu x)^2 (y - \mu y)^2}}$$

The value of a R-Value can be understood in terms of one of the following:

- Exactly –1 - A perfect downhill or negative linear relationship
- –0.70 - A strong downhill or negative linear relationship
- –0.50 - A moderate downhill or negative relationship
- –0.30 - A weak downhill or negative linear relationship
- 0 - No linear relationship
- + 0.30 - A weak uphill or positive linear relationship
- + 0.50 - A moderate uphill or positive relationship
- + 0.70 - A strong uphill or positive linear relationship
- Exactly + 1 - A perfect uphill or positive linear relationship

For this study, the values of +0.5 and -0.5 was considered appropriate as the cut-off mark to explain a relationship. As such, any value greater to +0.5 and smaller than -0.5 was considered to have a relationship. Similarly, any value between +0.5 and -0.5 was considered to not have a relationship.

Table 44 Role r-Value

r-Value	A	B	C	D
A	-	0.0568	0.1078	0.0111
B	0.0568	-	0.0036	-0.0509
C	0.1078	0.0036	-	0.0165
D	0.0111	-0.0509	0.0165	-

Table legend: *A = Developer, B = Business Analyst, C = Project Manager, D = Architect*

Table 44 (*Role r-Value*) provides a cross-tabulation of all R-values calculated for this dimension.

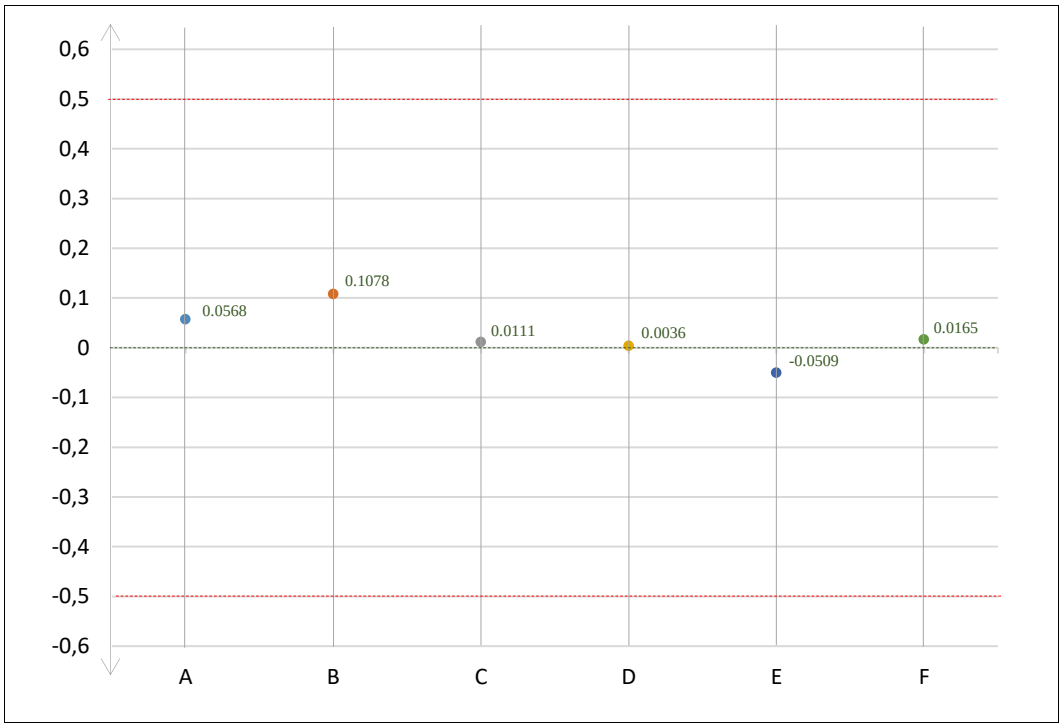


Figure 28: Role R-Value view

Graph legend: *A: Developer-Business Analyst, B: Developer-Project Manager, C: Developer-Architect, D: Business Analyst-Project Manager, E: Business Analyst-Architect, F: Project Manager-Architect*

2.1.3.3.1 ***Outcomes:***

This dimension proposed six different relationships, all related to a combination of each roles. In considering the validity of each relationship, the following outcomes were found:

- Developer versus Business Analyst = 0.0568
    - With a R-Value of 0.06, no linear relationship was found
- Developer versus Project Manager = 0.1078
    - With a R-Value of 0.1, a marginally weak positive linear relationship can be found below 0.3

- Developer versus Architect = 0.0111
    - With a R-Value of 0.01, no linear relationship was found
- Business Analyst versus Project Manager = 0.0036
    - With a R-Value of 0.00, no linear relationship was found
- Business Analyst versus Architect = -0.0509
    - With a R-Value of -0.05, no linear relationship was found
- Project Manager versus Architect = 0.0165
    - With a R-Value of 0.01, no linear relationship was found

It should be noted that a low correlational strength, such as $r = 0.3$ can still be considered statistically significant at $p < 0.01$.

2.2 RQ3: Generations

It was suggested that experts in different generations reason differently when factors that influence software development outcomes were considered. To describe this phenomenon, the following research sub-question (RQ3) was posed:

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts from different generations?*

To allow testing of this question, the following hypotheses were posed:

- H03: There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts from different generations.
- H3: Experts in different generations in a software development team have differing views of "internal variability" and "political impacts" as the factors that influence bespoke software development outcomes in South Africa.

2.2.1 View number 27

Software failure and success including "internal variability" and "political impacts" against expert generation

This view consolidates all research data into a single dataset that represents a combination of failed and successful software development projects as well as "internal variability" and "political impacts" as factor categorisations. This combination represents an expert's perspective of software outcomes, regardless of factors that influence software development. The independent variable of generation was used as a lens to illustrate the relationship of experts in different generations as a secondary viewpoint.

When considering the means of each generation, it could be argued that a difference in perspectives existed between each generation. Table 45 (*Generation – Mean, median, mode and standard deviation*) below shows that the means for each generation differ from between 0.12 and 3.97 percent and Figure 29 (*Generation mean comparison*) at this level paints a picture of this difference.

Table 45 Generation - Mean, median, mode and standard deviation

Generation	Mean	Median	Mode	Standard Deviation
Generation Z (18 - 22)	6.44835	6.17045	6.47727	1.10984
Early-Mid Generation Y (23 - 30)	6.70984	6.81818	6.00000	1.58034
Late Generation Y - Early Generation X (31 - 40)	6.66396	6.72727	6.61364	1.39645
Mid - Late Generation X (41 - 50)	6.73570	6.81818	7.65909	1.47086
Late Generation X - Early Boomers (51 - 65)	6.72790	6.72727	7.75000	1.22761

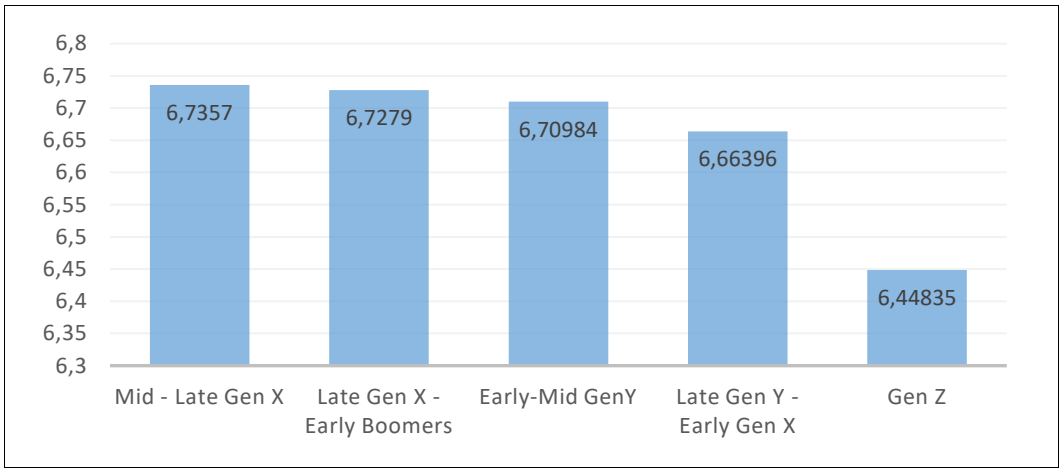


Figure 29: Generation mean comparison

2.2.2 Generation claims

In terms of what was presented in Figure 29 (*Generation mean comparison*), the following claims were made:

- When considering Generation Z and Early-Mid Generation Y, there appears to be a difference of 3.97% between perspective
- When considering Generation Z and Late Generation Y - Early Generation X, there appears to be a difference of 3.29% between perspective
- When considering Generation Z and Mid - Late Generation X, there appears to be a difference of 4.36% between perspective
- When considering Generation Z and Late Generation X - Early Boomers, there appears to be a difference of 4.24% between perspective
- When considering Early-Mid Generation Y and Late Generation Y - Early Generation X, there appears to be a difference of 0.69% between perspective
- When considering Early-Mid Generation Y and Mid - Late Generation X, there appears to be a difference of 0.38% between perspective
- When considering Early-Mid Generation Y and Late Generation X - Early Boomers, there appears to be a difference of 0.27% between perspective
- When considering Late Generation Y - Early Generation X and Mid - Late Generation X, there appears to be a difference of 1.07% between perspective

- When considering Late Generation Y - Early Generation X and Late Generation X - Early Boomers, there appears to be a difference of 0.95% between perspective
- When considering Mid - Late Generation X and Late Generation X - Early Boomers, there appears to be a difference of 0.12% between perspective

2.2.3 Hypothesis testing

Hypothesis testing was done to describe the validity of claims made about the research data. The previous section described differences between each generation in the way questions were answered by the research sample. From a pragmatic perspective, these claims shed some light in terms of what the data is saying, but without statistical examination, little relevance could be proven. As such hypothesis testing was included to assist in describing the differences suggested above. Hypothesis testing was done using the standard T-Test as well as the Student's probability test.

To cater for each relationship between the generations defined above, the following null hypotheses were posed to globally represent the null hypothesis, H03.

- $H03_a$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{early-mid generation y (23 - 30)}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of generation z (18 - 22) and early-mid generation y (23 - 30).

$H03_b$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{late generation y - early generation x (31 - 40)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of generation z (18 - 22) and late generation y - early generation x (31 - 40).

$H03_c$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of generation z (18 - 22) and mid - late generation x (41 - 50).

$H03_d$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of generation z (18 - 22) and late generation x - early boomers (51 - 65).

$H03_e$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{late generation y - early generation x (31 - 40)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of early-mid generation y (23 - 30) and late generation y - early generation x (31 - 40).

$H03_f$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of early-mid generation y (23 - 30) and mid - late generation x (41 - 50).

$H03_g$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of early-mid generation y (23 - 30) and late generation x - early boomers (51 - 65).

$H03_h$: $\mu_{\text{late generation y - early generation x (31 - 40)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of late generation y - early generation x (31 - 40) and mid - late generation x (41 - 50).

$H03_i$: $\mu_{\text{late generation y - early generation x (31 - 40)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of late generation y - early generation x (31 - 40) and late generation x - early boomers (51 - 65).

$H03_j$: $\mu_{\text{mid - late generation x (41 - 50)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of mid - late generation x (41 - 50) and late generation x - early boomers (51 - 65).

2.2.3.1 *P-Value & T-Test*

The P-Value test is commonly used for hypothesis testing as it describes the validity of claims made about data. For this research, the values calculated from the P-Value calculation were interpreted with the following in mind.

If a null hypothesis suggested that there was no difference between role and a P-Value was calculated, then the null-hypothesis would be tested against the following:

- **P-Values < 0.01**: *Suggests that strong evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- P-Values < 0.05: *Suggests that sufficient evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- P-Values > 0.06: *Suggests that little evidence can be found that there is a difference between means. The null hypothesis cannot be rejected.*

To calculate the P-Value, the t-Value was required. The standard t-value equation was used to calculate all t-Values for each generation combination (*Table 45 Generation – Mean, median, mode and standard deviation*).

Table 46 (*Generation t-Value*) provides a cross-tabulation of all t-Values calculated for this dimension.

Table 46 Generation t-Value

t-Value	A	B	C	D	E
A	-	3.0712	3.4584	6.1261	11.0187
B	-3.0712	-	-0.4434	0.2722	0.2090
C	-3.4584	0.4434	-	0.9537	0.9984
D	-6.1261	-0.2722	-0.9537	-	0.1586
E	-11.0187	-0.2090	-0.9984	0.1586	-

Table legend: *A = Generation Z (18 - 22), B = Early-Mid Generation Y (23 - 30), C = Late Generation Y - Early Generation X (31 - 40), D = Mid - Late Generation X (41 - 50), E = Late Generation X - Early Boomers (51 - 65)*

Table 47 (*Generation P-Value*) provides a cross-tabulation of all P-Values calculated for each generation combination.

Table 47 Generation P-Value

P-Value	A	B	C	D	E
A	-	0.0011	0.0003	0.0000	0.0000
B	0.0011	-	0.3288	0.3928	0.4172
C	0.0003	0.3288	-	0.1703	0.1592
D	0.0000	0.3928	0.1703	-	0.4370
E	0.0000	0.4172	0.1592	0.4370	-

Table legend: ***A = Generation Z (18 - 22), B = Early-Mid Generation Y (23 - 30), C = Late Generation Y - Early Generation X (31 - 40), D = Mid - Late Generation X (41 - 50), E = Late Generation X - Early Boomers (51 - 65)***

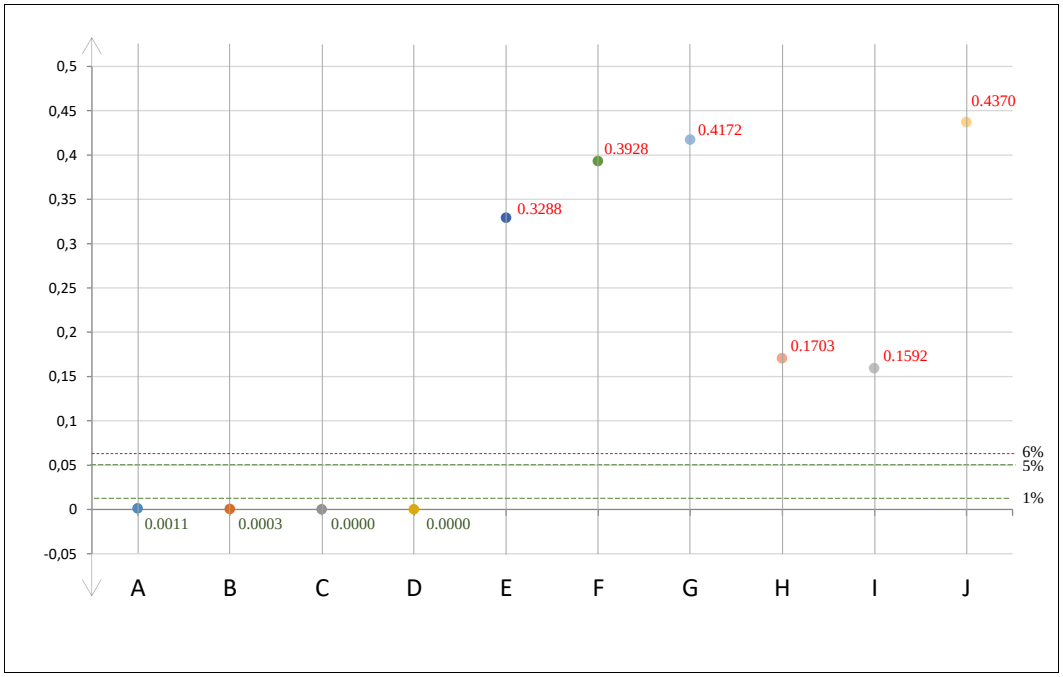


Figure 30: Generation P-Value view

Graph legend: *A = Generation Z -Early-Mid Generation Y , B = Generation Z -Late Generation Y - Early Generation X, C = Generation Z -Mid - Late Generation X , D = Generation Z -Late Generation X - Early Boomers , E = Early-Mid Generation Y -Late Generation Y - Early Generation X , F = Early-Mid Generation Y -Mid - Late Generation X , G = Early-Mid Generation Y -Late Generation X - Early Boomers , H = Late Generation Y - Early Generation X -Mid - Late Generation X , I = Late Generation Y - Early Generation X -Late Generation X - Early Boomers , J = Mid - Late Generation X -Late Generation X - Early Boomers*

2.2.3.1.1 *Outcomes:*

In considering each generation combination and the null hypotheses presented above, the following was found using the standard T-Test:

- $H03_a$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{early-mid generation y (23 - 30)}} = 0.0011$

- With a P-Value of 0.11%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_b$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{late generation y - early generation x (31 - 40)}} = 0.0003$

- With a P-Value of 0.03%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_c$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0.0000$

- With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_d$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.0000$

- With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_e$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{late generation y - early generation x (31 - 40)}} = 0.3288$

- With a P-Value of 32.88%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_f$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0.3928$

- With a P-Value of 39.28%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_g$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.4172$

- With a P-Value of 41.72%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_h$: $\mu_{\text{late generation y - early generation x (31 - 40)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0.1703$

- With a P-Value of 17.03%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_i$: $\mu_{\text{late generation y - early generation x (31 - 40)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.1592$

- With a P-Value of 15.92%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_j$: $\mu_{\text{mid - late generation x (41 - 50)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.437$

- With a P-Value of 43.70%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

2.2.3.2 *Student's Probability*

Like the P-Value test, the Student's probability test was used for hypothesis testing. The probability test calculated a cumulative two-tailed density using ACM algorithm 209. The interpretations of the probability outcomes were considered in a similar way than that of the T-Test. These included the following:

If a null hypothesis suggested that there was no difference between role and a probability was calculated, then the null-hypothesis would be tested against the following:

- **probability < 0.01**: *Suggests that strong evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- **probability** < 0.05: *Suggests that sufficient evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- probability > 0.06: *Suggests that some evidence can be found that there is a difference between means. The null hypothesis cannot be rejected.*

The following degrees of freedom were calculated for use in the Student's probability test:

Table 48 Degree of freedom

df	A	B	C	D	E
A	-	376.620	396.314	430.782	638.824
B	376.620	-	646.259	540.617	398.163
C	396.314	646.259	-	652.059	436.876
D	430.782	540.617	652.059	-	501.635
E	638.824	398.163	436.876	501.635	-

Table legend: *A = Generation Z (18 - 22), B = Early-Mid Generation Y (23 - 30), C = Late Generation Y - Early Generation X (31 - 40), D = Mid - Late Generation X (41 - 50), E = Late Generation X - Early Boomers (51 - 65)*

Table 49 (*Probability test*) provides a cross-tabulation of all probability values calculated for each role combination.

Table 49 Probability test

Probability	A	B	C	D	E
A	-	0.0022866947	0.0006022471	0.0000000020	0.0000000000
B	0.0022866947	-	0.6576153576	0.7855710087	0.8345294680
C	0.0006022471	0.6576153576	-	0.3405990727	0.3186298591
D	0.0000000020	0.7855710087	0.3405990727	-	0.8740576008
E	0.0000000000	0.8345294680	0.3186298591	0.8740576008	-

Table legend: *A = Generation Z (18 - 22), B = Early-Mid Generation Y (23 - 30), C = Late Generation Y - Early Generation X (31 - 40), D = Mid - Late Generation X (41 - 50), E = Late Generation X - Early Boomers (51 - 65)*

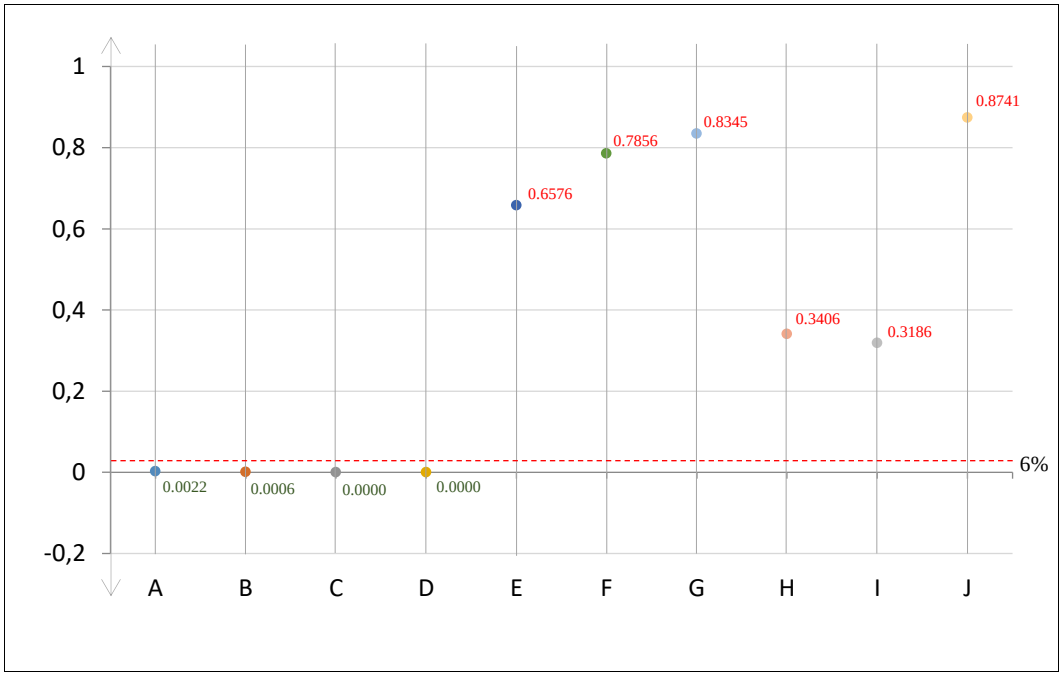


Figure 31: Generation Probability test view

Graph legend: *A = Generation Z -Early-Mid Generation Y , B = Generation Z -Late Generation Y - Early Generation X, C = Generation Z -Mid - Late Generation X , D = Generation Z -Late Generation X - Early Boomers , E = Early-Mid Generation Y -Late Generation Y - Early Generation X , F = Early-Mid Generation Y -Mid - Late Generation X , G = Early-Mid Generation Y -Late Generation X - Early Boomers , H = Late Generation Y - Early Generation X -Mid - Late Generation X , I = Late Generation Y - Early Generation X -Late Generation X - Early Boomers , J = Mid - Late Generation X -Late Generation X - Early Boomers*

2.2.3.2.1 *Outcomes:*

In considering each role combination and the null hypotheses presented above, the following was found using the Student's probability test:

- $H03_a$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{early-mid generation y (23 - 30)}} = 0.0023$
    - With a P-Value of 0.23%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_b$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{late generation y - early generation x (31 - 40)}} = 0.0006$

- With a P-Value of 0.06%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_c$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0.0000$

- With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_d$: $\mu_{\text{generation z (18 - 22)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.0000$

- With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.

$H03_e$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{late generation y - early generation x (31 - 40)}} = 0.6576$

- With a P-Value of 65.76%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_f$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0.7856$

- With a P-Value of 78.56%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_g$: $\mu_{\text{early-mid generation y (23 - 30)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.8345$

- With a P-Value of 83.45%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_h$: $\mu_{\text{late generation y - early generation x (31 - 40)}} - \mu_{\text{mid - late generation x (41 - 50)}} = 0.3406$

- With a P-Value of 35.06%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_i$: $\mu_{\text{late generation y - early generation x (31 - 40)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.3186$

- With a P-Value of 31.86%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

$H03_j$: $\mu_{\text{mid - late generation x (41 - 50)}} - \mu_{\text{late generation x - early boomers (51 - 65)}} = 0.8741$

- With a P-Value of 87.41%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

2.2.3.3 *Pearson correlation coefficient*

The Pearson correlation method is a parametric correlation test that depends on a distribution of data. The method was used to measure the linear dependence between each generation relationship or the strength and direction of a linear relationship between two variables. The Pearson correlation formula produces a R-Value that is between + 1 and –1, where numbers greater than 0 are considered positive relational and numbers smaller than 0 negative relational. The closer the number is to 0, the greater the variation is between data points. More variation means weaker relationships between data sets.

The following basic formula was used to determine the correlation between different generations:

$$r = \frac{\sum(x - \mu_x)(y - \mu_y)}{\sqrt{\sum(x - \mu x)^2 (y - \mu y)^2}}$$

The value of a R-Value can be understood in terms of one of the following:

- Exactly –1 - A perfect downhill or negative linear relationship
- –0.70 - A strong downhill or negative linear relationship
- –0.50 - A moderate downhill or negative relationship
- –0.30 - A weak downhill or negative linear relationship
- 0 - No linear relationship
- + 0.30 - A weak uphill or positive linear relationship
- + 0.50 - A moderate uphill or positive relationship
- + 0.70 - A strong uphill or positive linear relationship
- Exactly + 1 - A perfect uphill or positive linear relationship

For this study, the values of +0.5 and -0.5 was considered appropriate as the cut-off mark to explain a relationship. As such, any value greater to +0.5 and smaller

than -0.5 was considered to have a relationship. Similarly, any value between +0.5 and -0.5 was considered to not have a relationship.

Table 50 Generation r-Value

r-Value	A	B	C	D	E
A	-	-0.0863	-0.0053	-0.0782	-0.0157
B	-0.0863	-	0.1613	-0.0149	0.0315
C	-0.0053	0.1613	-	0.0244	-0.1401
D	-0.0782	-0.0149	0.0244	-	0.0021
E	-0.0157	0.0315	-0.1401	0.0021	-

Table legend: *A = Generation Z, B = Early-Mid Generation Y, C = Late Generation Y - Early Generation X, D = Mid - Late Generation X, E = Late Generation X - Early Boomers*

Table 50 (*Generation r-Value*) provides a cross-tabulation of all R-values calculated for this dimension.

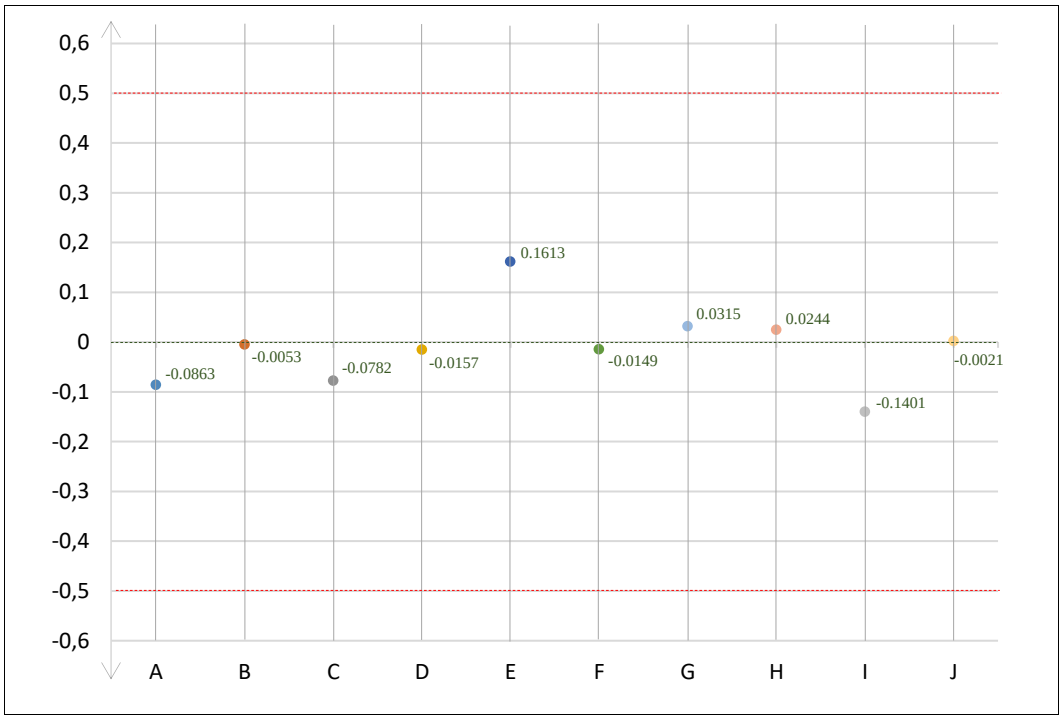


Figure 32: Generation R-Value view

Graph legend: *A = Generation Z -Early-Mid Generation Y , B = Generation Z -Late Generation Y - Early Generation X, C = Generation Z -Mid - Late Generation X , D = Generation Z -Late Generation X - Early Boomers , E = Early-Mid Generation Y -Late Generation Y - Early Generation X , F = Early-Mid Generation Y -Mid - Late Generation X , G = Early-Mid Generation Y -Late Generation X - Early Boomers , H = Late Generation Y - Early Generation X -Mid - Late Generation X , I = Late Generation Y - Early Generation X -Late Generation X - Early Boomers , J = Mid - Late Generation X -Late Generation X - Early Boomers*

2.2.3.3.1 *Outcomes:*

This dimension proposed ten different relationships, all related to a combination of each generation. In considering the validity of each relationship, the following outcomes were found:

- Generation Z versus Early-Mid Generation Y = -0.0863
    - With a R-Value of -0.08, no linear relationship was found
- Generation Z versus Late Generation Y - Early Generation X = -0.0053
    - With a R-Value of -0.00, no linear relationship was found
- Generation Z versus Mid - Late Generation X = -0.0782
    - With a R-Value of -0.08, no linear relationship was found
- Generation Z versus Late Generation X - Early Boomers = -0.0157

    - With a R-Value of -0.02, no linear relationship was found
- Early-Mid Generation Y versus Late Generation Y - Early Generation X = 0.1613
    - With a R-Value of 0.16, a marginally weak positive linear relationship can be found below 0.3
- Early-Mid Generation Y versus Mid - Late Generation X = -0.0149
    - With a R-Value of -0.01, no linear relationship was found
- Early-Mid Generation Y versus Late Generation X - Early Boomers = 0.0315
    - With a R-Value of 0.03, no linear relationship was found
- Late Generation Y - Early Generation X versus Mid - Late Generation X = 0.0244
    - With a R-Value of 0.02, no linear relationship was found
- Late Generation Y - Early Generation X versus Late Generation X – Early Boomers = -0.1401
    - With a R-Value of -0.14, no linear relationship was found
- Mid - Late Generation X versus Late Generation X - Early Boomers = 0.0021
    - With a R-Value of 0.00, no linear relationship was found

It should be noted that a low correlational strength, such as $r = 0.3$ can still be considered statistically significant at $p < 0.01$.

2.3 RQ4: Tenure

It was suggested that experts with different tenure reason differently when factors that influence software development outcomes were considered. To describe this phenomenon, the following research sub-question (RQ4) was posed:

> *What difference in recognising the factors that influence bespoke software development outcomes in South Africa, exists between software development experts of different tenure?*

To allow testing of this question, the following hypotheses were posed:

- H04: There is no significant difference in how the factors that influence bespoke software development outcomes in South Africa are recognised among software development experts with different tenure.
- H4: Experts with different tenure in a software development team have differing views of “internal variability” and “political impacts” as the factors that influence bespoke software development project outcomes in South Africa.

2.3.1 View number 26

Software failure and success including “internal variability” and “political impacts” against expert tenure

This view consolidates all research data into a single dataset that represents a combination of failed and successful software development projects as well as “internal variability” and “political impacts” as factor categorisations. This combination represents an expert's perspective of software outcomes, regardless of factors that influence software development. The independent variable of tenure was used as a lens to illustrate the relationship of experts with different tenure as a secondary viewpoint.

When considering the means of each role, it could be argued that a difference in perspectives existed between each tenure. Table 51 (*Tenure – Mean, median, mode and standard deviation*) below shows that the means for each tenure differ from between 0.15 and 7.67 percent and Figure 33 (*Tenure mean comparison*) at this level paints a picture of this difference.

Table 51 Tenure - Mean, median, mode and standard deviation

Tenure	Mean	Median	Mode	Standard Deviation
Learner level	6.67262	6.53409	6.97727	1.34788
Junior level	6.36913	6.42045	6.43182	1.70713
Junior to intermediate level	6.66245	6.71591	6.00000	1.56255

Intermediate to Senior level	6.87692	6.90909	7.75000	1.28177
Senior level	6.69096	6.80682	7.61364	1.44135

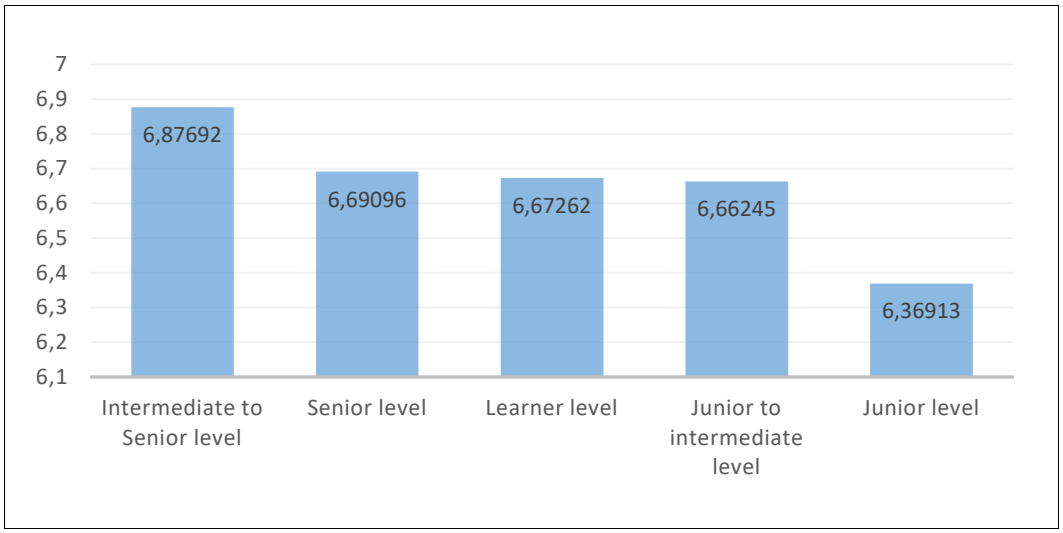


Figure 33: Tenure mean comparison

2.3.2 Tenure claims

In terms of what was presented in Figure 33 (*Tenure mean comparison*), the following claims were made:

- When considering Learner level - Junior level, there appears to be a difference of 4.65% between perspective
- When considering Learner level - Junior to intermediate level, there appears to be a difference of 0.15% between perspective
- When considering Learner level - Intermediate to Senior level, there appears to be a difference of 3.02% between perspective
- When considering Learner level - Senior level, there appears to be a difference of 0.27% between perspective
- When considering Junior Level - Junior to intermediate level, there appears to be a difference of 4.50% between perspective

- When considering Junior level - Intermediate to Senior level, there appears to be a difference of 7.67% between perspective
- When considering Junior level - Senior level, there appears to be a difference of 4.93% between perspective
- When considering Junior to intermediate level - Intermediate to Senior level, there appears to be a difference of 3.17% between perspective
- When considering Junior to intermediate level - Senior level, there appears to be a difference of 0.43% between perspective
- When considering Intermediate to Senior level - Senior level, there appears to be a difference of 2.74% between perspective

2.3.3 Hypothesis testing

Hypothesis testing was done to describe the validity of claims made about the research data. The previous section described differences between each tenure in the way questions were answered by the research sample. From a pragmatic perspective, these claims shed some light in terms of what the data is saying, but without statistical examination, little relevance could be proven. As such hypothesis testing was included to assist in describing the differences suggested above. Hypothesis testing was done using the standard T-Test as well as the Student's probability test.

To cater for each relationship between the tenure defined above, the following null hypotheses were posed to globally represent the null hypothesis, H04.

- $H04_a$: $\mu_{\text{learner level}} - \mu_{\text{junior level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of learner level and junior level.
- $H04_b$: $\mu_{\text{learner level}} - \mu_{\text{junior to intermediate level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of learner level and junior to intermediate level.
- $H04_c$: $\mu_{\text{learner level}} - \mu_{\text{intermediate to senior level}} = 0$

When considering the outcomes of software development projects, there is no difference between the means of learner level and intermediate to senior level.

- $H04_d$: $\mu_{\text{learner level}} - \mu_{\text{senior level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of learner level and senior level.

- $H04_e$: $\mu_{\text{junior level}} - \mu_{\text{junior to intermediate level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of junior level and junior to intermediate level.

- $H04_f$: $\mu_{\text{junior level}} - \mu_{\text{intermediate to senior level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of junior level and intermediate to senior level.

- $H04_g$: $\mu_{\text{junior level}} - \mu_{\text{senior level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of junior level and senior level.

- $H04_h$: $\mu_{\text{junior to intermediate level}} - \mu_{\text{intermediate to senior level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of junior to intermediate level and intermediate to senior level.

- $H04_i$: $\mu_{\text{junior to intermediate level}} - \mu_{\text{senior level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of junior to intermediate level and senior level.

- $H04_j$: $\mu_{\text{intermediate to senior level}} - \mu_{\text{senior level}} = 0$

  When considering the outcomes of software development projects, there is no difference between the means of intermediate to senior level and senior level.

2.3.3.1 *P-Value & T-Test*

The P-Value test is commonly used for hypothesis testing as it describes the validity of claims made about data. For this research, the values calculated from the P-Value calculation were interpreted with the following in mind.

If a null hypothesis suggested that there was no difference between role and a P-Value was calculated, then the null-hypothesis would be tested against the following:

- **P-Values < 0.01**: *Suggests that strong evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- P-Values < 0.05: *Suggests that sufficient evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- P-Values > 0.06: *Suggests that little evidence can be found that there is a difference between means. The null hypothesis cannot be rejected.*

To calculate the P-Value, the t-Value was required. The standard t-value equation was used to calculate all t-Values for each tenure combination (*Table 51 Tenure – Mean, median, mode and standard deviation*).

Table 52 (*Tenure t-Value*) provides a cross-tabulation of all t-Values calculated for this dimension.

Table 52 Tenure t-Value

t-Value	A	B	C	D	E
A	-	-5.7456	-0.1078	3.3189	0.2542
B	5.7456	-	2.9303	7.2375	4.0427
C	0.1078	-2.9303	-	2.0431	0.2556
D	-3.3189	-7.2375	-2.0431	-	-2.1711
E	-0.2542	-4.0427	-0.2556	2.1711	-

Table legend: *A = Learner level, B = Junior level, C = Junior to intermediate level, D = Intermediate to Senior level, E = Senior level*

Table 53 (*Tenure P-Value*) provides a cross-tabulation of all P-Values calculated for each tenure combination.

Table 53 Tenure P-Value

P-Value	A	B	C	D	E
A	-	0.0000	0.4571	0.0005	0.3997
B	0.0000	-	0.0018	0.0000	0.0000
C	0.4571	0.0018	-	0.0207	0.3992
D	0.0005	0.0000	0.0207	-	0.0152
E	0.3997	0.0000	0.3992	0.0152	-

Table legend: *A = Learner level, B = Junior level, C = Junior to intermediate level, D = Intermediate to Senior level, E = Senior level*

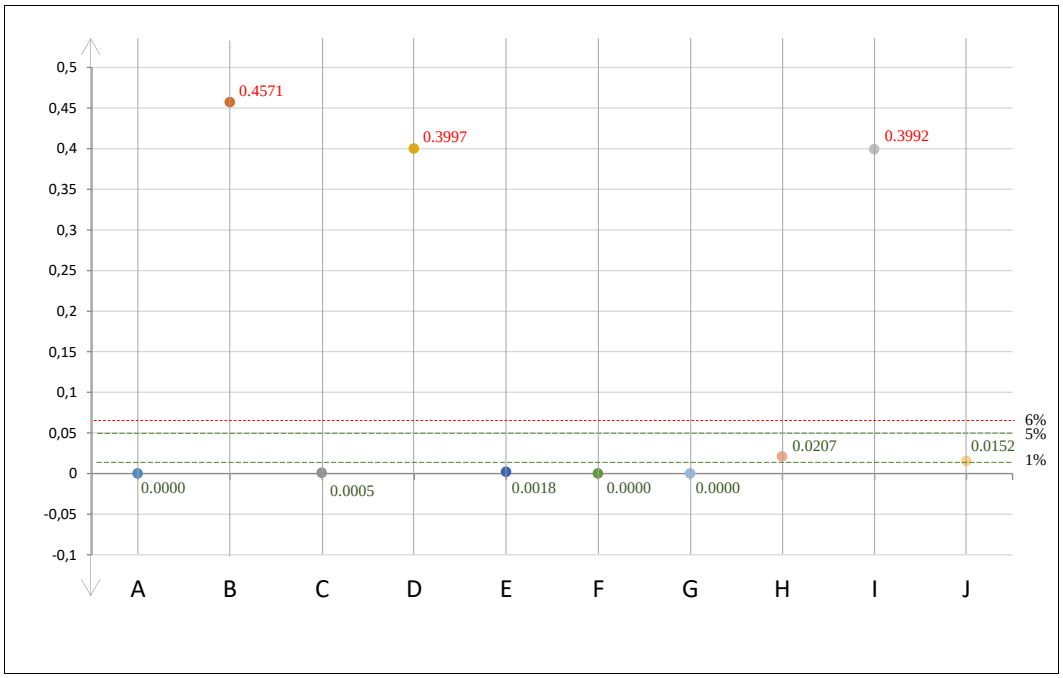


Figure 34: Tenure P-Value view

Graph legend: *A= Learner level - Junior level, B = Learner level - Junior to intermediate level, C = Learner level - Intermediate to Senior level, D = Learner level - Senior level, E = Junior Level - Junior to intermediate level, F = Junior level - Intermediate to Senior level, G = Junior level - Senior level, H = Junior to intermediate level - Intermediate to Senior level, I = Junior to intermediate level - Senior level, J = Intermediate to Senior level - Senior level*

2.3.3.1.1 *Outcomes:*

In considering each tenure combination and the null hypotheses presented above, the following was found using the standard T-Test:

- $H04_a$: $\mu_{\text{learner level}} - \mu_{\text{junior level}} = 0.0000$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_b$: $\mu_{\text{learner level}} - \mu_{\text{junior to intermediate level}} = 0.4571$
  - With a P-Value of 45.71%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.
- $H04_c$: $\mu_{\text{learner level}} - \mu_{\text{intermediate to senior level}} = 0.0005$
  - With a P-Value of 0.05%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_d$: $\mu_{\text{learner level}} - \mu_{\text{senior level}} = 0.3997$

- With a P-Value of 39.77%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

- $H04_e$: $\mu_{\text{junior level}} - \mu_{\text{junior to intermediate level}} = 0.0018$
  - With a P-Value of 0.18%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_f$: $\mu_{\text{junior level}} - \mu_{\text{intermediate to senior level}} = 0.0000$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_g$: $\mu_{\text{junior level}} - \mu_{\text{senior level}} = 0.0000$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_h$: $\mu_{\text{junior to intermediate level}} - \mu_{\text{intermediate to senior level}} = 0.0207$
  - With a P-Value of 2.07%, a difference can be found at 5%. As such, this hypothesis was rejected.
- $H04_i$: $\mu_{\text{junior to intermediate level}} - \mu_{\text{senior level}} = 0.3992$
  - With a P-Value of 39.92%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.
- $H04_j$: $\mu_{\text{intermediate to senior level}} - \mu_{\text{senior level}} = 0.0152$
  - With a P-Value of 1.52%, a difference can be found at 5%. As such, this hypothesis was rejected.

2.3.3.2 *Student's Probability*

Like the P-Value test, the Student's probability test was used for hypothesis testing. The probability test calculated a cumulative two-tailed density using ACM algorithm 209. The interpretations of the probability outcomes were considered in a similar way than that of the T-Test. These included the following:

If a null hypothesis suggested that there was no difference between role and a probability was calculated, then the null-hypothesis would be tested against the following:

- **probability < 0.01**: *Suggests that strong evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- probability < 0.05: *Suggests that sufficient evidence can be found that there is a difference between means. The null hypothesis must be rejected.*
- probability > 0.06: *Suggests that some evidence can be found that there is a difference between means. The null hypothesis cannot be rejected.*

The following degrees of freedom were calculated for use in the Student's probability test:

Table 54 Degree of freedom

df	A	B	C	D	E
A	-	104.724	364.648	460.241	413.762
B	104.724	-	442.268	582.017	529.213
C	364.648	442.268	-	499.182	557.129
D	460.241	582.017	499.182	-	584.275
E	413.762	529.213	557.129	584.275	-

Table legend: *A = Learner level, B = Junior level, C = Junior to intermediate level, D = Intermediate to Senior level, E = Senior level*

Table 56 (*Probability test*) provides a cross-tabulation of all probability values calculated for each role combination.

Table 55 Probability test

Probability	A	B	C	D	E
A	-	0.2469769296	0.9141983519	0.0009755912	0.7994726011
B	0.2469769296	-	0.0035612130	0.0000000000	0.0000606929
C	0.9141983519	0.0035612130	-	0.0415701356	0.7983537884
D	0.0009755912	0.0000000000	0.0415701356	-	0.0303240420
E	0.7994726011	0.0000606929	0.7983537884	0.0303240420	-

Table legend: *A = Learner level, B = Junior level, C = Junior to intermediate level, D = Intermediate to Senior level, E = Senior level*

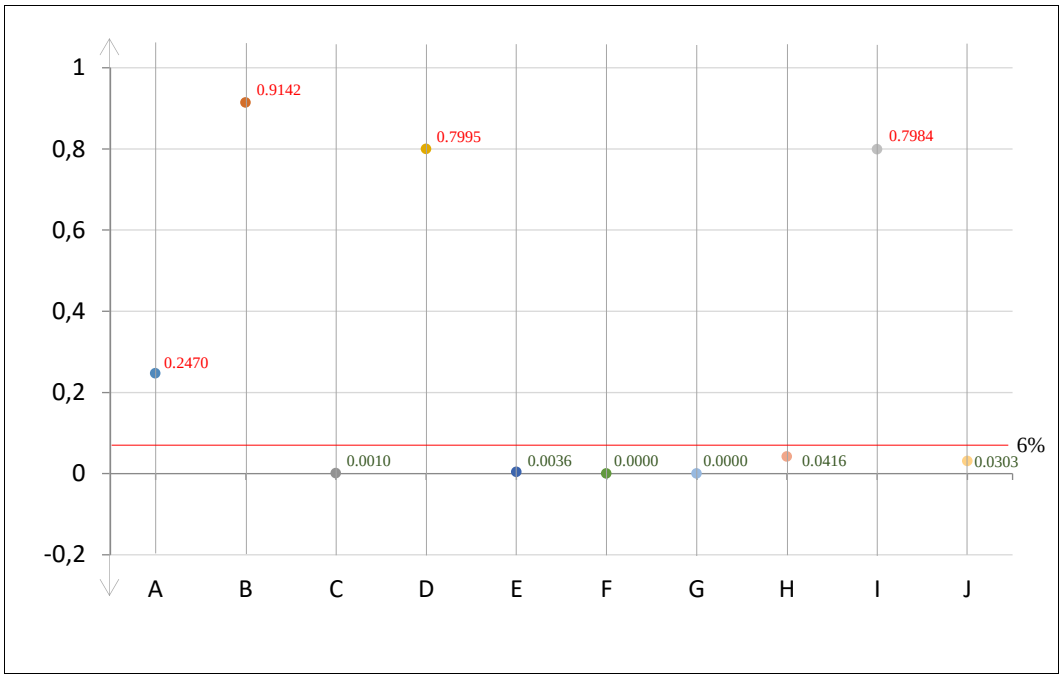


Figure 35: Tenure Probability test view

Graph legend: *A= Learner level - Junior level, B = Learner level - Junior to intermediate level, C = Learner level - Intermediate to Senior level, D = Learner level - Senior level, E = Junior Level - Junior to intermediate level, F = Junior level - Intermediate to Senior level, G = Junior level - Senior level, H = Junior to intermediate level - Intermediate to Senior level, I = Junior to intermediate level - Senior level, J = Intermediate to Senior level - Senior level*

2.3.3.2.1 *Outcomes:*

In considering each tenure combination and the null hypotheses presented above, the following was found using the standard T-Test:

- $H04_a$: $\mu_{\text{learner level}} - \mu_{\text{junior level}} = 0.2470$
  - With a P-Value of 24.70%%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.
- $H04_b$: $\mu_{\text{learner level}} - \mu_{\text{junior to intermediate level}} = 0.9142$
  - With a P-Value of 91.42%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.
- $H04_c$: $\mu_{\text{learner level}} - \mu_{\text{intermediate to senior level}} = 0.0010$
  - With a P-Value of 0.10%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_d$: $\mu_{\text{learner level}} - \mu_{\text{senior level}} = 0.7995$

- With a P-Value of 79.95%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.

- $H04_e$: $\mu_{\text{junior level}} - \mu_{\text{junior to intermediate level}} = 0.0036$
  - With a P-Value of 0.36%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_f$: $\mu_{\text{junior level}} - \mu_{\text{intermediate to senior level}} = 0.0000$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_g$: $\mu_{\text{junior level}} - \mu_{\text{senior level}} = 0.0001$
  - With a P-Value of 0.00%, a difference can be found at 1%. As such, this hypothesis was rejected.
- $H04_h$: $\mu_{\text{junior to intermediate level}} - \mu_{\text{intermediate to senior level}} = 0.0416$
  - With a P-Value of 4.16%, a difference can be found at 5%. As such, this hypothesis was rejected.
- $H04_i$: $\mu_{\text{junior to intermediate level}} - \mu_{\text{senior level}} = 0.7984$
  - With a P-Value of 79.84%, there is not enough evidence to demonstrate a difference. As such, this hypothesis cannot be rejected.
- $H04_j$: $\mu_{\text{intermediate to senior level}} - \mu_{\text{senior level}} = 0.0303$
  - With a P-Value of 3.03%, a difference can be found at 5%. As such, this hypothesis was rejected.

2.3.3.3 *Pearson correlation coefficient*

The Pearson correlation method is a parametric correlation test that depends on a distribution of data. The method was used to measure the linear dependence between each generation relationship or the strength and direction of a linear relationship between two variables. The Pearson correlation formula produces a R-Value that is between + 1 and –1, where numbers greater than 0 are considered positive relational and numbers smaller than 0 negative relational. The closer the number is to 0, the greater the variation is between data points. More variation means weaker relationships between data sets.

The following basic formula was used to determine the correlation between different tenure:

$$r = \frac{\sum(x - \mu_x)(y - \mu_y)}{\sqrt{\sum(x - \mu x)^2 (y - \mu y)^2}}$$

The value of a R-Value can be understood in terms of one of the following:

- Exactly –1 - A perfect downhill or negative linear relationship
- –0.70 - A strong downhill or negative linear relationship
- –0.50 - A moderate downhill or negative relationship
- –0.30 - A weak downhill or negative linear relationship
- 0 - No linear relationship
- + 0.30 - A weak uphill or positive linear relationship
- + 0.50 - A moderate uphill or positive relationship
- + 0.70 - A strong uphill or positive linear relationship
- Exactly + 1 - A perfect uphill or positive linear relationship

For this study, the values of +0.5 and -0.5 was considered appropriate as the cut-off mark to explain a relationship. As such, any value greater to +0.5 and smaller than -0.5 was considered to have a relationship. Similarly, any value between +0.5 and -0.5 was considered to not have a relationship.

Table 56 Tenure r-Value

r-Value	A	B	C	D	E
A	-	0.1484	0.0146	0.0420	-0.0389
B	0.1484	-	0.1178	0.0385	0.0353
C	0.0146	0.1178	-	0.1426	0.1058
D	0.0420	0.0385	0.1426	-	0.0771
E	-0.0389	0.0353	0.1058	0.0771	-

Table legend: *A = Learner level, B = Junior level, C = Junior to intermediate level, D = Intermediate to Senior level, E = Senior level*

Table 56 (*Tenure r-Value*) provides a cross-tabulation of all R-values calculated for this dimension.

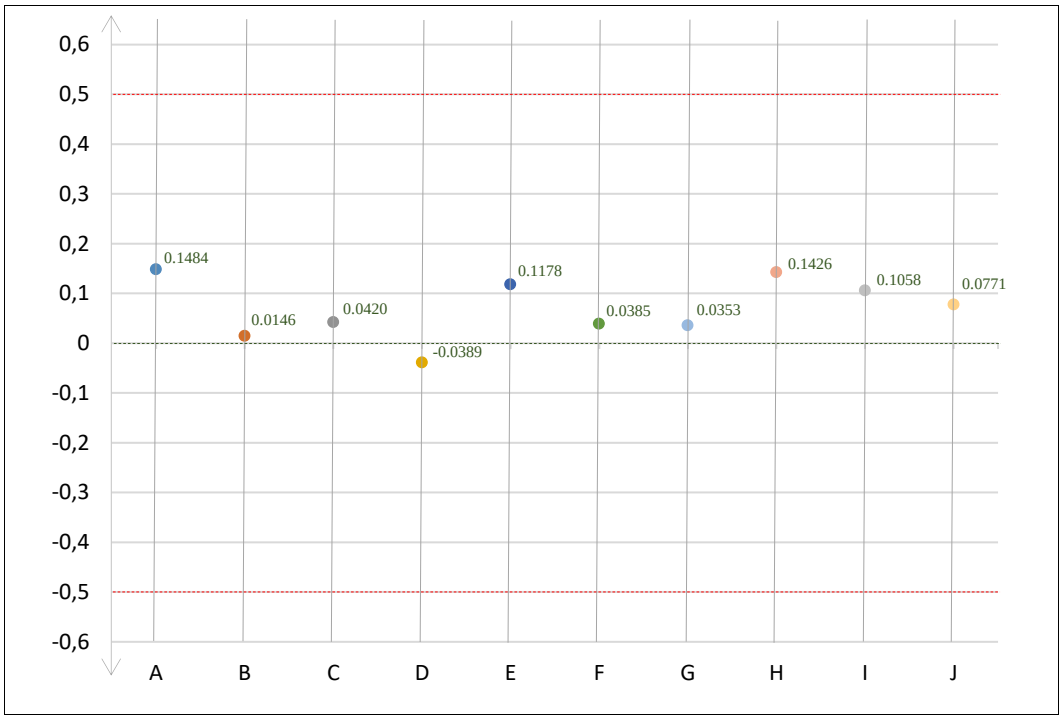


Figure 36: Tenure R-Value view

Graph legend: *A= Learner level - Junior level, B = Learner level - Junior to intermediate level, C = Learner level - Intermediate to Senior level, D = Learner level - Senior level, E = Junior Level - Junior to intermediate level, F = Junior level - Intermediate to Senior level, G = Junior level - Senior level, H = Junior to intermediate level - Intermediate to Senior level, I = Junior to intermediate level - Senior level, J = Intermediate to Senior level - Senior level*

2.3.3.3.1 *Outcomes:*

This dimension proposed ten different relationships, all related to a combination of each tenure. In considering the validity of each relationship, the following outcomes were found:

- Learner level - Junior level = 0.1484
    - With a R-Value of 0.15, a marginally weak positive linear relationship can be found below 0.3
- Learner level - Junior to intermediate level = 0.0146
    - With a R-Value of 0.01, no linear relationship was found
- Learner level - Intermediate to Senior level = 0.0420
    - With a R-Value of 0.04, no linear relationship was found
- Learner level - Senior level = -0.0389
    - With a R-Value of -0.04, no linear relationship was found
- Junior Level - Junior to intermediate level = 0.1178
    - With a R-Value of 0.12, a marginally weak positive linear relationship can be found below 0.3
- Junior level - Intermediate to Senior level = 0.0385
    - With a R-Value of 0.04, no linear relationship was found
- Junior level - Senior level = 0.0353
    - With a R-Value of 0.04, no linear relationship was found
- Junior to intermediate level - Intermediate to Senior level = 0.1426
    - With a R-Value of 0.14, a marginally weak positive linear relationship can be found below 0.3
- Junior to intermediate level - Senior level = 0.1058
    - With a R-Value of 0.11, a marginally weak positive linear relationship can be found below 0.3
- Intermediate to Senior level - Senior level = 0.0771
    - With a R-Value of 0.08, no linear relationship was found

It should be noted that a low correlational strength, such as $r = 0.3$ can still be considered statistically significant at $p < 0.01$.